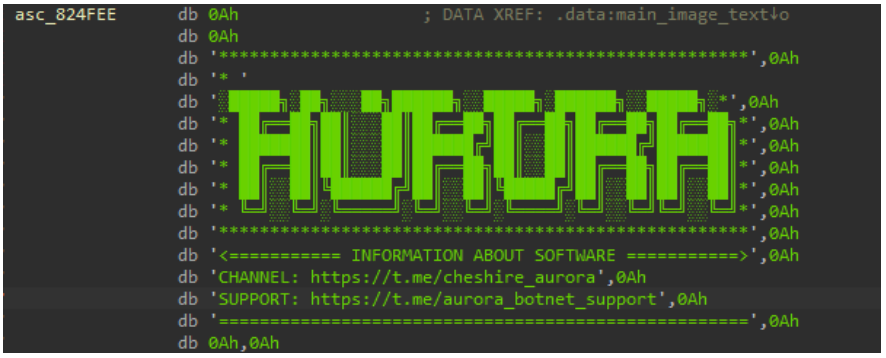
Aurora Stealer Builder

By Mohamed Adel

Published: 2023-04-23 · Archived: 2026-04-05 19:40:55 UTC

in the previous article, I discussed what's inside Aurora Stealer. After the release, @Gi7w0rm provided me samples of some versions of Aurora Stealer builder, a new version that was created recently and another one that was created in 2022. The newer version has some improvements in the builder and new features we will discuss in this article. Before we start this article, it is important to note that the Builder also contains and creates the Web panel to control the bots. This means the binaries we are looking at are actually a hybrid between a builder and a panel.

In `main_main` the first display page is prepared to accept the credentials of the user and start checking them. It first displays an ASCII art of the word Aurora and provides communication channels for contacting the Aurora developers.



After the initial screen, it saves the UUID of the user, with the same function discussed before to make sure that only one user is using the builder.

Then it asks for the login and password of the user

```
.text:0000000000762C25 mov     qword ptr [rsp+0C0h+mw_login], rcx
.text:0000000000762C2D lea     rdx, off_8B27F0 ; "[$] Login: "
.text:0000000000762C34 mov     qword ptr [rsp+0C0h+mw_login+8], rdx
.text:0000000000762C3C mov     rbx, cs:os_Stdout
.text:0000000000762C43 mov     edi, 1
.text:0000000000762C48 mov     rsi, rdi
.text:0000000000762C4B lea     rax, go_itab__os_File_io_Writer
.text:0000000000762C52 lea     rcx, [rsp+0C0h+mw_login]
.text:0000000000762C5A call    fmt_Fprint
.text:0000000000762C5F movups  [rsp+0C0h+mw_login_cred], xmm15
.text:0000000000762C68 lea     rcx, unk_78AA60
.text:0000000000762C6F mov     qword ptr [rsp+0C0h+mw_login_cred], rcx
.text:0000000000762C77 mov     rdx, [rsp+0C0h+var_58]
.text:0000000000762C7C mov     qword ptr [rsp+0C0h+mw_login_cred+8], rdx
.text:0000000000762C84 mov     rbx, cs:os_Stdin
.text:0000000000762C8B lea     rax, go_itab__os_File_io_Reader
.text:0000000000762C92 mov     edi, 1
.text:0000000000762C97 mov     rsi, rdi
.text:0000000000762C9A lea     rcx, [rsp+0C0h+mw_login_cred]
.text:0000000000762CA2 call    fmt_Fscan
.text:0000000000762CA7 movups  [rsp+0C0h+mw_password], xmm15
.text:0000000000762CAD lea     rcx, unk_793580
.text:0000000000762CB4 mov     qword ptr [rsp+0C0h+mw_password], rcx
.text:0000000000762CB9 lea     rcx, off_8B2800 ; "[%] Password: "
.text:0000000000762CC0 mov     qword ptr [rsp+0C0h+mw_password+8], rcx
.text:0000000000762CC5 mov     rbx, cs:os_Stdout
.text:0000000000762CCC lea     rax, go_itab__os_File_io_Writer
.text:0000000000762CD3 lea     rcx, [rsp+0C0h+mw_password]
.text:0000000000762CD8 mov     edi, 1
.text:0000000000762CDD mov     rsi, rdi
.text:0000000000762CE0 call    fmt_Fprint
.text:0000000000762CE5 movups  [rsp+0C0h+mw_login_cred], xmm15
.text:0000000000762CEE lea     rcx, unk_78AA60
.text:0000000000762CF5 mov     qword ptr [rsp+0C0h+mw_login_cred], rcx
.text:0000000000762CFD mov     rcx, [rsp+0C0h+var_60]
.text:0000000000762D02 mov     qword ptr [rsp+0C0h+mw_login_cred+8], rcx
.text:0000000000762D0A mov     rbx, cs:os_Stdin
.text:0000000000762D11 lea     rax, go_itab__os_File_io_Reader
.text:0000000000762D18 mov     edi, 1
.text:0000000000762D1D mov     rsi, rdi
.text:0000000000762D20 lea     rcx, [rsp+0C0h+mw_login_cred]
.text:0000000000762D28 call    fmt_Fscan
```

After the credentials where provided, it calls `main_createAccess`. it saves the string `123` It passes the directory `./cache/Auth.aurora` to a function called `main_exists` that checks if the file exists or not. If it existed it will ask for hand deleting it, if not it will create it.

```
.text:000000000075BCCA lea     rax, [rsp+0E0h+UUID_Aur_tech]
.text:000000000075BCD2 lea     rdi, aAuroraTechnolo ; "AURORA_TECHNOLOGY"
.text:000000000075BCD9 mov     esi, 11h
.text:000000000075BCDE xchg    ax, ax
.text:000000000075BCE0 call    runtime_concatstring2
.text:000000000075BCE5 call    main_MD5_HASH
.text:000000000075BCEA mov     rdx, [rsp+0E0h+_123]
.text:000000000075BCF2 mov     [rsp+0E0h+var_E0], rdx ; __int64
.text:000000000075BCF6 mov     rdx, [rsp+0E0h+_len_3_]
.text:000000000075BCFE mov     [rsp+0E0h+var_D8], rdx ; __int64
.text:000000000075BD03 mov     rcx, [rsp+0E0h+_len_3]
.text:000000000075BD08 lea     rdi, aAurora_0  ; "_aurora_"
.text:000000000075BD12 mov     esi, 8
.text:000000000075BD17 mov     r8, rax
.text:000000000075BD1A mov     r9, rbx
.text:000000000075BD1D lea     r10, aTechnology ; "_technology_"
.text:000000000075BD24 mov     r11d, 0Ch
.text:000000000075BD2A lea     rax, [rsp+0E0h+var_60]
.text:000000000075BD32 mov     rbx, [rsp+0E0h+_123_]
.text:000000000075BD3A call    runtime_concatstring5
.text:000000000075BD3F nop
.text:000000000075BD40 call    main_SHA_HASH
.text:000000000075BD45 mov     [rsp+0E0h+var_20], rax
.text:000000000075BD4D mov     [rsp+0E0h+var_88], rbx
.text:000000000075BD52 mov     rdx, cs:main_GLOBAL_HWID
.text:000000000075BD59 mov     rcx, cs:qword_AFE488
.text:000000000075BD60 lea     rdi, aAuroraTechnolo ; "AURORA_TECHNOLOGY"
.text:000000000075BD67 mov     esi, 11h
.text:000000000075BD6C lea     rax, [rsp+0E0h+var_80]
.text:000000000075BD71 mov     rbx, rdx
.text:000000000075BD74 call    runtime_concatstring2
.text:000000000075BD79 call    main_MD5_HASH
.text:000000000075BD7E mov     rcx, [rsp+0E0h+var_20]
.text:000000000075BD86 mov     rdi, [rsp+0E0h+var_88]
.text:000000000075BD8B call    main_AES_Crypt
.text:000000000075BD90 mov     rdi, rbx
.text:000000000075BD93 mov     rsi, rcx
.text:000000000075BD96 mov     r8d, 1B4h
.text:000000000075BD9C mov     ebx, 13h
.text:000000000075BDA1 mov     rcx, rax
.text:000000000075BDA4 lea     rax, aCacheAuthAuror ; "./cache/Auth.aurora"
.text:000000000075BDAB call    os_WriteFile
```

It appends the UUID and the string `AURORA_TECHNOLOGY` and calculates the MD5 hash to it using the form

`<UUID>AURORA_TECNOLOGY`

after which it takes this hash to make a string in the following form:

`123_aurora_<MD5_OF(<UUID>AURORA_TECNOLOGY)>_technology_123`

```
31 32 33 5F 61 75 72 6F 72 61 5F 65 33 63 30 35 123_aurora_e3c05
37 65 62 62 61 66 64 64 64 65 66 38 63 39 63 33 7ebbafdddef8c9c3
65 35 64 32 32 39 33 35 61 31 39 5F 74 65 63 68 e5d22935a19_tech
6E 6F 6C 6F 67 79 5F 31 32 33 00 00 00 00 00 00 nology_123......
```

