DGA家族Orchard持续变化，新版本用比特币交易信息生成DGA域名

blog.netlab.360.com/orchard-dga/

daji August 5, 2022

5 August 2022 / Botnet

DGA是一种经典的botnet对抗检测的技术，其原理是使用某种DGA算法，结合特定的种子和当前日期，定期生成大量的域名，而攻击者只是选择性的注册其中的极少数。对于防御者而言，因为难以事先确定哪些域名会被生成和注册，因而防御难度极大。

360 netlab长期专注于botnet攻防技术的研究，维护了专门的DGA算法和情报库，并通过订阅情报的方式与业界分享研究成果。近期我们在分析未知DGA域名时发现一例不但使用日期，还会同时使用中本聪的比特币账号交易信息来生成DGA域名的例子。因为比特币交易的不确定性，该技术比使用时间生成的DGA更难预测，因而防御难度更大。

该技术发现于一个名为Orchard的botnet家族。自从2021年2月份首次检测到该家族以来，我们发现它至少经历了3个版本的变化，中间甚至切换过编程语言。结合长期的跟踪结果和其它维度的信息，我们认为Orchard会是一个长期活跃、持续发展的botnet家族，值得警惕。本文将介绍Orchard的最新DGA技术，以及它这3个版本的发展过程。本文要点如下：

- Orchard是一个使用了DGA技术的botnet家族，核心功能是在受害者机器上安装各种恶意软件。
- 从2021年2月至今，我们先后检测到3个版本的Orchard样本，均使用了DGA技术。
- Orchard的DGA算法一直未变，但日期的使用方式一直在变，最新版同时支持使用比特币账号信息来生成单独的DGA域名。
- 除了DGA，Orchard还硬编码了C2域名。
- Orchard目前仍在活跃，致力于门罗币挖矿。

传播方式、规模以及影响范围

Orchard采用了“硬编码域名+DGA”的冗余C2机制，并且每个版本都硬编码了1个唯一的DuckDNS动态域名作为C2，根据它们的DGA实现方式和硬编码的域名，我们把已经检测到的Orchard样本分为3个版本：

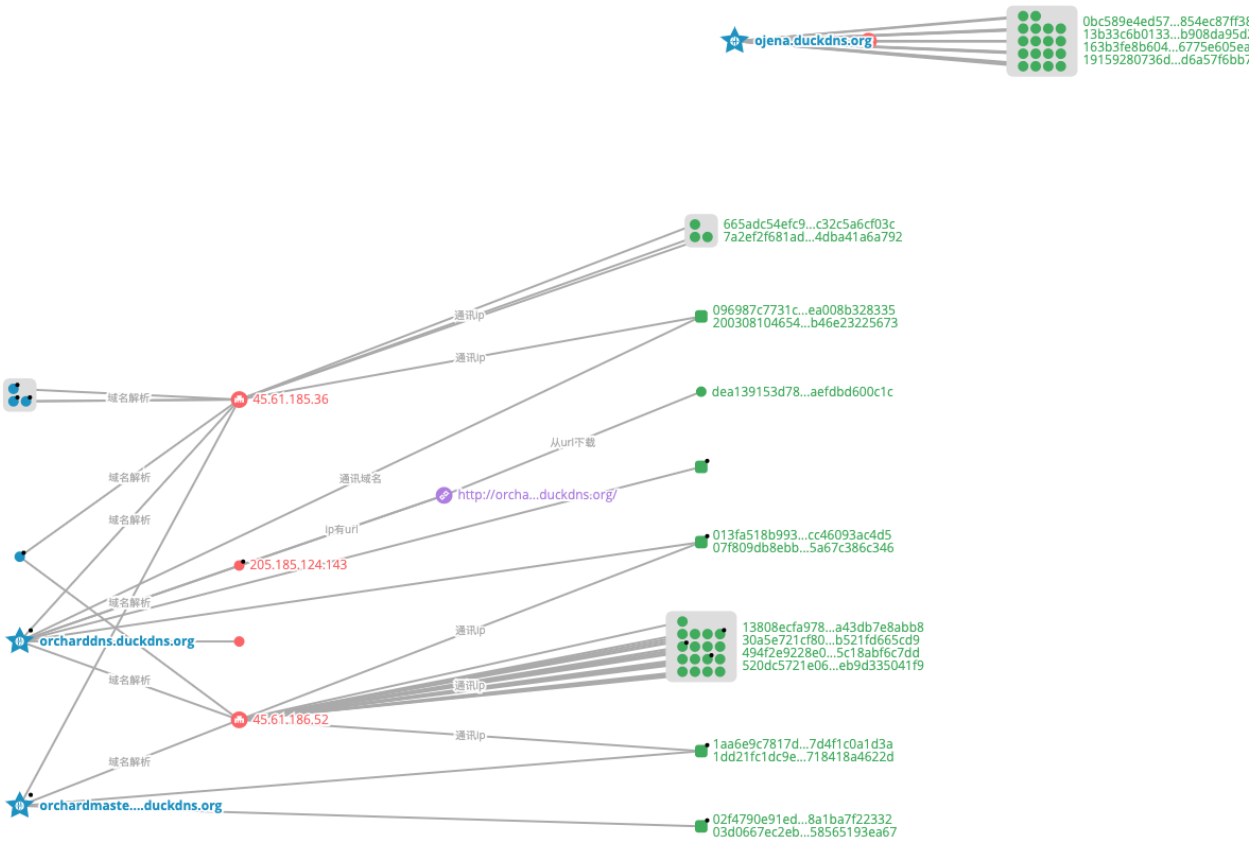
```
v1, orcharddns.duckdns.org
v2，orchardmaster.duckdns.org
v3, ojena.duckdns.org
```

它们的时间线如下：

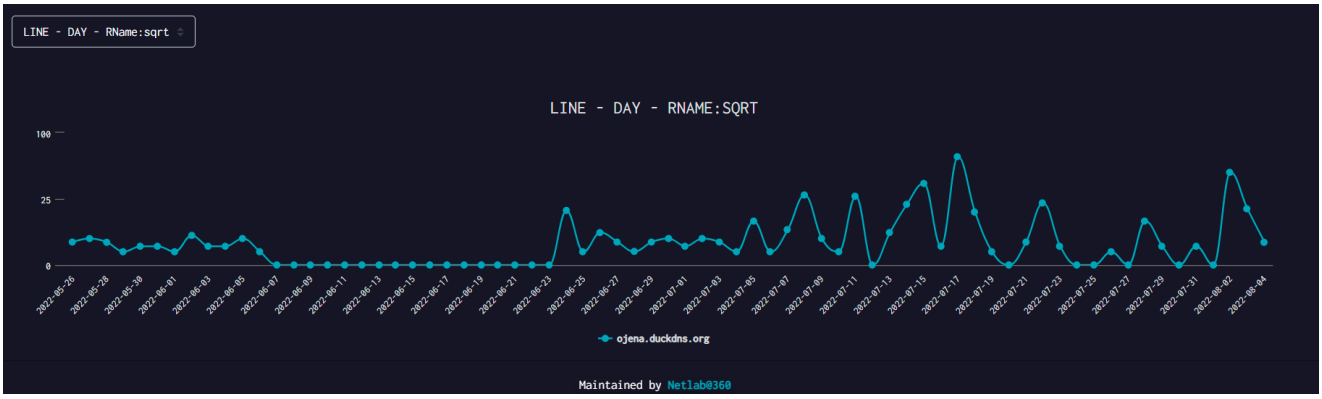
```
* 2021年3月，检测到v1版本，使用C++开发。结合历史数据，我们将v1首次出现时间提前到2021年2月。
* 2021年9月，检测到v2版本，它使用Golang和C++编写。
* 2022年7月，检测到v3版本，编写语言回到C++。
```

