

Botnet

# “双头龙”源自海莲花组织？



Alex.Turing

May 6, 2021 • 6 min read

我们的[双头龙](#)blog发布后引起了较大反响，除了媒体转载，一些安全同行还纷纷在我们blog下面留言和提问，其中5月4号的一则留言提到双头龙跟海莲花（OceanLotus）样本的C2行为有联系：



PE • 20 hours ago

Wow. Amazing work. Here's a real comparison for you Jerry... looks a lot like the C2 activity associated with OceanLotus from 2016...  
[https://www.virustotal.com/...](https://www.virustotal.com/)

[^](#) | [v](#) • [Replv](#) • [Share](#) >

留言所提到的[样本](#)为一个zip打包文件，2016年就已出现。该zip可以解压出多个文件，那个名为[Noi dung chi tiet](#)（对应中文[详细信息](#)）的Mach-O格式可执行文件即是海莲花样本。对比分析显示该样本确实与双头龙样本存在多个相似之处，所以它们或许可以解开双头龙的身世之谜：它极可能是海莲花的Linux版本。本文主要从2进制代码层面介绍这些相似点。

## 相似点1：C2会话建立函数

Linux常见的域名解析函数为gethostbyname()，但双头龙使用了相对小众的getaddrinfo()函数，C2域名的解析和会话建立都在一个函数中完成，而海莲花样本中也存在一个模式类似的相同功能的函数，2者的函数对比如下：

能看出来它们不但功能相同，对  
s  
pr  
in  
tf  
(  
和  
g  
et  
a  
d  
dr  
in  
fo  
(  
的使用方式也几乎一模一样。此外

```
*(_QWORD *)s = 0LL;
v16 = _readfsqword(0x28u);
memset(&req, 0, sizeof(req));
req.ai_family = 2;
v15 = 0;
req.ai_socktype = 1;
sprintf(s, "%d", v2);
if (!getaddrinfo(name, s, &req, &pai))
    return 0LL;
v3 = pai;
if (!pai)
{
    while (1)
{
    v4 = socket(v3->ai_family, v3->ai_socktype, v3->ai_protocol);
    v1 = v4;
    if (v4 != -1)
    {
        if (connect(v4, v3->ai_addr, v3->ai_addrlen) != -1)
        {
            freeaddrinfo(pai);
            optval = 1;
            setsockopt(v1, 6, 1, &optval, 4u);
        }
    }
}
```

RotaJakiro

```
*(_QWORD *)&v6.ai_addr = 0LL;
*(_QWORD *)&v6.ai_addrlen = 0LL;
v6.ai_family = 2;
v6.ai_socktype = 1;
v6.ai_flags = 0;
v6.ai_protocol = 0;
v9 = 0;
*(_QWORD *)v8 = 0LL;
sprintf(v8, "%d", *((unsigned int *)c2_info + 3));
if (!getaddrinfo((const char **)((char *)c2_info + 4), v8, &v6, &v5))
    return 0LL;
for (i = v5; i; i = i->ai_next)
{
    v2 = socket(i->ai_family, i->ai_socktype, i->ai_protocol);
    *(_DWORD *)c2_info = v2;
    if (v2 != -1)
    {
        v3 = 1;
        if (connect(v2, i->ai_addr, i->ai_addrlen) != -1)
            goto LABEL_9;
        close(*(_DWORD *)c2_info);
    }
}
```

OceanLotus

, 双头龙和海莲花都使用了单独的数据结构来保存 C2 会话信息 , 比如 socketfd , 是否 active

e,  
ti  
m  
e  
o  
ut  
等  
,而且它们的数据结构也很相似。

```
*(_DWORD *)v7 = a2;  
v7[4] = 1;  
*((_DWORD *)v7 + 2) = 300;
```

Rotakiro

vs

```
*(_DWORD *)(a1 + 12) = v3;  
*(_DWORD *)(a1 + 16) = 300;  
*(_WORD *)(a1 + 20) = 1;
```

OceanLotus

这些相似性是是2者存在关联的第一个证据。

## 相似点2：上线包构造手法

双头龙和海莲花的网络数据包都是由 Head, Key, Payload 三部分组成，其中 Head 是必须的，长度为82字节，而 Key 和 Payload 则是可选的。Head 中的关键字段包括：

- 偏移1，DWORD类型，存放一个magic；
- 偏移9，DWORD类型，存放Payload长度；
- 偏移13，WORD类型，存放Key长度；

- 偏移15， DWORD类型，存放消息码。

双头龙通过一个单独的函数初始化上线包的Head:

这个函数先调用malloc()函数为上线包动态分配内存，接下来依次调用time()/srand()/rand()函数生成一个随机字符然后赋给上线包的第一个字段，剩下的大片代码就是用多个常量对其余字段逐一赋值，所以该函数最明显的特点就是用多个常量初始化上线包。