Then the SHA1 hash is calculated for this string:

```
.text:000000000075E93C lea     rax, [rsp+0C8h+var_78]
.text:000000000075E941 call    crypto_sha1__digest_Write
.text:000000000075E946 lea     rax, [rsp+0C8h+var_78]
.text:000000000075E94B xor     ebx, ebx
.text:000000000075E94D xor     ecx, ecx
.text:000000000075E94F mov     rdi, rcx
.text:000000000075E952 call    crypto_sha1__digest_Sum
.text:000000000075E957 mov     [rsp+0C8h+var_10], rax
```

It generates the first string again and its MD5 hash. It uses the MD5 hash as a key for the AES GCM encryption routine. The generated bytes are then written to `./cache/Auth.aurora`

To know what was written to the file, we can use this script:

```python
from Crypto.Cipher import AES
import binascii

# key is MD5 hash of <UUID>AURORA_TECHNOLOGY
key = b"<KEY>"
# Auth.aurora content
cipher = "<CIPHER>"

data = binascii.unhexlify(cipher)
nonce, tag = data[:12], data[-16:]
cipher = AES.new(key, AES.MODE_GCM, nonce)
cleartext = cipher.decrypt_and_verify(data[12:-16], tag)
print(cleartext)
# cleartext is SHA1 hash of the string "123_aurora_<MD5_OF(<UUID>AURORA_TECNOLOGY)>_technology_123 "
```

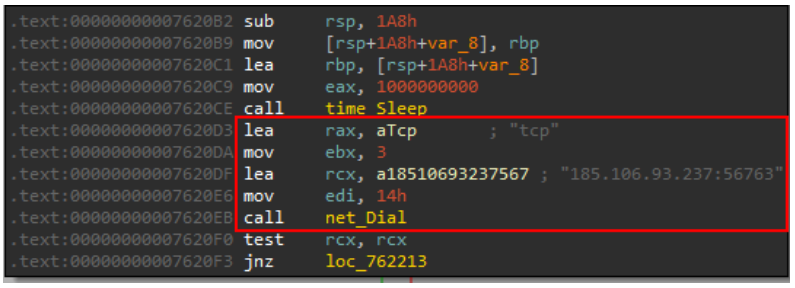
which shows us the SHA-1 Hash of the string: `123_aurora_<MD5_OF(<UUID>AURORA_TECNOLOGY)>_technology_123`

Going back to `main_main` , where it creates yet another hash:

```
.text:0000000000762D40                 call    main_CreateAccess
.text:0000000000762D45                 mov     rcx, [rsp+0C0h+login]
.text:0000000000762D4A                 mov     rbx, [rcx]
.text:0000000000762D4D                 mov     rdx, [rsp+0C0h+pass]
.text:0000000000762D52                 mov     r8, [rdx]
.text:0000000000762D55                 mov     rcx, [rcx+8]
.text:0000000000762D59                 mov     r9, [rdx+8]
.text:0000000000762D5D                 lea     rax, [rsp+0C0h+var_80]
.text:0000000000762D62                 lea     rdi, aAurora2023Tech ; "_Aurora_2023_Technology_"
.text:0000000000762D69                 mov     esi, 18h
.text:0000000000762D6E                 call    runtime_concatstring3
.text:0000000000762D73                 call    main_SHA_HASH
.text:0000000000762D78                 mov     cs:sha1_len, rbx
.text:0000000000762D7F                 cmp     cs:runtime_writeBarrier, 0
.text:0000000000762D86                 jnz     short loc_762D91
.text:0000000000762D88                 mov     cs:main_AUTH_HASH, rax
.text:0000000000762D8F                 jmp     short loc_762D9D
.text:0000000000762D91 ; ---------------------------------------------------------------------------
.text:0000000000762D91
.text:0000000000762D91 loc_762D91:                             ; CODE XREF: main_main+286↑j
.text:0000000000762D91                 lea     rdi, main_AUTH_HASH
.text:0000000000762D98                 call    runtime_gcWriteBarrier
.text:0000000000762D9D
.text:0000000000762D9D loc_762D9D:                             ; CODE XREF: main_main+28F↑j
.text:0000000000762D9D                 nop     dword ptr [rax]
.text:0000000000762DA0                 call    main_SERVER
.text:0000000000762DA5
.text:0000000000762DA5 loc_762DA5:                             ; CODE XREF: main_main+2A6↓j
.text:0000000000762DA5                 nop
.text:0000000000762DA6                 jmp     short loc_762DA5
.text:0000000000762DA8 ; ---------------------------------------------------------------------------
.text:0000000000762DA8                 call    runtime_deferreturn
.text:0000000000762DAD                 mov     rbp, [rsp+0C0h+var_8]
.text:0000000000762DB5                 add     rsp, 0C0h
.text:0000000000762DBC                 retn
```

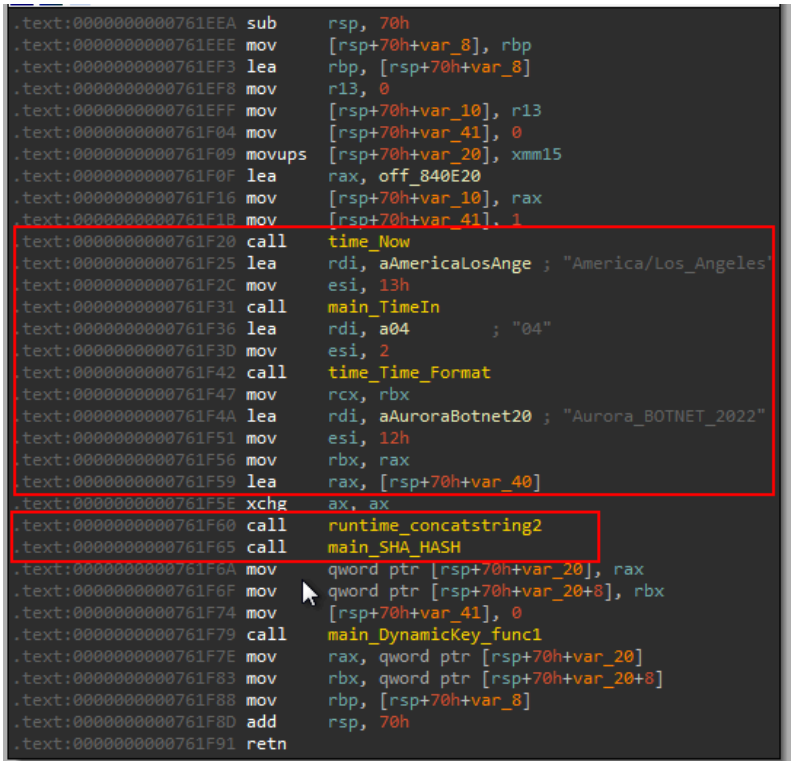
This time, the password and login is used to create a string using the following form `<LOGIN>_*Aurora_2023_Technology_<PASS>.` then it calculates the SHA1 hash of it.*

Then, it calls `main_server` . This could be where the authentication of the user happens, just a hypothesis.



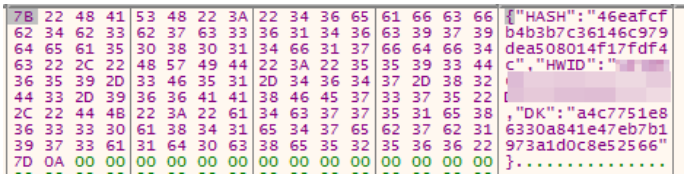
it sleeps 1000000000 nanoseconds. Then it makes a TCP connection with `185.106.93.237:56763` which seems to be the server where user authentication is done.

If the connection is established, it calls `main_DynamicKey` which generates a key based on the current minutes in the current time, In `America/Los_Angeles` time format.

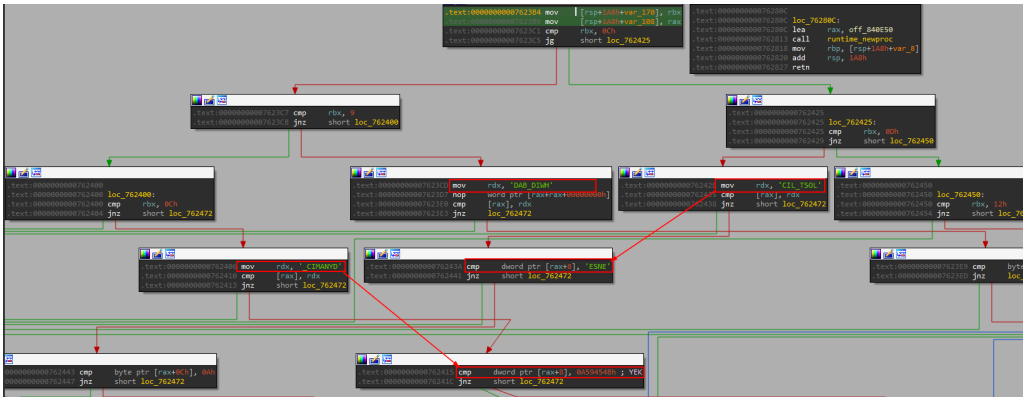


and calculate the SHA1 hash of it.