这3个版本都支持通过感染USB盘的方式进行传播，这一点跟传统的病毒很像，具体实现参考后面的“USB感染逻辑”部分。理论上，Orchard也完全可以通过其它方式传播。

利用我们的图系统结合PDNS和其它维度的数据，我们发现v1和v2的C2存在明显的共享IP的情况，如下图所示。



图系统帮我们找到了更多的C2 IP和域名，详见后面的IoC部分，这里的域名特点是都以duckdns.org结尾。v3因为比较新，没发现其它的关联域名，下面是v3域名的活跃情况。



能看到它是今年5月上线，然后逐渐活跃，目前应该仍然在活跃期内。

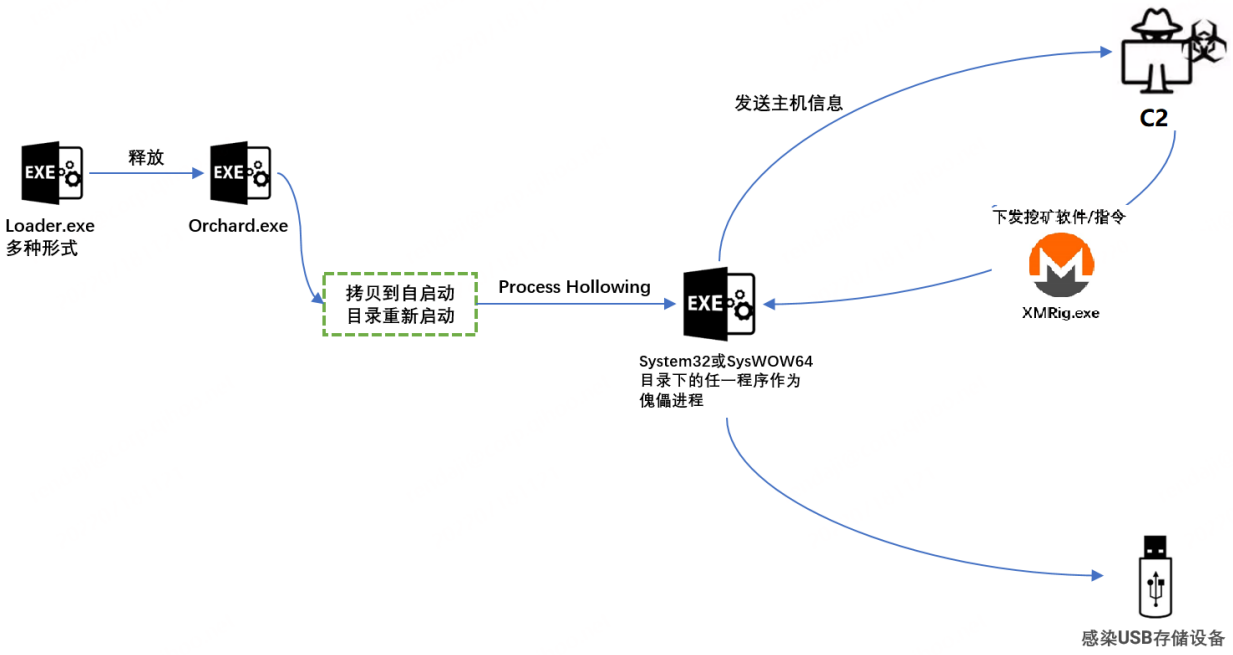
基于PDNS我们对3个版本的感染规模做了评估，其中v1和v2节点数近千，v3因为出现较晚，节点数不到500，下面是各个版本域名到具体IP的详细解析数。

```
# v1, orcharddns.duckdns.org
37, 45.61.185.36
413,  45.61.186.52
1301,  45.61.187.240
207, 205.185.124.143

# v2, orchardmaster.duckdns.org
45,  45.61.185.36
104, 45.61.186.52
659,  45.61.187.240

# v3, ojena.duckdns.org
418, 45.61.185.231
```

需要强调的是上面的规模数据只是我们视野内看到的，实际的应该要比这更多。

样本分析

Orchard样本在样本层面多使用loader，用于对抗分析和自我保护。目前看到的Orchard loader并不固定，即使单个版本也会出现多种loader的情况，比如v1版本的Orchard以base64字符串的形式存在于loader中，v2/v3版本的样本有的以资源文件的形式存放在loader中。各个版本还都曾使用过例如VMP、Enigma等虚拟壳来保护自身。总的来说，Orchard的工作流程可以用下面的图来总结。



