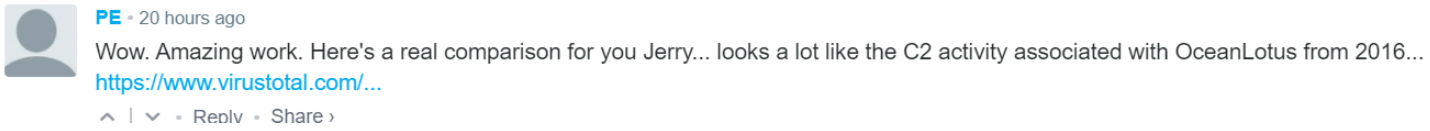
Botnet

RotaJakiro, the Linux version of the OceanLotus

Alex.Turing
May 6, 2021 • 4 min read

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 our blog post.



The sample mentioned in the message is a zip packing file, which has appeared in 2016. The zip contains multiple files, the Mach-O format executable file named `Noi dung chi tiet` (translated to `detailed information`) is the OceanLotus sample. When we compare the this file with the RotaJakiro sample, we noticed there are multiple similarities and it is `VERY likely` that this is **the Linux version of the OceanLotus**.

Similarity 1: Function for C2 session creation

The common domain name resolution function for Linux is `gethostbyname()`, but RotaJakiro uses the relatively niche `getaddrinfo()` function. C2 domain name resolution and session establishment are performed in one function, this is also used by the the OceanLotus sample. The comparison of the 2 functions is as follows.

RotaJakiro

```
*(_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);
```

OceanLotus

```
*(_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);
  }
```

It can be seen that they not only have the same function, but also use `sprintf()` and `getaddrinfo()` in almost exactly the same way. In addition, both RotaJakiro and OceanLotus use separate data structures to hold C2 session information, such as `socket fd, whether active, timeout`, etc., and their data structures are also very similar.

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

RotaJakiro

VS

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

OceanLotus

Similarity 2: registration packet construction method

The network packets of both RotaJakiro and OceanLotus are composed of `Head, Key, and Payload`, of which Head is mandatory and has a length of 82 bytes, while Key and Payload are optional.

- Offset 1, type DWORD, which holds a magic.
- Offset 9, type DWORD, holding the length of the Payload.
- Offset 13, type WORD, holding the Key length.
- Offset 15, type DWORD, holds the message code.

The RotaJakiro initializes the Head of the registration packets with a separate function.

```
{
  char *register_buf; // rbx
  unsigned int v1; // eax
  char *result; // rax

  register_buf = (char *)malloc(0x52uLL);
  v1 = time(0LL);
  srand(v1);
  *register_buf = rand();
  *(_DWORD *)(register_buf + 1) = 0x3B91011;
  *(_DWORD *)(register_buf + 5) = 0x4FB0CB1;
  *(_WORD *)(register_buf + 13) = 0;
  *(_DWORD *)(register_buf + 9) = 0;
  register_buf[19] = 0xC2u;
  *((_DWORD *)register_buf + 5) = 0x1206420;
  register_buf[24] = 0xE2u;
  *(_DWORD *)(register_buf + 25) = 0;
  register_buf[29] = 0xC2u;
  *(_DWORD *)(register_buf + 30) = 0;
  bzero(register_buf + 34, 0x20uLL);
  result = register_buf;
  register_buf[66] = -56;
  *(_WORD *)(register_buf + 75) = 255;
  register_buf[77] = 9;
  return result;
}
```

This function first calls the malloc() function to dynamically allocate memory for the registrationpacket, then calls the time()/srand()/rand() function in turn to generate a random character and then assign it to the first field of the registration packet, and the remaining large swath of code is to assign values to the remaining fields one by one with multiple constants, so the most obvious feature of this function is to `initialize the registration packet with multiple constants`.
There is also a function in the OceanLotus sample that is dedicated to initializing the Head of the registration packets.