Back in the `main_Server` function the builder then puts all the hashes in JSON format to be sent to the server.



the remote server then verifies the given data and response with one of the few response strings below:



Response	Action
HWID_BAD	[Aurora] HWID has a different value on the license server, write support
NOT_FOUND_ACCOUNT	[Aurora] Account has been not found, wrong login or password.
LOST_LICENSE	[Aurora] License expired.
DYNAMIC_KEY	[Aurora] Dynamic key wrong, check time your OS or write support.

I tried to emulate the C2 communication with fakenet. After a very long time trying to do that. it works to respond to it with the format of data it waits for, but there is something still missing.

I edited the configs of the `TCPListener` of fakenet as can be seen below:

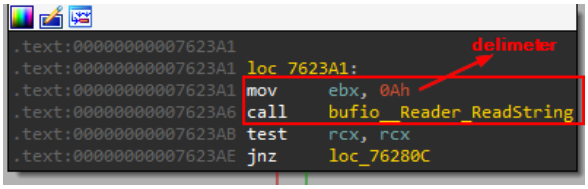
1. In `default.ini` edit the default configs to the following:

```
[RawTCPListener]
Enabled:    True
Port:       56763 # port it comm over
Protocol:   TCP
Listener:   RawListener
UseSSL:     No
Timeout:    100
Hidden:     False
# To read about customizing responses, see docs/CustomResponse.md
Custom:   sample_custom_response.ini
```

2. Create or use the `sample_custom_response.ini` provided to contain the following, this is already set by default:

```
[ExampleTCP]
InstanceName:    RawTCPListener
TcpDynamic:      CustomProviderExample.py
```

1. The builder waits for a JSON string delimited by the character `0x0A` if this is not in the response it will wait forever.

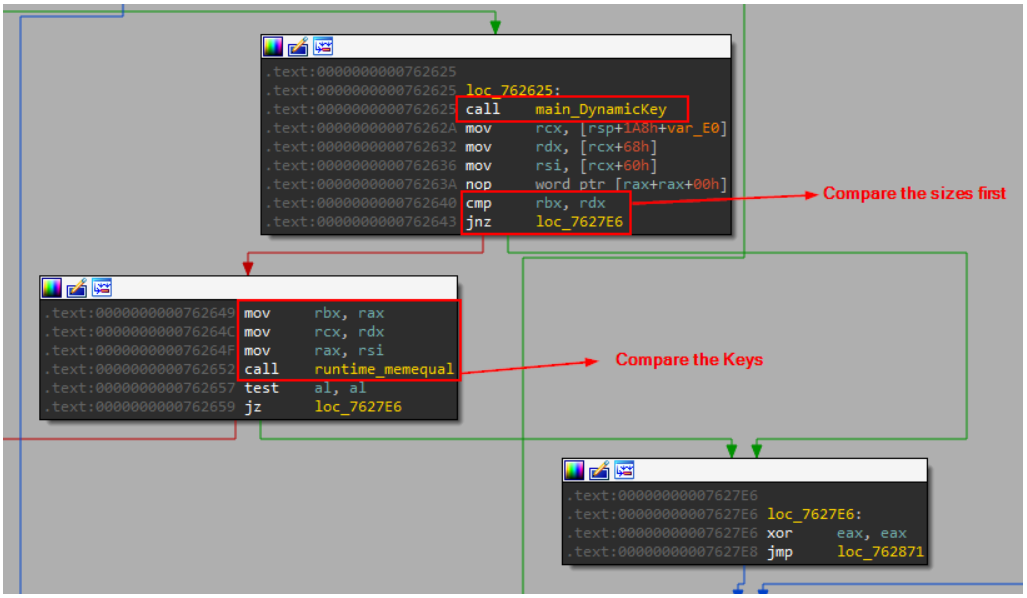


As a result `CustomProviderExample.py` should contain a JSON string ending with `0x0A` , I was testing with the following code:

```
def HandleTcp(sock):
    """Handle a TCP buffer.

    Parameters
    ----------
    sock : socket
        The connected socket with which to recv and send data
    """
    while True:
        try:
            data = None
            data = sock.recv(1024)
        except socket.timeout:
            pass

        if not data:
            break

        resp = b'{"Test":"test","Test2":"Test2"}\x0A'
        sock.sendall(resp)
```

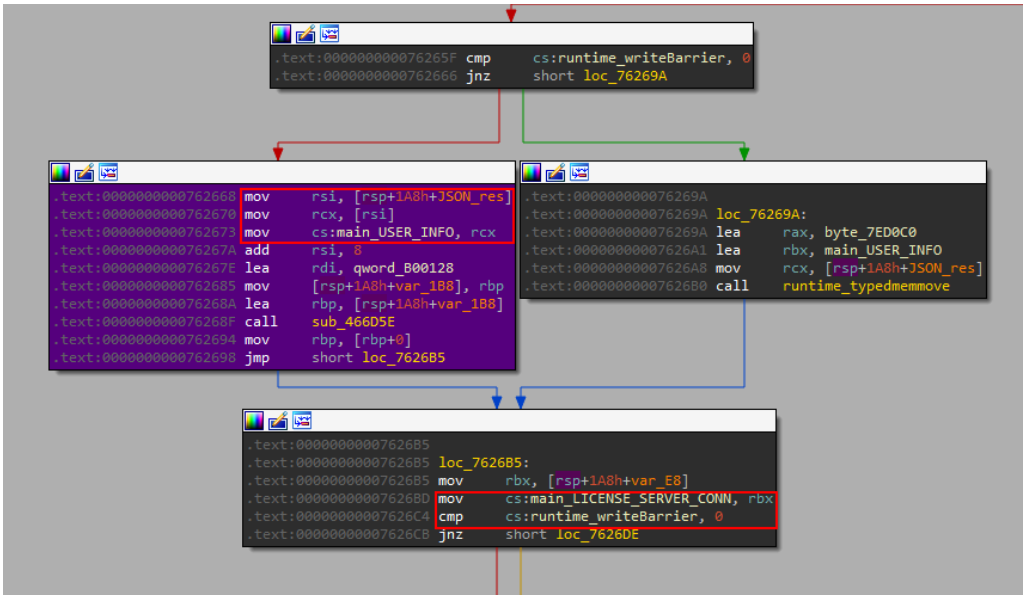
A value of the JSON string accepted must be the Dynamic key which is generated based on the local time of the user.

This Dynamic key is calculated again and the two values are compared in order to check if the sample is being debugged. Nice!