Orchard三个版本的功能基本相同，包括：

- 上传设备及用户信息
- 响应指令/下载执行下一阶段的模块
- 感染USB存储设备

下面从DGA算法、C2通信和主机行为等几个维度分别分析3个版本Orchard的核心功能。

v1版本

该版本的分析以MD5=5c883ff8539b8d04be017a51a84e3af8的样本为基础。它在运行时首先释放内嵌的PE文件到自启动目录下，所释放的PE在内存中进行base64解码得到orchard的数据，随后该PE将System32/SysWOW64下的任一exe作为傀儡进程，来运行保存的orchard代码。该版本Orchard整体逻辑如下图，主要分为网络通信和USB感染两部分，最终功能取决于C2下发的具体模块，因此orchard本身可以认为是一个Downloader的角色。

```
void __cdecl mw_CoreWork_40C760()
{
  int v0; // [esp+20h] [ebp+8h]
  int v1; // [esp+24h] [ebp+Ch]
  int v2; // [esp+28h] [ebp+10h]
  void *v3; // [esp+2Ch] [ebp+14h] BYREF
  int v4; // [esp+40h] [ebp+28h]
  int v5; // [esp+48h] [ebp+30h]
  int v6; // [esp+4Ch] [ebp+34h]
  int v7; // [esp+50h] [ebp+38h]
  int v8; // [esp+54h] [ebp+3Ch]
  int v9; // [esp+58h] [ebp+40h]

  dword_5D6288 = v0;
  dword_5D628C = v1;
  dword_5D6290 = v2;
  memcpya32_4105C0(0xFFFFFFFF, lpParentProcImagePath, &v3, 0);
  dword_5D62B0 = v5;
  dword_5D62B4 = v6;
  dword_5D62BC = v8;
  dword_5D62B8 = v7;
  dword_5D62C0 = v9;
  strcpy_411F80(duckdns_DDNS_5D6218, "orcharddns.duckdns.org", 0x16u);// main c2
  strcpy_411F80(port5890_5D6234, "5890", 4u);
  strcpy_411F80(OrchardString_5D6250, "[o.r.c.h.a.r.d]", 0xFu);// 后续将以[o.r.c.h.a.r.d]作为分隔符拼接所收集信息
  memcpya12_412510(L"0.1|+", String0101_5D626C, 5u);
  diff_response_order_4035A0();                 // 定义指令对应关系
  if ( dword_5D62A4 )
    infect_usb_409980();                        // 感染移动存储设备
  dword_5D7430 = CreateThread(0, 0, thread_net_work_40bfa0, 0, 0, 0);// 核心线程
  if ( !dword_5D7430 )
    thread_net_work_40bfa0(0);
  if ( v4 >= 8 )
    free_416AC5(v3);
}
```

此处主要描述其网络通信过程（三个版本的USB感染逻辑相同，详见USB感染一节）。

C2通信过程较为简单，bot在check-in过程中向C2发送收集到的主机信息，然后等待C2响应的指令。v1版本所收集信息包括：卷序列号（HWID）、电脑名称、用户名、操作系统名称、系统版本、已安装的捕获驱动程序名称、杀软信息、父进程文件修改时间、置顶窗口名称及窗口标题等，这些信息以“[o.r.c.h.a.r.d]”作为分隔符进行拼接后发送，如下图所示。

```
    00000000  01 00 00 00 2e                                     .....
00000000  04 01 00 00                                          ....
00000004  5b 2a 5d 5b 6f 2e 72 2e  63 2e 68 2e 61 2e 72 2e   [*][o.r. c.h.a.r.
00000014  64 5d 42 43 43 39 37 37  45 46 5c 48 4f 4d 45 2d   d]BCC977 EF\HOME-
00000024  50 43 5c 61 64 6d 69 6e  5b 6f 2e 72 2e 63 2e 68   PC\admin [o.r.c.h
00000034  2e 61 2e 72 2e 64 5d 4d  69 63 72 6f 73 6f 66 74   .a.r.d]M icrosoft
00000044  20 57 69 6e 64 6f 77 73  20 38 2e 31 20 e4 bc 81    Windows  8.1 ...
00000054  e4 b8 9a e7 89 88 5b 6f  2e 72 2e 63 2e 68 2e 61   ......[o .r.c.h.a
00000064  2e 72 2e 64 5d 78 36 34  5b 6f 2e 72 2e 63 2e 68   .r.d]x64 [o.r.c.h
00000074  2e 61 2e 72 2e 64 5d 2d  5b 6f 2e 72 2e 63 2e 68   .a.r.d]- [o.r.c.h
00000084  2e 61 2e 72 2e 64 5d 57  69 6e 64 6f 77 73 20 44   .a.r.d]W indows D
00000094  65 66 65 6e 64 65 72 2e  20 5b 6f 2e 72 2e 63 2e   efender.  [o.r.c.
000000A4  68 2e 61 2e 72 2e 64 5d  2d 5b 6f 2e 72 2e 63 2e   h.a.r.d] -[o.r.c.
000000B4  68 2e 61 2e 72 2e 64 5d  30 2e 31 7c 2b 5b 6f 2e   h.a.r.d] 0.1|+[o.
000000C4  72 2e 63 2e 68 2e 61 2e  72 2e 64 5d 33 31 2d 30   r.c.h.a. r.d]31-0
000000D4  33 2d 32 30 32 32 5b 6f  2e 72 2e 63 2e 68 2e 61   3-2022[o .r.c.h.a
000000E4  2e 72 2e 64 5d 2d 5b 6f  2e 72 2e 63 2e 68 2e 61   .r.d]-[o .r.c.h.a
000000F4  2e 72 2e 64 5d 50 72 6f  67 72 61 6d 20 4d 61 6e   .r.d]Pro gram Man
00000104  61 67 65 72                                       ager
    00000005  01 00 00 00 2e                                     .....
00000108  02 00 00 00                                          ....
0000010C  2e 2e                                                ..
```

C2响应数据格式一般为“指令+数据”，指令的功能通过指令码指定。下面是一个具体的C2响应，其中"[&&]"代表指令码2，代表下载执行，具体处理过程分为2种：响应数据如果是URL，则下载URL对应的PE并执行；如果是base64编码的内容，则先解码然后执行解码后的数据。此处响应的数据实际是base64编码的新版本PE文件，相当于升级，这也表明老版本可能已经废弃。

```
000000E4  72 2e 64 5d 50 72 6f 67  72 61 6d 20 4d 61 6e 61   r.d]Prog ram Mana
000000F4  67 65 72                                          ger
    00000000  01 00 00 00 2e                                    .....
    00000005  60 88 0d 00 5b 26 26 5d  5b 6f 2e 72 2e 63 2e 68   `...[&&] [o.r.c.h
    00000015  2e 61 2e 72 2e 64 5d 54  56 71 51 41 41 4d 41 41   .a.r.d]T VqQAAMAA
    00000025  41 41 45 41 41 41 41 2f  2f 38 41 41 4c 67 41 41   AAEAAAA/ /8AALgAA
    00000035  41 41 41 41 41 41 41 51  41 41 41 41 41 41 41 41   AAAAAAAQ AAAAAAAA
    00000045  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
    00000055  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
    00000065  41 41 41 41 41 41 41 43  41 45 41 41 41 34 66 75   AAAAAAAC AEAAA4fu
    00000075  67 34 41 74 41 6e 4e 49  62 67 42 54 4d 30 68 56   g4AtAnNI bgBTM0hV
    00000085  47 68 70 63 79 42 77 63  6d 39 6e 63 6d 46 74 49   GhpcyBwc m9ncmFtI
    00000095  47 4e 68 62 6d 35 76 64  43 42 69 5a 53 42 79 64   GNhbm5vd CBiZSByd
    000000A5  57 34 67 61 57 34 67 52  45 39 54 49 47 31 76 5a   W4gaW4gR E9TIG1vZ
    000000B5  47 55 75 44 51 30 4b 4a  41 41 41 41 41 41 41 41   GUuDQ0KJ AAAAAAAA
    000000C5  41 44 4d 4c 32 63 47 69  45 34 4a 56 59 68 4f 43   ADML2cGi E4JVYhOC
    000000D5  56 57 49 54 67 6c 56 6e  43 55 4b 56 49 4a 4f 43   VWITglVn CUKVIJOC
    000000E5  56 57 63 4a 51 78 55 45  30 34 4a 56 64 6f 37 44   VWcJQxUE 04JVdo7D
    000000F5  56 53 62 54 67 6c 56 32  6a 73 4b 56 4a 70 4f 43   VSbTglV2 jsKVJpOC
    00000105  56 58 61 4f 77 78 55 6f  30 34 4a 56 5a 77 6c 44   VXaOwxUo 04JVZwlD
    00000115  56 53 62 54 67 6c 56 6e  43 55 50 56 49 6c 4f 43   VSbTglVn CUPVIlOC
    00000125  56 57 63 4a 51 68 55 67  30 34 4a 56 59 68 4f 43   VWcJQhUg 04JVYhOC
    00000135  46 58 36 54 67 6c 56 50  54 73 41 56 49 74 4f 43   FX6TglVP TsAVItOC
    00000145  56 55 39 4f 2f 5a 56 69  55 34 4a 56 59 68 4f 6e   VU9O/ZVi U4JVYhOn
    00000155  6c 57 4a 54 67 6c 56 50  54 73 4c 56 49 6c 4f 43   lWJTglVP TsLVIlOC
    00000165  56 56 53 61 57 4e 6f 69  45 34 4a 56 51 41 41 41   VVSaWNoi E4JVQAAA
    00000175  41 41 41 41 41 41 41 55  45 55 41 41 45 77 42 42   AAAAAAAU EUAAEwBB
    00000185  51 42 6a 67 4b 39 69 41  41 41 41 41 41 41 41 41   QBjgK9iA AAAAAAAA
```