This function has no code for memory allocation and random character generation, and the whole function uses multiple constants to assign values to

specific fields of the registration packet one by one, `exactly like the RotaJakiro`. In addition, OceanLotus shares the same field values with RotaJakiro at offsets 1, 24 and 75, especially the magic at offset 1 is `0x3B91011`, which is hard to describe as a coincidence, so it greatly increases the probability that these two pieces of code are the same origin. In addition, both the RotaJakiro and the OceanLotus have assigned message codes to the registration packets, and both are `0x2170272`:

```
if ( (unsigned int)sub_4046E0((__int64)v5, 0x2170272, 0x3B91011) )
```
RotaJakiro

VS

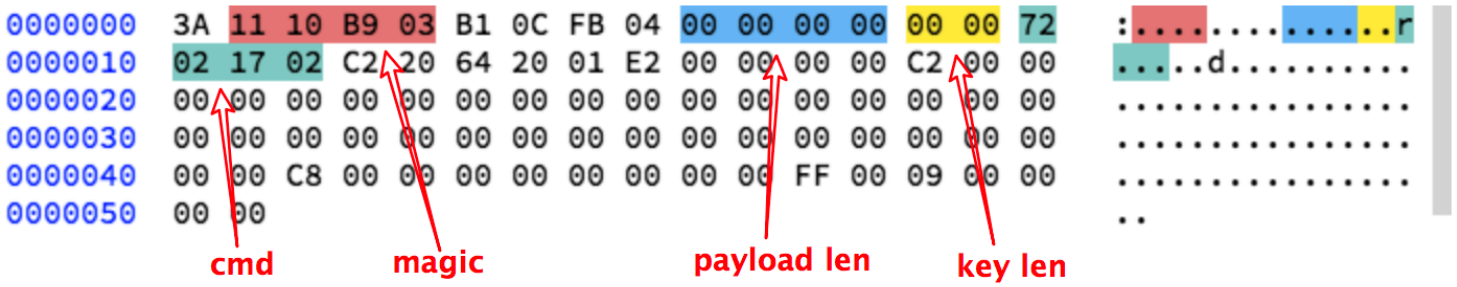
```
if ( sub_10000316A(*((_QWORD *)v1 + 2), 0x2170272LL, 0x3B91011) )
```
OceanLotus

The resulting registration packets is also very similar, and the RotaJakiro registration packets is as follows.

```
00000000  24 41 61 54 03 55 e2 1c  e3 63 63 63 63 63 63 2d  $AaT.U.. .ccccc-
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 63 9c 63 42 63 63  cczccccc ccc.cBcc
00000050  63 63                                             cc
```

The following is the OceanLotus registration packets analyzed by PAN in 2017.

The decrypted registration packets for the RotaJakiro is shown below.



The following is the OceanLotus plaintext registration packets from PAN's analysis.

You can see that they have the same plaintext structure and basically the same key field values.

Similarity 3: rotate function

Both RotaJakiro and OceanLotus have a function we called `rotate()` for encryption/decryption, the rotate function of RotaJakiro is as follows.

For OceanLotus

It is easy to see the commonalities between them.

1. Both accept 3 parameters.
2. The prototype is the same, where the first parameter is the actual rotate object, the second parameter is the length field, and the third parameter plays a control role.

In actual use, for example, in the process of encrypting the registration packets, you can see that the RotaJakiro and the OceanLotus `use the same parameters`.

```
*(++v8 - 1) = rotete((char)(v10 ^ 0x1B), 3, 1);          RotaJakiro
```

VS

```
result = (void *)rotate(*((char *)*a2 + v7) ^ 0x1Bu, 3, 1); OceanLotus
```

Similarity 4: Same instruction code

Both RotaJakiro and OceanLotus use DWORD type instruction codes to specify the function of the message, and share several semantically identical instruction codes, some of which are featured as shown in the following table.

CMD
0x18320e0
0x2170272
0x1B25503
0x1532e65
0x25D5082
This similarity obviously cannot be explained by coincidence, **it is an extremely strong evidence o**

Summary

Although the RotaJakiro and the Mac version of the OceanLotus are implemented in different languages, their similarity in function and message format design, and their similarity in specific implementation, can no longer be explained by coincidence. **It is highly likely that RotaJakiro is a Linux version of the OceanLotus.**

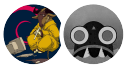
僵尸网络911 S5的数字遗产

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

See all 114 posts →

Background In Feb 2021, we came across an ELF sample using some CWP's Ndays exploits, we did some analysis, but after checking with a partner who has some nice visibility in network traffic in some China areas, we discovered there is literarily 0 hit for the C2 traffic. So



May 27, 2021 · 13 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： 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...

· May 6, 2021 · 6 min read