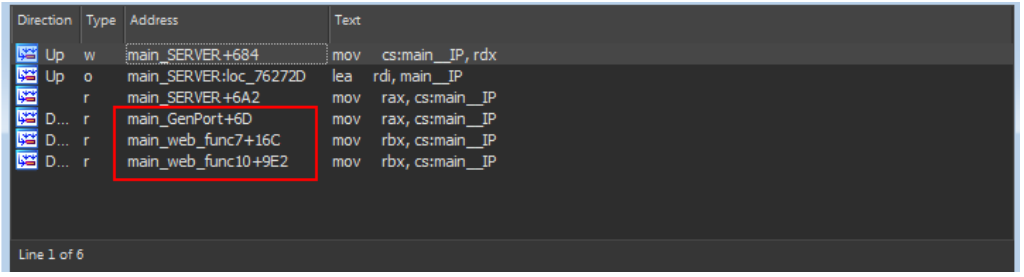


License info and IP used

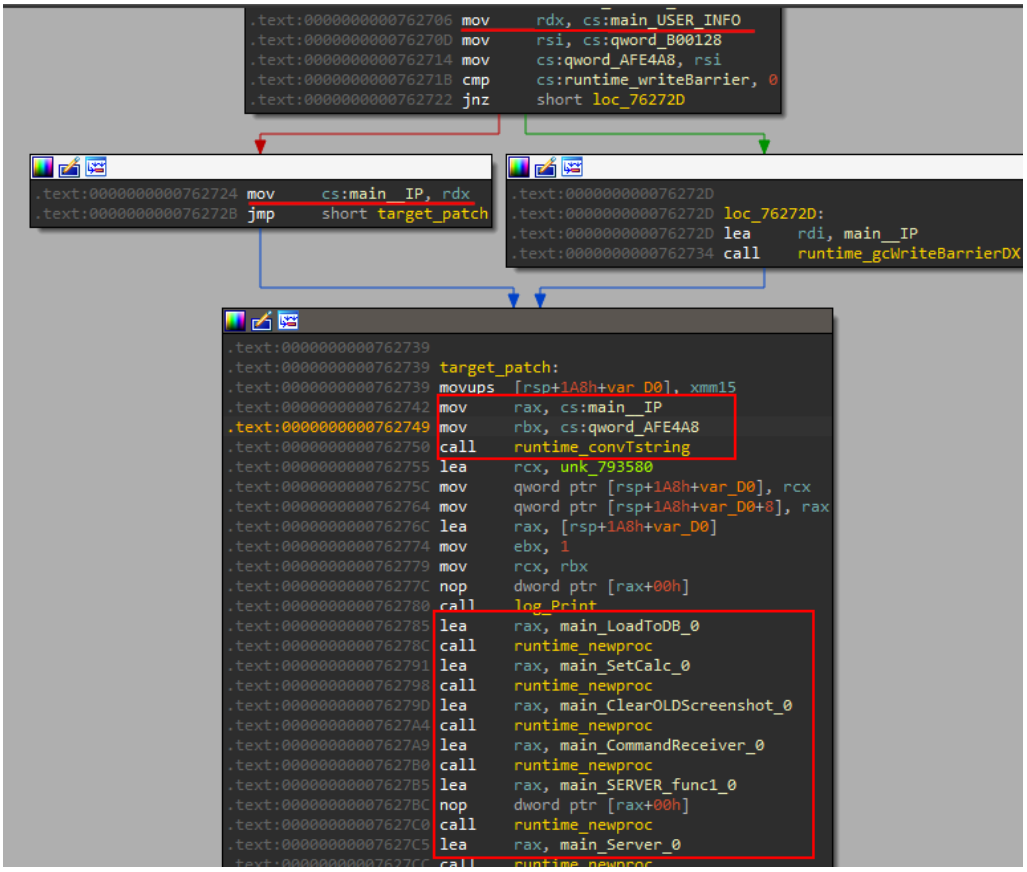
The JSON strings also contain some other information about the User and the license



Also, it contains an IP that is used later in some other interesting functions. the author expects only one IP to be used by the builder.



It calls `convTstring` which takes a generic value -any type- and converts it to a string. I don't really know why it calls `convTstring` as it is an IP it would be passed as a string in the JSON. maybe later we realize what's going on here.



We see some calls to `runtime.newProc` . This function generates a new go running function and put it in a running Queue of other go functions waiting to run. This is generated by the compiler when using `go` keyword. Interested topic hah? Read more about it here. Sadly it makes debugging more difficult.

Back to the JSON data, it's decoded with `json.Unmashal` function which takes a structure as an input and with the second parameter being the data in bytes. How is the data mapped to the structure? Well, according to Go documentation

> How does `Unmarshal` identify the fields in which to store the decoded data? For a given JSON key `"Foo"`, `Unmarshal` will look through the destination struct's fields to find (in order of preference):
>
> - An exported field with a tag of `"Foo"` (see the Go spec for more on struct tags),
> - An exported field named `"Foo"`, or
> - An exported field named `"FOO"` or `"FoO"` or some other case-insensitive match of `"Foo"`.
>
> What happens when the structure of the JSON data doesn't exactly match the Go type?
>
> `Unmarshal` will decode only the fields that it can find in the destination type

So, we should guess the names of the JSON data. One of them is `Dynamic key` but we should figure out how it's decoded.

We can use the pattern of the previously sent data, It was called `DK` . Sadly, this and other attempts didn't work. So, I will continue the other things only static in IDA.

Main Functionality

The main functionality of the builder is invoked with a series of goroutine calls. Each called function is preparing some data to be used later or to start the server itself. This serves as the main function of the builder.

The first function of the series of `newProc` calls is `main_LoadToDB` which loads a very huge file called `geo.aurora` that contains a list of IP ranges all over the world.

```
.text:000000000075F30A sub     rsp, 40h
.text:000000000075F30E mov     [rsp+40h+var_8], rbp
.text:000000000075F313 lea     rbp, [rsp+40h+var_8]
.text:000000000075F318 movups  [rsp+40h+var_18], xmm15
.text:000000000075F31E lea     rdx, unk_793580
.text:000000000075F325 mov     qword ptr [rsp+40h+var_18], rdx
.text:000000000075F32A lea     rsi, off_8B2750 ; "[Server] Load - GEO Database"
.text:000000000075F331 mov     qword ptr [rsp+40h+var_18+8], rsi
.text:000000000075F336 lea     rax, [rsp+40h+var_18]
.text:000000000075F33B mov     ebx, 1
.text:000000000075F340 mov     rcx, rbx
.text:000000000075F343 call    log_Print
.text:000000000075F348 nop
.text:000000000075F349 lea     rax, aGeoGeoAurora ; "./geo/geo.Aurora"
.text:000000000075F350 mov     ebx, 10h
.text:000000000075F355 call    os_ReadFile
.text:000000000075F35A lea     rdi, unk_784FE0
.text:000000000075F361 lea     rsi, main__DB_GEO
.text:000000000075F368 call    encoding_json_Unmarshal
.text:000000000075F36D movups  [rsp+40h+var_18], xmm15
.text:000000000075F373 lea     rdx, unk_793580
.text:000000000075F37A mov     qword ptr [rsp+40h+var_18], rdx
.text:000000000075F37F lea     rdx, off_8B2760 ; "[Server] Load - Success"
.text:000000000075F386 mov     qword ptr [rsp+40h+var_18+8], rdx
.text:000000000075F38B lea     rax, [rsp+40h+var_18]
.text:000000000075F390 mov     ebx, 1
.text:000000000075F395 mov     rcx, rbx
.text:000000000075F398 call    log_Print
.text:000000000075F39D mov     rbp, [rsp+40h+var_8]
.text:000000000075F3A2 add     rsp, 40h
.text:000000000075F3A6 retn
```

Viewing the cross-reference we can deduce that it is used to identify the geo-location of a victim.

Directio	Type	Address	Text
	o	main_LoadToDB+61	lea rsi, main__DB_GEO
D...	r	main_GetGeo+7A	mov rdx, cs:main__DB_GEO

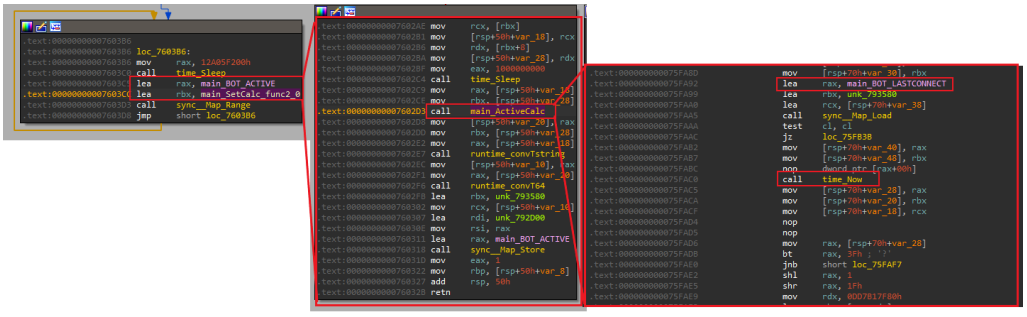
A sample of the content of `geo.Aurora` can be seen below. The file contains ~380MB of data like this.

```
[
 {
   "Country_short": "AU",
   "City": "Queensland",
   "Region": "",
   "Zipcode": "",
   "Timezone": "",
   "In": "1.0.0.0",
   "Out": "1.0.0.255"
 },
 {
   "Country_short": "CN",
   "City": "Fujian",
   "Region": "",
   "Zipcode": "",
   "Timezone": "",
   "In": "1.0.1.0",
   "Out": "1.0.3.255"
 },
 {
   "Country_short": "AU",
   "City": "Victoria",
   "Region": "",
   "Zipcode": "",
   "Timezone": "",
   "In": "1.0.4.0",
   "Out": "1.0.7.255"
 },
 {
```

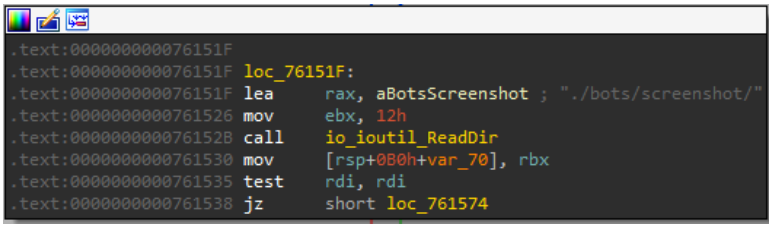
```
    "Country_short": "CN",
    "City": "Guangdong",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.8.0",
    "Out": "1.0.15.255"
  },
  {
    "Country_short": "JP",
    "City": "Tokyo",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.16.0",
    "Out": "1.0.16.255"
  },
  {
    "Country_short": "JP",
    "City": "Tokyo",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.17.0",
    "Out": "1.0.31.255"
  },
  {
    "Country_short": "CN",
    "City": "Guangdong",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.32.0",
    "Out": "1.0.63.255"
  },
  {
    "Country_short": "JP",
    "City": "Hiroshima",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.64.0",
    "Out": "1.0.64.255"
  },
  {
    "Country_short": "JP",
    "City": "Hiroshima",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.65.0",
    "Out": "1.0.66.255"
  },
  {
    "Country_short": "JP",
    "City": "Hiroshima",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.67.0",
    "Out": "1.0.67.255"
  },
  {
    "Country_short": "JP",
    "City": "Hiroshima",
```

```
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.68.0",
    "Out": "1.0.68.127"
  },
  {
    "Country_short": "JP",
    "City": "Miyagi",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.68.128",
    "Out": "1.0.69.255"
  },
  {
    "Country_short": "JP",
    "City": "Hiroshima",
    "Region": "",
    "Zipcode": "",
    "Timezone": "",
    "In": "1.0.70.0",
    "Out": "1.0.71.255"
  },
....
]
```

The second function is to get the status of the infected systems. This includes a check if the bot is active, the last connection time of the bot, and the current time.