v1版本一共定义了8个指令，指令码与指令字符串的对应关系如下：

```
1 \[=]
2 \[&&]
3 \[##]
4 \[###]
5 \[%%]
6 \[%%%]
7 \[#\_#]
8 \[\_\_\] \[>>] \[<<] \[^^] \[\*\] \[\~\] \[@] \[!] \[#\*\#\] \[#@#]
```

由于某些指令置空，8个指令实际对应三种操作（后续版本大同小异）：

- 指令码1和2：判断响应数据为URL或者PE，如果是URL则下载执行，如果是PE，则创建进程执行（CreateProcess创建进程、傀儡进程、远程线程注入等）。
- 指令码3、4、8：终结当前进程删除原始文件，或者重新启动。
- 指令码7：再次收集C2、port、PID、文件名信息向C2进行发送，示例：orcharddns.duckdns.org[o.r.c.h.a.r.d]5890[o.r.c.h.a.r.d]2260[o.r.c.h.a.r.d]stage-3_.exe[o.r.c.h.a.r.d]

DGA算法

v1的DGA以日期字符串（比如“2022/07/05”）作为输入，计算其MD5值，然后将MD5字符串均分成长度为8的四个子字符串，依次与 .com、.net、.org、.duckdns.org 这4个后缀拼接，得到每天4组16个DGA域名，算法实现如下。

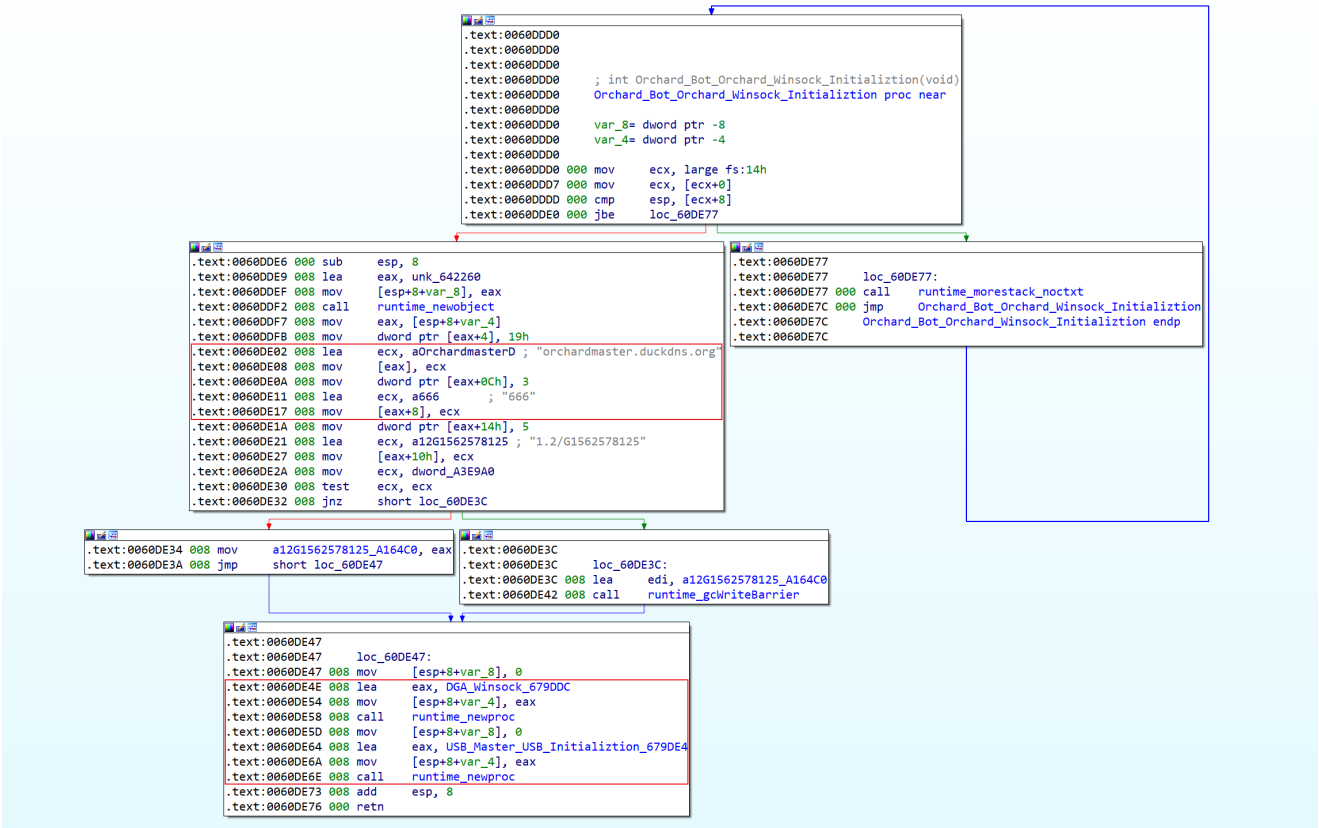
```
# 2021/04/15
import datetime
import hashlib

days=30
for i in range(0, days):
    datex = (datetime.datetime.now() -
datetime.timedelta(days=i)).strftime('%Y/%m/%d')
    print("seed: ", datex)
    md5 = hashlib.md5(datex.encode()).hexdigest()
    print('md5: ', md5)

    dga_list = []
    dga_list.append(md5[:8])
    dga_list.append(md5[8:16])
    dga_list.append(md5[16:24])
    dga_list.append(md5[24:32])
    for j in range(len(dga_list)):
        print(dga_list[j] + '.com')
        print(dga_list[j] + '.net')
        print(dga_list[j] + '.org')
        print(dga_list[j] + '.duckdns.org')
```

