Botnet

# DGA家族Orchard持续变化，新版本用比特币交易信息生成DGA域名

daji, suqitian

Aug 5, 2022 • 18 min read

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：

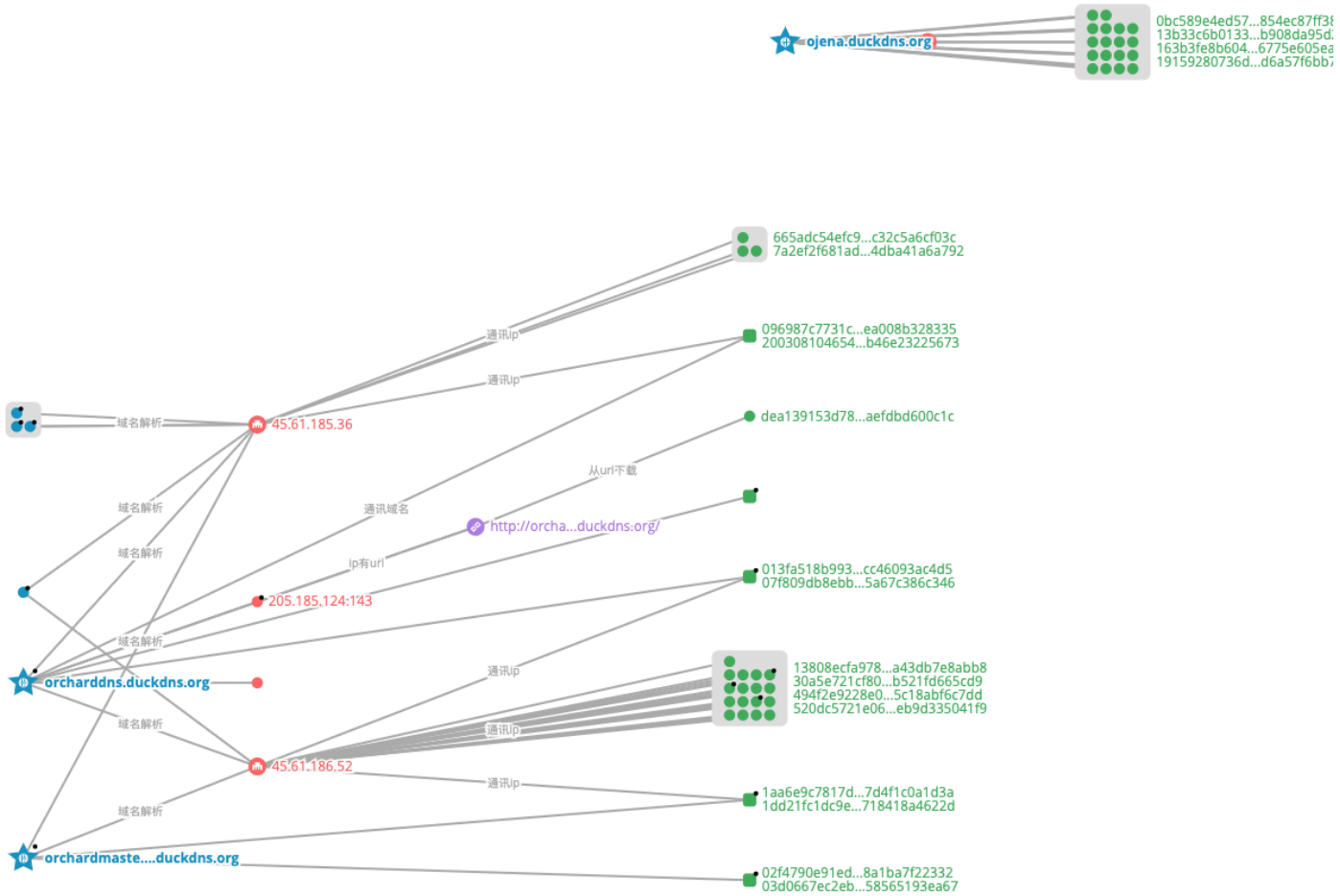
```
v1, orcharddns.duckdns.org
v2, orchardmaster.duckdns.org
v3, ojena.duckdns.org
```

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。

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。

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。

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。

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。

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。

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。

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。

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。

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_
```

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-%
        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
```

— 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

Botnet

A new botnet Orchard Generates DGA Domains with Bitcoin Transaction Information

公有云威胁情报

公有云网络安全威胁情报（202204）

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

See all 114 posts →

DGA is one of the classic techniques for botnets to hide their C2s, attacker only needs to selectively register a very small number of C2 domains, while for the defenders, it is difficult to determine in advance which domain names will be generated and registered. 360 netlab has long focused


Aug 5, 2022 · 13 min read

概述 本文聚焦于云上重点资产的扫描攻击、云服务器总体攻击情况分析、热门漏洞及恶意程序的攻击威胁。 * 360高级威胁狩猎蜜罐系统发现全球9.2万个云服务器IP进行网络扫描、漏洞攻击、传播恶意软件等行为。其中包括国内39家单位所属的云服务资产IP，这些单位涉及政府、医疗、建筑、军工等多个行业。 * 2022年4月，WSO2多个产品和Apache...



· May 11, 2022 · 12 min read