The third function deletes all the screenshots stored in the `bot` directory!



It sorts the pictures to be deleted by `_` in it, then it gets what has `ACTUAL` word in it, lastly, it deletes the file extension `.png` from the string using `strings.Trim` and the new string should be a number as it calls `strconv.atoi` and then gets the current time. What a mess!

```
.text:00000000007615A1  mov     rsi, [rdx+30h]
.text:00000000007615A5  mov     rax, rbx
.text:00000000007615A8  call    rsi
.text:00000000007615AA  lea     rcx, asc_804EF3 ; "_"
.text:00000000007615B1  mov     edi, 1
.text:00000000007615B6  xor     esi, esi
.text:00000000007615B8  mov     r8, 0FFFFFFFFFFFFFFFFh
.text:00000000007615BF  nop
.text:00000000007615C0  call    strings_genSplit
.text:00000000007615C5  test    rbx, rbx
.text:00000000007615C8  jz      loc_7616DF
.text:00000000007615CE  cmp     rbx, 1
.text:00000000007615D2  jbe     loc_7616FA
.text:00000000007615D8  mov     [rsp+0B0h+var_58], rax
.text:00000000007615DD  mov     rdx, [rax+10h]
.text:00000000007615E1  mov     rbx, [rax+18h]
.text:00000000007615E5  lea     rcx, aActual    ; "ACTUAL"
.text:00000000007615EC  mov     edi, 6
.text:00000000007615F1  mov     rax, rdx
.text:00000000007615F4  call    strings_Index
.text:00000000007615F9  nop     dword ptr [rax+00000000h]
.text:0000000000761600  test    rax, rax
.text:0000000000761603  jge     loc_7616DF
.text:0000000000761609  mov     rdx, [rsp+0B0h+var_58]
.text:000000000076160E  mov     rax, [rdx+10h]
.text:0000000000761612  mov     rbx, [rdx+18h]
.text:0000000000761616  lea     rcx, aPng       ; ".png"
.text:000000000076161D  mov     edi, 4
.text:0000000000761622  call    strings_Trim
.text:0000000000761627  call    strconv_Atoi
.text:000000000076162C  mov     [rsp+0B0h+var_78], rax
.text:0000000000761631  call    time_Now
.text:0000000000761636  mov     [rsp+0B0h+var_28], rax
.text:000000000076163E  mov     [rsp+0B0h+var_20], rbx
.text:0000000000761646  mov     [rsp+0B0h+var_18], rcx
.text:000000000076164E  nop
.text:000000000076164F  mov     rdx, [rsp+0B0h+var_28]
.text:0000000000761657  bt      rdx, 3Fh ; '?'
.text:000000000076165C  jnb     short loc_761675
.text:000000000076165E  shl     rdx, 1
.text:0000000000761661  shr     rdx, 1Fh
```

It then proceeds to finally delete the file.

```
.text:000000000076169D mov     rcx, [rsp+0B0h+var_68]
.text:00000000007616A2 mov     rcx, [rcx+30h]
.text:00000000007616A6 mov     rax, [rsp+0B0h+var_48]
.text:00000000007616AB call    rcx
.text:00000000007616AD mov     ecx, 12h
.text:00000000007616B2 mov     rdi, rax
.text:00000000007616B5 mov     rsi, rbx
.text:00000000007616B8 xor     eax, eax
.text:00000000007616BA lea     rbx, aBotsScreenshot ; "./bots/screenshot/"
.text:00000000007616C1 call    runtime_concatstring2
.text:00000000007616C6 call    os_Remove
.text:00000000007616CB mov     rcx, 0DD7B17F80h
.text:00000000007616D5 mov     rdx, 0FFFFFFF1886E0900h
```

Command Receiver

The next function is `main_CommandReceiver`. It queues the commands received by the builder.

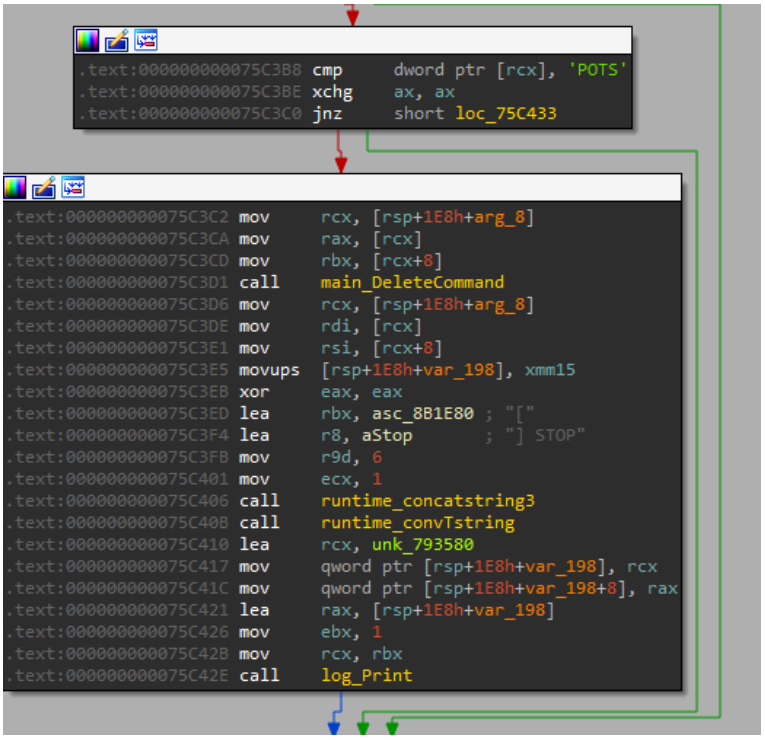
```
.text:000000000075CFD6
.text:000000000075CFD6 loc_75CFD6:
.text:000000000075CFD6 mov     rax, 2540BE400h
.text:000000000075CFE0 call    time_Sleep
.text:000000000075CFE5 lea     rax, main_CMD_QUEUE
.text:000000000075CFEC lea     rbx, main_CommandReceiver_func2_0
.text:000000000075CFF3 call    sync_Map_Range
.text:000000000075CFF8 jmp     short loc_75CFD6
```

The function `map.Range` has the definition:

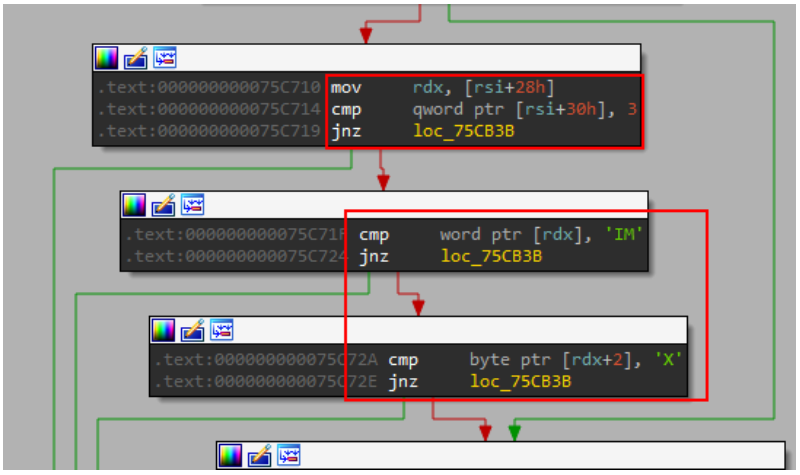
```
1  func (m *Map) Range(f func(key, value any) bool)
```

where f is a function called for each <key,value> pair. So the variable `CMD_QUEUE` would contain the received commands.