示例域名如下：

```
seed:  2022/07/05
md5:  91ac64d29f78281ad802f44648b2137f
91ac64d2.com
91ac64d2.net
91ac64d2.org
91ac64d2.duckdns.org
9f78281a.com
9f78281a.net
9f78281a.org
9f78281a.duckdns.org
d802f446.com
d802f446.net
d802f446.org
d802f446.duckdns.org
48b2137f.com
48b2137f.net
48b2137f.org
48b2137f.duckdns.org
```

v2版本

v2版本出现了两种编程语言实现的样本，分别是Golang和C++，但是功能相同。这里的分析以MD5=f3e0b960a48b433bc4bfe6ac44183b74的Golang样本为例，它的C2初始化函数如下图所示，能明显看到硬编码的C2域名。



v2版本开始使用json格式，字段含义相对清晰。其收集的信息跟v1大致相同，包括：卷序列号（HWID）、电脑名称、用户名、系统版本、杀软信息、活动窗口信息等，新增的字段有：.net框架版本（比如v2.0.50727）、USB 状态、发包类型及自身版本。下面是一个实际观察到的版本号信息，Bot_Version=1.2/G可能的解释为：版本=v1.2，编写语言=Golang。

```
00000000  d6 00 00 00                                       ....
00000004  7b 22 49 64 65 6e 74 69  74 79 22 3a 22 44 37 34  {"Identi ty":"D74
00000014  30 37 32 33 5c 5c 41 44  4d 49 4e 2d 50 43 5c 5c  0723\\AD MIN-PC\\
00000024  41 64 6d 69 6e 22 2c 22  41 72 63 68 69 74 65 63  Admin"," Architec
00000034  74 75 72 65 22 3a 30 2c  22 41 6e 74 69 76 69 72  ture":0, "Antivir
00000044  75 73 22 3a 5b 22 2d 22  5d 2c 22 44 6f 74 4e 65  us":["-" ],"DotNe
00000054  74 53 74 61 74 75 73 22  3a 74 72 75 65 2c 22 42  tStatus" :true,"B
00000064  6f 74 5f 56 65 72 73 69  6f 6e 22 3a 22 31 2e 32  ot_Versi on":"1.2
00000074  2f 47 22 2c 22 55 53 42  5f 53 74 61 74 75 73 22  /G","USB _Status"
00000084  3a 66 61 6c 73 65 2c 22  41 63 74 69 76 65 5f 57  :false," Active_W
00000094  69 6e 64 6f 77 22 3a 22  54 43 50 56 69 65 77 20  indow":" TCPView
000000A4  2d 20 53 79 73 69 6e 74  65 72 6e 61 6c 73 3a 20  - Sysint ernals:
000000B4  77 77 77 2e 73 79 73 69  6e 74 65 72 6e 61 6c 73  www.sysi nternals
000000C4  2e 63 6f 6d 22 2c 22 50  61 63 6b 65 74 5f 54 79  .com","P acket_Ty
000000D4  70 65 22 3a 30 7d                                 pe":0}
    00000000  02 00 00 00 2e 2e                                 ......
    00000006  02 00 00 00 2e 2e                                 ......
000000DA  02 00 00 00                                       ....
000000DE  2e 2e                                             ..
```

v2版本的C++语言样本集成了同样的C2，上线包中的版本信息则变成了“Bot_version:1/C”，它所收集的信息如下图所示。

```
00000000  c6 00 00 00                                       ....
00000004  7b 22 41 63 74 69 76 65  5f 57 69 6e 64 6f 77 22  {"Active _Window"
00000014  3a 22 50 72 6f 67 72 61  6d 20 4d 61 6e 61 67 65  :"Progra m Manage
00000024  72 22 2c 22 41 6e 74 69  76 69 72 75 73 22 3a 5b  r","Anti virus":[
00000034  22 57 69 6e 64 6f 77 73  20 44 65 66 65 6e 64 65  "Windows  Defende
00000044  72 22 5d 2c 22 41 72 63  68 69 74 65 63 74 75 72  r"],"Arc hitectur
00000054  65 22 3a 30 2c 22 42 6f  74 5f 56 65 72 73 69 6f  e":0,"Bo t_Versio
00000064  6e 22 3a 22 31 2f 43 22  2c 22 44 6f 74 4e 65 74  n":"1/C" ,"DotNet
00000074  53 74 61 74 75 73 22 3a  30 2c 22 49 64 65 6e 74  Status": 0,"Ident
00000084  69 74 79 22 3a 22 32 34  46 39 43 33 30 42 5c 5c  ity":"24 F9C30B\\
00000094  44 45 53 4b 54 4f 50 2d  50 32 38 38 39 30 4e 5c  DESKTOP- P28890N\
000000A4  5c 64 61 6a 69 22 2c 22  50 61 63 6b 65 74 5f 54  \daji"," Packet_T
000000B4  79 70 65 22 3a 30 2c 22  55 53 42 5f 53 74 61 74  ype":0," USB_Stat
000000C4  75 73 22 3a 30 7d                                 us":0}
    00000000  02 00 00 00 2e 2e                                 ......
    00000006  02 00 00 00 2e 2e                                 ......
```

根据代码相似性分析，v2版本的C++样本跟后来的v3版本代码同源，说明后者是从前者进化而来。

v2版本一共有两种指令：

- 指令1：终结当前进程删除原始文件，或者重新启动。
- 指令2：判断响应数据为URL或者PE，如果是URL则下载执行，如果是PE，则创建进程执行（CreateProcess创建进程、傀儡进程、远程线程注入等）。

DGA算法

v2版本的DGA算法跟v1相同，差别在于对日期字符串的处理，v2会在日期字符串后拼接硬编码的域名“orchardmaster.duckdns.org”，形如“2022/07/05orchardmaster.duckdns.org"，然后套用v1版本的DGA算法生成域名。

v3版本

v3的开发语言回到C++编写，同样包括C2通信和USB感染功能。C2通信逻辑在一个线程中运行，同时该线程还包括一个跟XMRig挖矿绑定的辅助线程，当Orchard接收完毕下发的XMrig程序并创建傀儡进程运行之后，辅助线程会向C2再次发送挖矿相关的硬件信息，尝试从C2读取挖矿软件的配置，目的是为了检查是否需要动态修改XMRig运行时的配置（XMRig提供了一套HTTP api，支持动态读取并修改运行时的挖矿配置）。