海莲花样本中也存在1个专门初始化上线包Head的函数：

该函数没有内存分配和随机生成字符的代码，整个函数都是用多个常量逐一赋值上线包的具体字段，这一点跟双头龙一模一样。此外，在1、24和75这三个偏移处海莲花跟双头龙共享了同样的字段值，尤其是偏移为1处的magic都为 0x3B91011，这很难用巧合来形容，所以大大增加了这两段代码同源的概率。

另外，双头龙和海莲花都为上线包分配了消息码，并且都是0x2170272:

<code>if ( (unsigned int)sub_4046E0((__int64)v5, 0x2170272, 0x3B91011) )</code>	RotaJakiro
VS	
<code>if ( sub_10000316A(*((__QWORD *)v1 + 2), 0x2170272LL, 0x3B91011) )</code>	OceanLotus

最终产生的上线包也非常相似，双头龙上线包如下：

00000000	24 41 61 54 03 55 e2 1c e3 63 63 63 63 63 63 2d	\$AaT.U.. .cccccc-
00000010	23 81 23 3b 67 ef 67 43 3f 63 63 63 63 3b 63 63	#.#;g.gC ?cccc;cc
00000020	63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63	cccccccc cccccccc
00000030	63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63	cccccccc cccccccc
00000040	63 63 7a 63 63 63 63 63 63 63 9c 63 42 63 63	cczcccccc ccc.cBcc
00000050	63 63	cc

下面是 PAN 2017年分析出的海莲花上线包：

双头龙解密后的上线包如下所示：

00000000	3A 11 10 B9 03 B1 0C FB 04 00 00 00 00 00 00 72	...r
00000010	02 17 02 C2 20 64 20 01 E2 00 00 00 00 C2 00 00	...d.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000040	00 00 C8 00 00 00 00 00 00 00 00 FF 00 09 00 00	.....
00000050	00 00	..

cmd      magic      payload len      key len

下面是PAN分析报告里的海莲花明文上线包：

可以看到它们的明文结构一样，关键字段值基本相同。

## 相似点3：均存在rotate函数

双头龙和海莲花都存在一个我们称之为“rotate()”的函数，用于加/解密，双头龙的rotate函数如下：

海莲花版本：

两相对比不难发现它们的共同点：

1. 均接受3个参数。
2. 原型相同，其中第1参数为实际的rotate对象，第2个参数为长度字段，第3个参数起控制作用。

在实际使用中，例如在加密上线包的过程中，可以看到，双头龙和海莲花使用了一样的参数。

```
*(++v8 - 1) = rotete((char)(v10 ^ 0x1B), 3, 1); RotJakiro  
VS  
result = (void *)rotate(*((char *)*a2 + v7) ^ 0x1Bu, 3, 1); OceanLotus
```

## 相似点4：相同的指令码

双头龙和海莲花都用 DWORD类型的指令码 来指定消息的功能，并且共享了多个语义相同的指令码，部分有特色的指令如下表所示：

CMD	FUNCTION
0x18320e0	Upload device Info
0x2170272	Register
0x1B25503	execute function from a plugin
0x1532e65	execute function from a plugin

CMD	FUNCTION
0x25D5082	execute function from a plugin
这种相似显然不能用巧合来解释了，这是它们代码同源的极强证据。	

## 总结

双头龙与MAC版的海莲花木马虽然是用不同语言实现的，但是它们在功能和消息格式设计上的相似，在具体实现上的雷同，已经不能用巧合来解释了。从目前的线索来看，双头龙极大可能是由海莲化组织开发的Linux后门木马，很可能就是Linux版本的海莲花。

感谢社区提供的各种线索，让我们一起 [Make Cyber Security Great Again](#)。

## 引用

<https://unit42.paloaltonetworks.com/unit42-new-improved-macos-backdoor-oceanlotus/>



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 —

## Botnet



僵尸网络911 S5的数字遗产

Heads up! Xdr33, A Variant Of CIA's HIVE Attack Kit Emerges

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

**Botnet**

## RotaJakiro, the Linux version of the OceanLotus

On Apr 28, we published our RotaJakiro backdoor blog, at that time, we didn't have the answer for a very important question, what is this backdoor exactly for? We asked the community for clues and two days ago we got a hint, PE(Thanks!) wrote the following comment on

**sysrv**

**Threat Alert: New update from Sysrv-hello, now infecting victims' webpages to push malicious exe to end users**

Overview From the end of last year to now, we have seen the uptick of the mining botnet families. While new families have been popping up, some old ones are getting frequently updated. Our BotMon system has recently reported about the [rinfo][z0miner]. And the latest case comes from Sysrv-hello.

[See all 114 posts →](#)



• May 6, 2021 • 4 min read



Apr 29,

3 min



2021

read