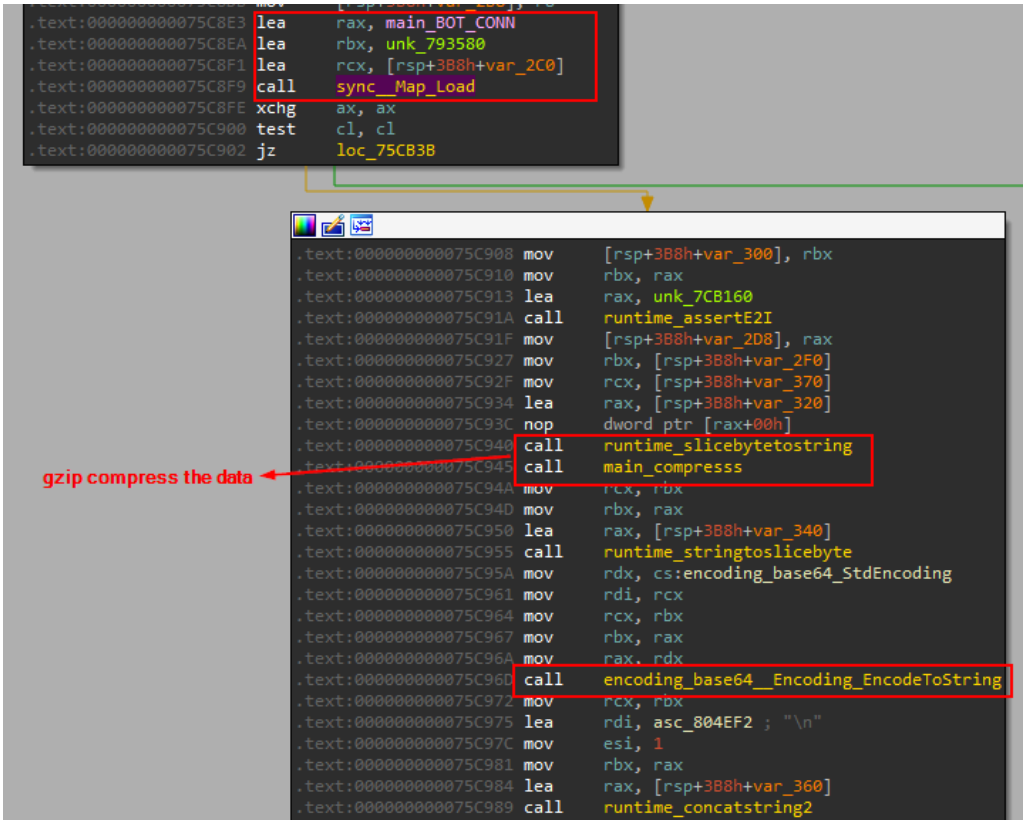
Going through the function `main_CommandReceiver_func2` we see that the software first checks if the received command is `STOP`. If the STOP command is received, the builder exits.



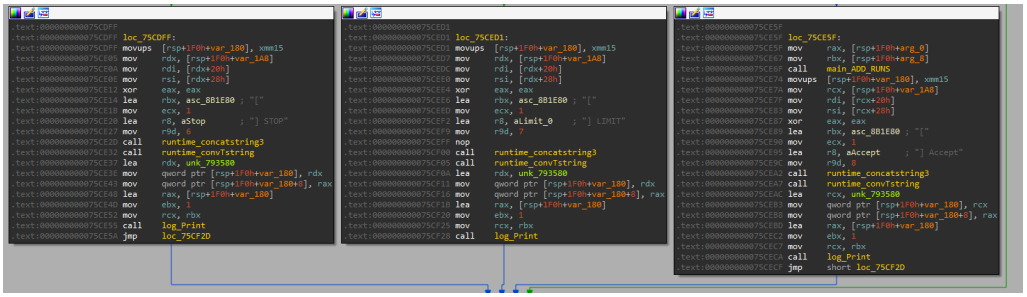
For all other commands, it goes to another function `main_CommandReceiver_func2_1`. It's expecting a 3-character long command `MIX`.



It packs data about the victims with GZip and base64 encode it then, stores it back using `map.store`



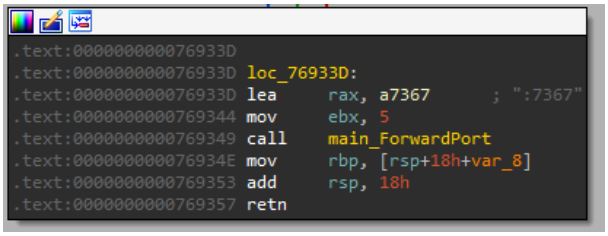
There were some log messages related to other commands here. However, I couldn't figure out how the commands are treated. Based on the sample I discussed in a previous article, I guess this is connected to the messages sent from the victim machine.



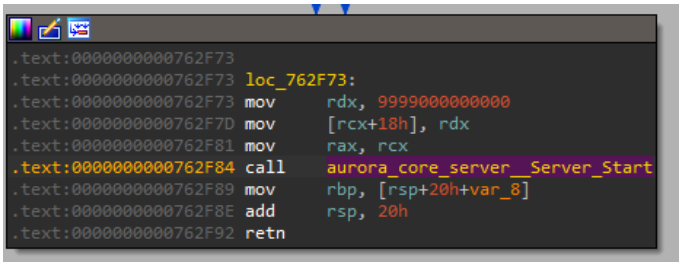
Main server functionality

The server is now ready to work and build the graphical interface of the builder to view the victim's data and state and further use the victims as Bots and Stealer hosting servers using SFTP.

Next function is `main_SERVER_func1` it calls `main_ForwardPort` with argument `:7367`



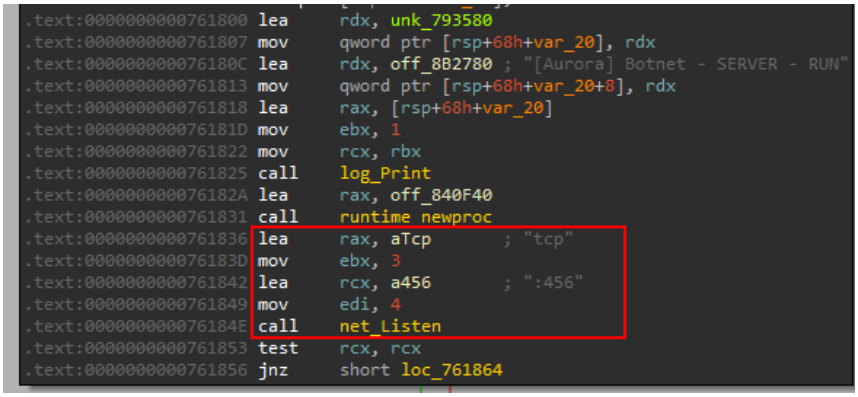
Then this function calls `aurora_core_server__Server_Start` , this long value is passed with the port number passed to its driver function



This function starts the main server that displays the dashboard. I tried to adjust the execution to continue, but the program crashed.

Note: `SixSixSix` is the author of the Stealer and not my username.

Back to function `main_Server_0` (`main_Server`).

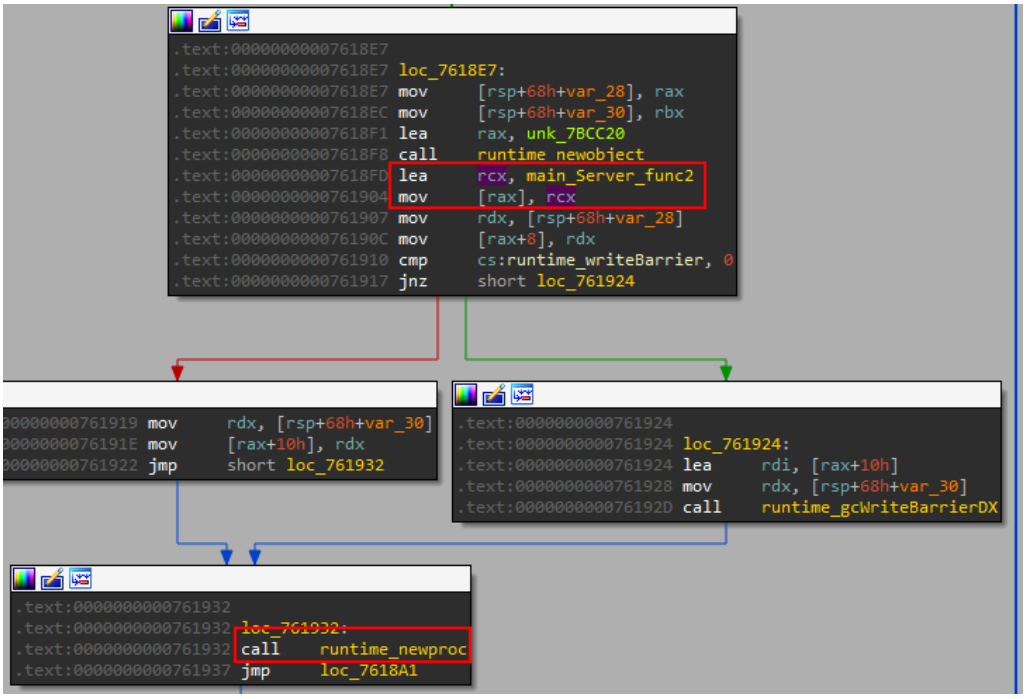


It logs the start of the server in the main display.