以MD5=cb442cbff066dfef2e3ff0c56610148f的样本为例，C2通信功能如下。

v3版本在C2通信中同样使用json格式来保存主机信息，发送数据的整体结构为**Byte_0x46+TotalLen+InfoLen+Info.json**。相比v2，v3增加了多个跟挖矿相关的字段，收集的信息包括：

- Active_Window：当前活动窗口名称
- Antivirus：杀软信息
- Authentiate_Type：Windows身份验证类型
- CPU_Model：CPU信息
- Camera：是否存在摄像头
- Elevated：是否是管理员权限
- GPU_Models：显卡信息
- Identity：HWID\用户名\电脑名称
- Operating_System：系统版本信息
- Ram_Size：运行内存大小
- System_Architecture：处理器个数
- Threads：每个处理器内核个数
- Version：Orchard版本

v3的上线包实例如下所示。

```
00000000  3f 03 00 00                                       ?...
00000004  47 97 01 00 00 af 01 00  00 00 00 00 80 7b 22 41  G....... .....{"A
00000014  63 74 69 76 65 5f 57 69  6e 64 6f 77 22 3a 22 45  ctive_Wi ndow":"E
00000024  76 65 72 79 74 68 69 6e  67 22 2c 22 04 08 00 a1  verythin g","....
00000034  41 6e 21 e1 69 72 75 73  22 3a 5b 22 14 fb 73 20  An!.irus ":["..s
00000044  44 65 66 65 6e 64 65 72  22 5d e1 63 75 74 68 65  Defender "].cuthe
00000054  91 2f 63 00 00 40 80 61  74 65 5f 54 79 70 65 22  ./c..@.a te_Type"
00000064  3a 30 2c 22 43 50 55 5f  4d 6f 64 65 6c 11 80 49  :0,"CPU_ Model..I
00000074  6e 74 65 6c 28 52 29 00  00 04 a0 20 43 6f 72 65  ntel(R). ... Core
00000084  28 54 4d 29 20 69 37 2d  38 37 30 30 20 61 51 20  (TM) i7- 8700 aQ
00000094  40 20 33 2e 32 30 47 48  7a 01 e0 43 00 00 d4 d8  @ 3.20GH z..C....
000000A4  61 6d 65 72 61 22 3a 66  61 6c 73 65 2c 22 45 6c  amera":f alse,"El
000000B4  65 76 61 23 64 17 c4 47  56 0a 12 1d 7b 22 4e 71  eva#d..G V...{"Nq
000000C4  b3 11 80 2d 01 e0 49 02  00 80 34 e5 32 7d f1 e7  ...-..I. ..4.2}..
000000D4  49 64 32 92 74 79 11 80  32 34 46 39 43 33 30 42  Id2.ty.. 24F9C30B
000000E4  5c 5c 64 61 6a 69 5c 5c  44 45 53 4b 54 00 a4 80  \\daji\\ DESKT...
000000F4  80 4f 50 2d 50 32 38 38  39 30 4e 01 e0 4f 70 21  .OP-P288 90N..Op!
00000104  47 74 f1 81 5f 53 79 73  74 65 6d 11 80 7b 5c 22  Gt.._Sys tem..{\"
00000114  42 75 69 6c 00 00 00 c0  64 4e 75 6d 62 65 72 5c  Buil.... dNumber\
00000124  22 3a 31 30 32 34 30 2c  5c 22 4d 61 6a 6f 72 56  ":10240, \"MajorV
00000134  65 72 73 69 6f 6e e3 1f  19 70 01 81 92 e0 69 6e  ersion.. .p....in
00000144  8a 70 22 9f 50 72 6f 64  75 63 74 32 e5 e1 1f 03  .p".Prod uct2....
00000154  7a 7d 01 e0 52 61 6d 5f  53 69 7a 71 1c 32 31 34  z}..Ram_ Sizq.214
00000164  36 34 31 40 00 08 a0 38  36 38 38 2c 22 44 e6 5f  641@...8 688,"D._
00000174  41 72 63 68 69 74 65 63  74 75 72 71 1c 31 2c 22  Architec turq.1,"
00000184  54 68 72 65 61 64 11 1d  34 04 00 00 80 2c 22 05  Thread.. 4....,".
00000194  27 22 3a 32 2e 30 7d 00  bc db 68 00 ee f1 ee f1  '":2.0}. ..h.....
000001A4  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
```

C2响应消息的body部分也为json格式，其结构为：TotalLen.dword+ Byte0x46+TotalLen+RespDataLen+RespData.json。v3支持8个指令，对应3种操作：

- 指令1：收集主机信息/自身运行状态并发送到C2（字段包括Domain、In_Memory、Install_Path、Is_Patched、Message_Type、Patch_Name、Port、Power_SaverMode、Process_ID、Process_Name、Process_Path、System_Idle、System_Uptime）
- 指令4、6：终结当前进程删除原始文件，或者重新启动。
- 指令7、8：下载&执行下发的矿机程序

下面是一个实际跟踪到的C2响应指令。

```
00000000  0f 00 00 00 47 0f 00 00  00 02 00 00 00 00 00 00  ....G... ........
00000010  80 2e 2e                                         ...
00000013  3b 00 00 00 46 3b 00 00  00 32 00 00 00 7b 22 54  ;...F;.. .2...{"T
00000023  79 70 65 22 3a 30 2c 22  54 72 61 6e 73 66 65 72  ype":0," Transfer
00000033  5f 50 6f 72 74 22 3a 22  32 39 32 39 22 2c 22 4d  _Port":" 2929","M
00000043  65 73 73 61 67 65 5f 54  79 70 65 22 3a 37 7d     essage_T ype":7}
```

其中Transfer_Port表示希望主机再次向2929进行请求，Message_Type表示指令码，其值为7，表示下载&执行。

收到上述指令后，bot再次向C2的TCP 2929端口发起请求，Cuda是Nvidia推出的只能用于自家GPU的并行计算框架，这里的Cuda_Version为0表示不支持Cuda。

```
00000000  b6 01 00 00                                        ....
00000004  45 31 26 00 00 00 80 7b  22 43 75 64 61 5f 56 65  E1&....{ "Cuda_Ve
00000014  72 73 69 6f 6e 22 3a 30  2c 22 52 65 71 75 69 72  rsion":0 ,"Requir
00000024  65 64 5f 42 69 6e 00 00  00 80 61 72 79 22 3a 30  ed_Bin.. ..ary":0
00000034  7d 77 73 20 44 65 66 65  6e 64 65 72 01 00 00 00  }ws Defe nder....
00000044  84 e8 67 00 94 e8 67 00  04 00 00 00 00 00 00 00  ..g...g. ........
00000054  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
00000064  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
```