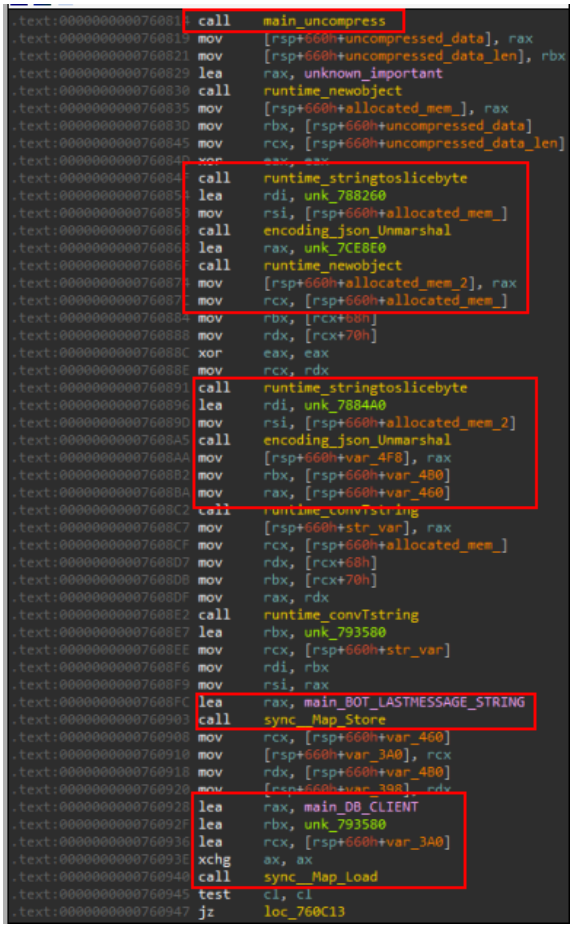
The server is started using `net.Listen` function that takes the protocol = `tcp` and port = `456` .

Main Client

After setting up the Server, the function `main_server_func2` is called.



This function only calls the `main_Client` function.



Handling incoming data

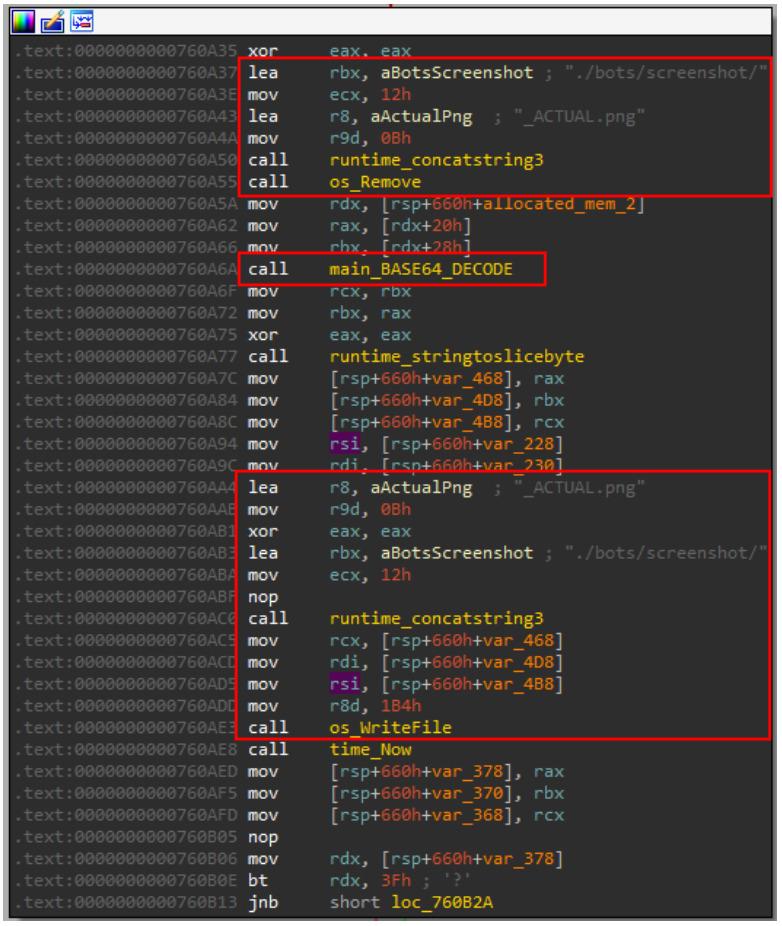
To handle incoming data from the victim, the panel/builder reads the data on the listening port using `bufio__Reader_ReadString`. This data must be delimited by `0x0A` as discussed previously. It comes in a compressed format, so the function `main_uncompress` is used to decompress it.

```
.text:000000000075E0E5                mov     [rsp+58h+var_39], 1
.text:000000000075E0EA                call    main_BASE64_DECODE
.text:000000000075E0EF                mov     rcx, rbx
.text:000000000075E0F2                mov     rbx, rax
.text:000000000075E0F5                xor     eax, eax
.text:000000000075E0F7                call    runtime_stringtoslicebyte
.text:000000000075E0FC                mov     [rsp+58h+var_28], rax
.text:000000000075E101                mov     [rsp+58h+var_38], rbx
.text:000000000075E106                mov     [rsp+58h+var_30], rcx
.text:000000000075E10B                lea     rax, unk_7CD3E0
.text:000000000075E112                call    runtime_newobject
.text:000000000075E117                mov     rcx, [rsp+58h+var_38]
.text:000000000075E11C                mov     [rax+8], rcx
.text:000000000075E120                mov     rcx, [rsp+58h+var_30]
.text:000000000075E125                mov     [rax+10h], rcx
.text:000000000075E129                cmp     cs:runtime_writeBarrier, 0
.text:000000000075E130                jnz     short loc_75E13C
.text:000000000075E132                mov     rcx, [rsp+58h+var_28]
.text:000000000075E137                mov     [rax], rcx
.text:000000000075E13A                jmp     short loc_75E149
.text:000000000075E13C ; ---------------------------------------------------------------------------
.text:000000000075E13C
.text:000000000075E13C loc_75E13C:                            ; CODE XREF: main_uncompress+90↑j
.text:000000000075E13C                mov     rdi, rax
.text:000000000075E13F                mov     rcx, [rsp+58h+var_28]
.text:000000000075E144                call    runtime_gcWriteBarrierCX
.text:000000000075E149
.text:000000000075E149 loc_75E149:                            ; CODE XREF: main_uncompress+9A↑j
.text:000000000075E149                mov     qword ptr [rax+18h], 0
.text:000000000075E151                mov     qword ptr [rax+20h], 0FFFFFFFFFFFFFFFFh
.text:000000000075E159                mov     rbx, rax
.text:000000000075E15C                lea     rax, go_itab__bytes_Reader_io_Reader
.text:000000000075E163                call    compress_gzip_NewReader
.text:000000000075E168                test    rbx, rbx
.text:000000000075E16B                jnz     loc_75E1F1
.text:000000000075E171                nop
.text:000000000075E172                mov     rbx, rax
.text:000000000075E175                lea     rax, go_itab__compress_gzip_Reader_io_Reader
.text:000000000075E17C                nop     dword ptr [rax+00h]
.text:000000000075E180                call    io_ReadAll
.text:000000000075E185                test    rdi, rdi
.text:000000000075E188                jz      short loc_75E1BC
```

To do so, the function takes the base64 encoded data and decodes it, then it is decompressed using GZip. You might remember from my last article, that this is the way the data was sent from the victim's device.

The data is in form of JSON so it's extracted with a call to `json.Unmarshal` . The resulting data is then stored in a victim database file. The last message is additionally stored in the map function.

One of the first packets received from the victim is a large base64 blob. After decoding it using the above-mentioned method, it can be seen that this blob is a screenshot from the victim's machine.