随后C2响应一个XMRig矿机程序，Client接收保存后根据指令7将XMRig注入傀儡进程开始执行挖矿工作。

```
00000000  cf 68 29 00 47 cf 68 29  00 00 2a 48 00 00 82 0d  .h).G.h) ..*H....
00000010  80 4d 5a 90 00 03 00 00  00 04 01 00 ff ff 00 00  .MZ..... ........
00000020  b8 01 00 02 00 40 02 00  00 00 1f 30 01 00 00 0e  .....@.. ...0....
00000030  1f ba 0e 00 b4 09 00 00  00 80 cd 21 b8 01 4c cd  ........ ...!..L.
00000040  21 54 68 69 73 20 70 72  6f 67 72 61 6d 20 63 61  !This pr ogram ca
00000050  6e 6e 6f 74 20 62 65 20  72 00 00 08 90 75 6e 20  nnot be  r....un
00000060  69 6e 20 44 4f 53 20 6d  6f 64 65 2e 0d 0d 0a 24  in DOS m ode....$
00000070  05 00 1c 31 47 6d 58 50  29 3e d6 2c 4c 3b 08 82  ...1GmXP )>.,L;..
00000080  20 a2 2a 3f 54 21 ab 4c  3b 2c 3f 93 21 ab 3e 3f   .*?T!.L ;,?.!.>?
00000090  d4 3e 5c 21 ab 0a 25 2d  3f 4b 23 ab 2a 3f 52 23  .>\!..%- ?K#.*?R#
000000A0  ab 2c 3f 05 21 ab 4c 50  41 00 80 3b 2d 3f 42 21  .,?.!.LP A..;-?B!
000000B0  ab ed 71 ed 4a 21 ab 4c  3b 28 3f 4f 21 ab 58 50  ..q.J!.L ;(?O!.XP
000000C0  28 3e 2f 51 29 3e dd 20  2d 3f 0b 52 29 3e 20 04  (>/Q)>.  -?.R)> .
000000D0  85 f0 ed 25 20 3f 58 31  ab ed 25 2a 3f 92 2c ed  ...% ?X1 ..%*?.,.
000000E0  25 d6 3e 59 23 ab be b3  c3 ed 25 2b 3f c2 2c 52  %.>Y#... ..%+?.,R
000000F0  69 63 68 d2 2c 05 00 0f  00 00 10 00 88 50 45 00  ich.,... .....PE.
00000100  00 64 86 0b 00 f3 89 af  62 06 00 f0 00 22 00 0b  .d...... b....".
00000110  02 0e 1d 00 ac 31 00 00  7e 04 04 ec 4f 2e c8 70  .....1.. ~...O..p
00000120  08 d0 00 00 10 03 00 40  01 02 00 02 01 02 00 00  .......@ ........
00000130  06 02 00 01 00 66 00 00  80 72 00 02 40 44 10 49  .....f.. .r..@D.I
00000140  00 02 00 60 81 06 10 00  02 01 0f 12 18 80 01 00  ...`.... ........
00000150  00 10 12 07 10 03 00 54  6b 44 00 f0 02 00 f0 71  .......T kD.....q
00000160  14 00 90 6f 00 94 e5 01  02 00 05 00 72 00 88 7f  ...o.... ....r...
00000170  00 00 80 6d 41 00 86 aa  41 fe 1c 05 00 0b 00 6f  ...mA... A......o
00000180  41 00 28 01 00 a0 91 16  38 1a 00 c0 11 03 0a 01  A.(..... 8.......
00000190  00 00 00 17 2e 74 65 78  74 01 00 f8 aa 11 03 02  .....tex t.......
000001A0  01 f2 1a 42 00 01 00 06  00 00 00 ea a0 20 00 00  ...B.... ..... ..
000001B0  60 2e 72 64 61 74 61 00  00 6c cd 12 00 00 32 1c  `.rdata. .l....2.
000001C0  ce 21 01 b0 11 03 06 00  01 00 40 00 00 40 2e 24  .!...... ..@..@.$
000001D0  62 00 00 f4 38 bd 0c fb  2a 00 00 90 44 00 00 f8  b...8... *...D...
```

分析过程中我们发现v3版本最近在持续分发一个同样的XMRig挖矿程序，后者集成了默认的挖矿配置信息，私有矿池地址：45.61.187.7:7733

DGA算法

v3的DGA算法未变，但输入的变化较大。实际上它会生成两组DGA域名，第一组域名的输入拼接算法是日期字符串+“ojena.duckdns.org”， 形如 “2022-08-02ojena.duckdns.org”。第二组域名的输入为 `https://blockchain.info/balance?active=1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa` 这个URL的返回结果，一个典型的返回结果如下所示：

```
{"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa":"final_balance":6854884253,"n_tx":3393,"total_re
```

相关字段的含义可以参考Blockchain的API手册：

```
n_tx：交易数量
total_received：接收比特币总量
final_balance：最终余额
```

值得强调的是v3版本并未对返回的结果进行解析，而是作为整体直接输入DGA算法来生成域名。而钱包地址 `1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa` 据说是中本聪本人所持有的比特币创世地址。过去的十几年间，由于各种原因，每天都会有人向该钱包转入小量比特币，因此它是变化的，并且该变化很难预测，因此该钱包的余额信息也可以作为DGA输入。

在我们编写文章时，发现近期已有其他研究人员注意到v3版本这种将比特币账号交易信息用作DGA输入的现象，所分析结果与我们一致，但对方并没有注意到Orchard其实早已出现。

完整的v3版本DGA算法如下：

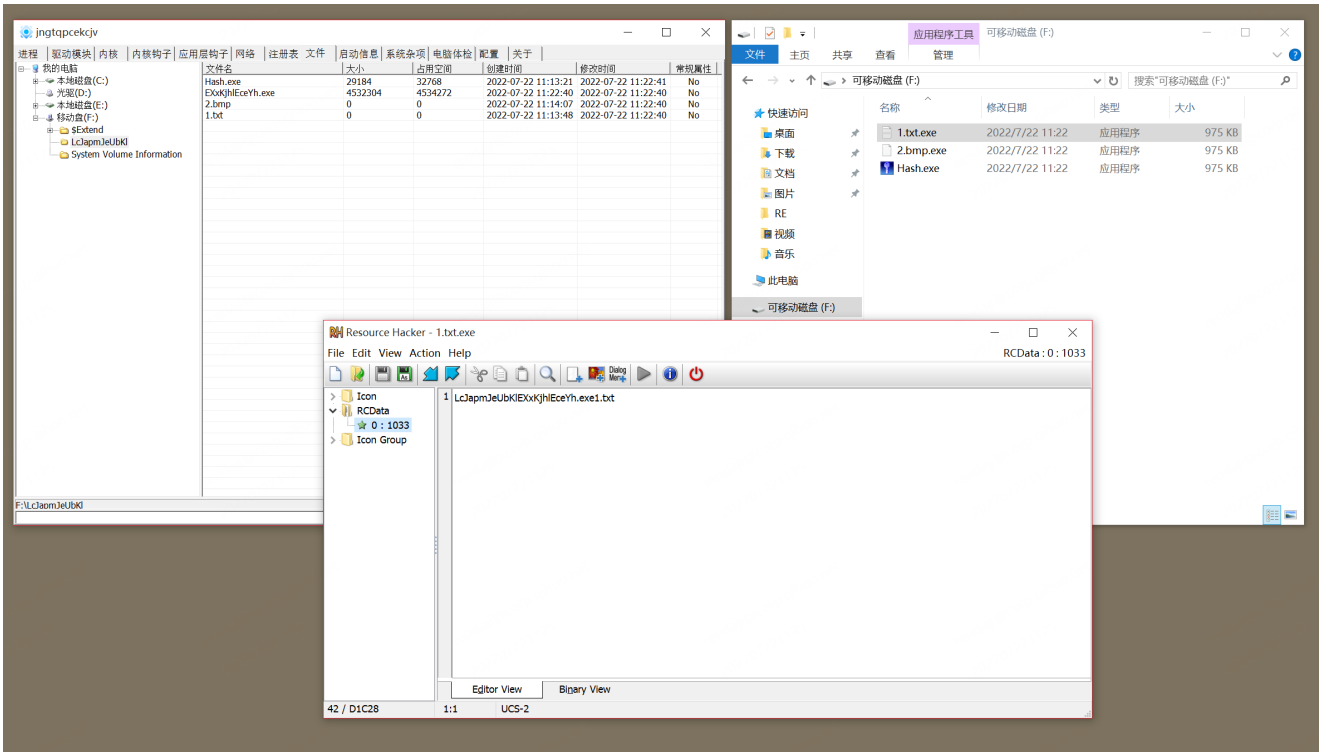
```
# 2022/07/05
import datetime
import requests
import hashlib

# cluster 1
days = 30
for i in range(0, days):
    domains = ['ojena.duckdns.org', 'vgzero.duckdns.org']
    for do in domains:
        datex = (datetime.datetime.now() - datetime.timedelta(days=i)).strftime('%Y-
%m-%d' + do)
        print("seed_1: %s" % datex)
        md5 = hashlib.md5(datex.encode()).hexdigest()
        print("md5: %s" % md5)

        dga_list = []
        dga_list.append(md5[:8])
        dga_list.append(md5[8:16])
        dga_list.append(md5[16:24])
        dga_list.append(md5[24:32])
        for j in range(len(dga_list)):
            print(dga_list[j] + '.com')
            print(dga_list[j] + '.net')
            print(dga_list[j] + '.org')
            print(dga_list[j] + '.duckdns.org')


# cluster 2
url = 'https://blockchain.info/balance?active=1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa'
res = requests.get(url)
wallet_info = res.text
print('seed_2: %s' % wallet_info)
md5 = hashlib.md5(wallet_info.encode()).hexdigest()
print('md5: %s' % md5)

dga_list = []
dga_list.append(md5[:8])
dga_list.append(md5[8:16])
dga_list.append(md5[16:24])
dga_list.append(md5[24:32])
for j in range(len(dga_list)):
    print(dga_list[j] + '.com')
    print(dga_list[j] + '.net')
    print(dga_list[j] + '.org')
    print(dga_list[j] + '.duckdns.org')
```

USB感染逻辑

Orchard的文件感染并非传统的代码插入，而是一种文件替换。当检测到USB存储设备时，Orchard会在设备根目录下创建隐藏目录，遍历所有文件进行感染，并将感染前和感染后的文件都备份到该隐藏目录下，被感染对象在感染时去掉了类型属性，感染后全部变为exe类型，并追加了.exe后缀，变成了可执行文件。随后样本会复制自身到被感染目录下并随机命名，该字符串保存到了被感染文件的资源里。当设备中的被感染文件在新系统中被用户执行后，则会启动隐藏目录中的样本文件，达到感染传播的目的。

USB感染过程会涉及两个内嵌的PE文件，第一个文件是DLL文件，会被释放到%LocalAppData%目录下，该DLL被Orchard称作CGO_Helper，主要用于提取和替换被感染文件的图标，其MD5是10D42F5465D5D8808B43619D8266BD99。第二个文件是exe文件，MD5为f3c06399c68c5fdf80bb2853f8f2934b，作为存储感染代码的模板文件，被感染文件的全部数据将被替换为该模板文件的数据。该模板的功能是根据资源中的exe名称寻找隐藏目录下对应的exe启动执行，所以被感染文件的资源中保存的是备份的Orchard样本名称。

USB感染情况示例如下，被感染文件资源中保存了Orchard样本的名称，当用户点击受感染的exe，将启动隐藏目录下的Orchard样本文件：



总结

Orchard是一个使用了DGA技术的botnet家族，最新版本致力于挖矿，并开始使用中本聪的比特币账号交易信息这类更难预测的信息作为DGA的输入，增加了检测难度。在1年多的时间里，Orchard先后出现了至少3个不同版本，编程语言和DGA实现都有变化，这说明Orchard是一个仍处于活跃期的botnet家族，预计后续会有更多的变种出现，值得我们警惕。对Orchard我们会持续保持关注，有新的发现会继续公开。

联系我们

感兴趣的读者，可以在 **twitter** 或者通过邮件**netlab[at]360.cn**联系我们。

IOCs

C2

```
orcharddns.duckdns.org
orchardmaster.duckdns.org
ojena.duckdns.org
vgzero.duckdns.org
victorynicholas.duckdns.org
zamarin1.duckdns.org

45.61.185.36
45.61.186.52
45.61.187.240
205.185.124.143
45.61.185.231
```

MD5

```
5c883ff8539b8d04be017a51a84e3af8
f3e0b960a48b433bc4bfe6ac44183b74
9cbe4bd27eba8c70b6eddaeb6707659b
cb442cbff066dfef2e3ff0c56610148f
10D42F5465D5D8808B43619D8266BD99
f3c06399c68c5fdf80bb2853f8f2934b
19159280736dbe6c11b7d6a57f6bb7b9
b5a6f78d5575a60316f4e784371d4f8c
3c20ba851edecd28c198691321429883
2b244a39571ab27f7bb4174d460adeef
ae1e9b3621ee041be6ab5e12bff37c53
00b1620f89b7980b34d53737d9e42fd3
4d2445a43591d041cabbbf3dfca6dfbd
```

私有矿池

```
45.61.187.7:7733
```