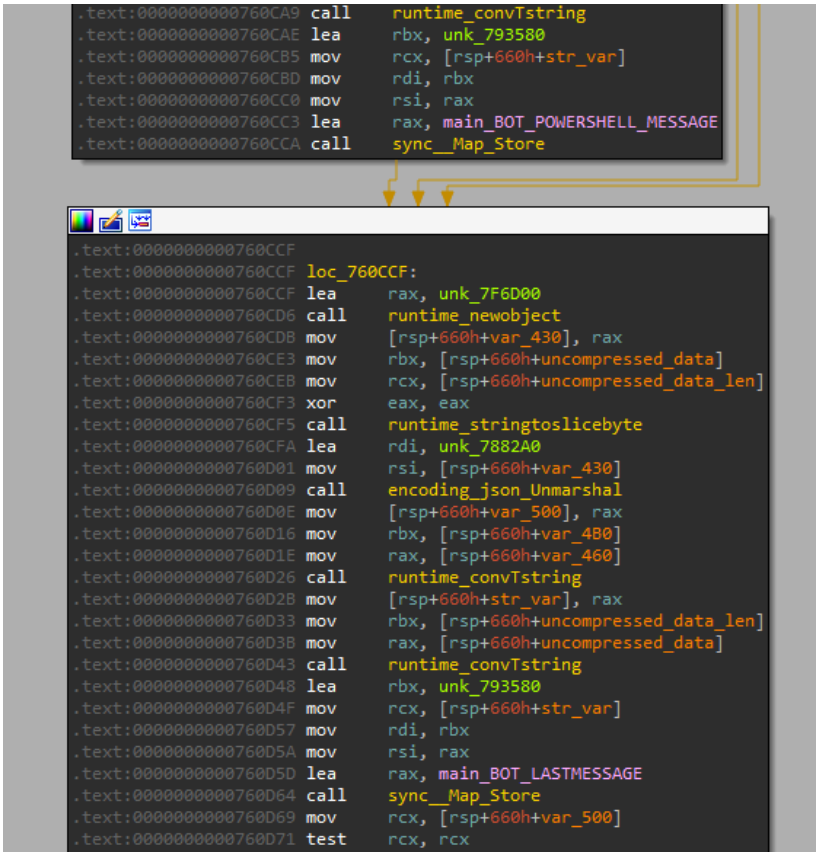
```
.text:0000000000760A35 xor     eax, eax
.text:0000000000760A37 lea     rbx, aBotsScreenshot ; "./bots/screenshot/"
.text:0000000000760A3E mov     ecx, 12h
.text:0000000000760A43 lea     r8, aActualPng  ; "_ACTUAL.png"
.text:0000000000760A4A mov     r9d, 0Bh
.text:0000000000760A50 call    runtime_concatstring3
.text:0000000000760A55 call    os_Remove
.text:0000000000760A5A mov     rdx, [rsp+660h+allocated_mem_2]
.text:0000000000760A62 mov     rax, [rdx+20h]
.text:0000000000760A66 mov     rbx, [rdx+28h]
.text:0000000000760A6A call    main_BASE64_DECODE
.text:0000000000760A6F mov     rcx, rbx
.text:0000000000760A72 mov     rbx, rax
.text:0000000000760A75 xor     eax, eax
.text:0000000000760A77 call    runtime_stringtoslicebyte
.text:0000000000760A7C mov     [rsp+660h+var_468], rax
.text:0000000000760A84 mov     [rsp+660h+var_4D8], rbx
.text:0000000000760A8C mov     [rsp+660h+var_4B8], rcx
.text:0000000000760A94 mov     rsi, [rsp+660h+var_228]
.text:0000000000760A9C mov     rdi, [rsp+660h+var_230]
.text:0000000000760AA4 lea     r8, aActualPng  ; "_ACTUAL.png"
.text:0000000000760AAB mov     r9d, 0Bh
.text:0000000000760AB1 xor     eax, eax
.text:0000000000760AB3 lea     rbx, aBotsScreenshot ; "./bots/screenshot/"
.text:0000000000760ABA mov     ecx, 12h
.text:0000000000760ABF nop
.text:0000000000760AC0 call    runtime_concatstring3
.text:0000000000760AC5 mov     rcx, [rsp+660h+var_468]
.text:0000000000760ACD mov     rdi, [rsp+660h+var_4D8]
.text:0000000000760AD5 mov     rsi, [rsp+660h+var_4B8]
.text:0000000000760ADD mov     r8d, 1B4h
.text:0000000000760AE3 call    os_WriteFile
.text:0000000000760AE8 call    time_Now
.text:0000000000760AED mov     [rsp+660h+var_378], rax
.text:0000000000760AF5 mov     [rsp+660h+var_370], rbx
.text:0000000000760AFD mov     [rsp+660h+var_368], rcx
.text:0000000000760B05 nop
.text:0000000000760B06 mov     rdx, [rsp+660h+var_378]
.text:0000000000760B0E bt      rdx, 3Fh ; '?'
.text:0000000000760B13 jnb     short loc_760B2A
```

This image is used to update the screenshot that contains `_ACTUAL.png` . The old one is then deleted.

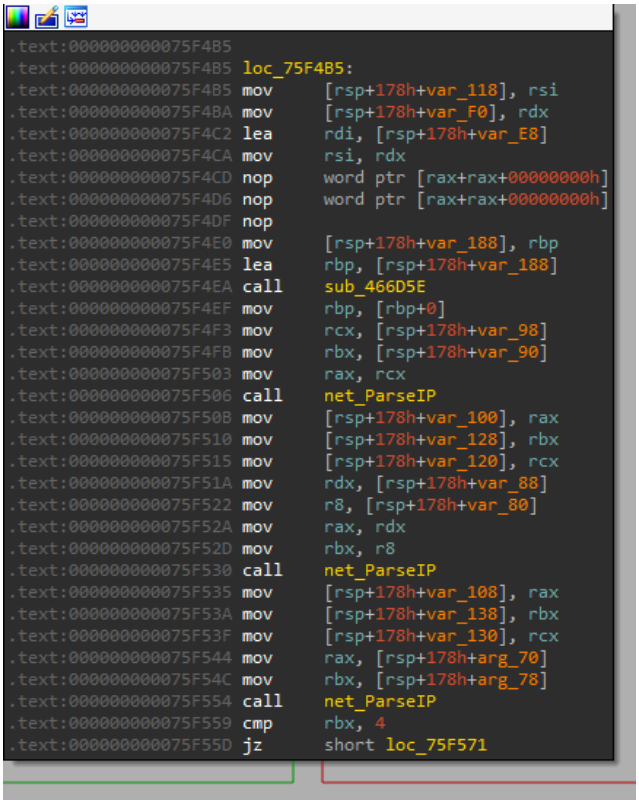
```
.text:0000000000760B2A loc_760B2A:
.text:0000000000760B2A mov     rcx, 0FFFFFFF1886E0900h
.text:0000000000760B34 lea     rax, [rbx+rcx]
.text:0000000000760B38 mov     ebx, 0Ah
.text:0000000000760B3D nop     dword ptr [rax]
.text:0000000000760B40 call    strconv_FormatInt
.text:0000000000760B45 mov     [rsp+660h+var_498], rax
.text:0000000000760B4D mov     [rsp+660h+var_518], rbx
.text:0000000000760B55 mov     rcx, [rsp+660h+allocated_mem_2]
.text:0000000000760B5D mov     rdx, [rcx+20h]
.text:0000000000760B61 mov     rcx, [rcx+28h]
.text:0000000000760B65 mov     rax, rdx
.text:0000000000760B68 mov     rbx, rcx
.text:0000000000760B6B call    main_BASE64_DECODE
.text:0000000000760B70 mov     rcx, rbx
.text:0000000000760B73 mov     rbx, rax
.text:0000000000760B76 xor     eax, eax
.text:0000000000760B78 call    runtime_stringtoslicebyte
.text:0000000000760B7D mov     [rsp+660h+var_470], rax
.text:0000000000760B85 mov     [rsp+660h+var_4E0], rbx
.text:0000000000760B8D mov     [rsp+660h+var_4C0], rcx
.text:0000000000760B95 mov     rsi, [rsp+660h+var_228]
.text:0000000000760B9D mov     rdi, [rsp+660h+var_230]
.text:0000000000760BA5 lea     rdx, aPng       ; ".png"
.text:0000000000760BAC mov     qword ptr [rsp+660h+var_660], rdx ; char
.text:0000000000760BB0 mov     [rsp+660h+var_658], 4 ; __int64
.text:0000000000760BB9 lea     r8, asc_804EF3  ; "_"
.text:0000000000760BC0 mov     r9d, 1
.text:0000000000760BC6 mov     r10, [rsp+660h+var_498]
.text:0000000000760BCE mov     r11, [rsp+660h+var_518]
.text:0000000000760BD6 xor     eax, eax
.text:0000000000760BD8 lea     rbx, aBotsScreenshot ; "./bots/screenshot/"
.text:0000000000760BDF mov     ecx, 12h
.text:0000000000760BE4 call    runtime_concatstring5
.text:0000000000760BE9 mov     rcx, [rsp+660h+var_470]
.text:0000000000760BF1 mov     rdi, [rsp+660h+var_4E0]
.text:0000000000760BF9 mov     rsi, [rsp+660h+var_4C0]
.text:0000000000760C01 mov     r8d, 1B4h
.text:0000000000760C07 call    os_WriteFile
.text:0000000000760C0C lea     rdx, unk_7F6D00
```

The other screenshots are stored in a similar way but the name is different.

It updates the stolen victim data as well, and the last response from each infected host is stored in the previously created map.



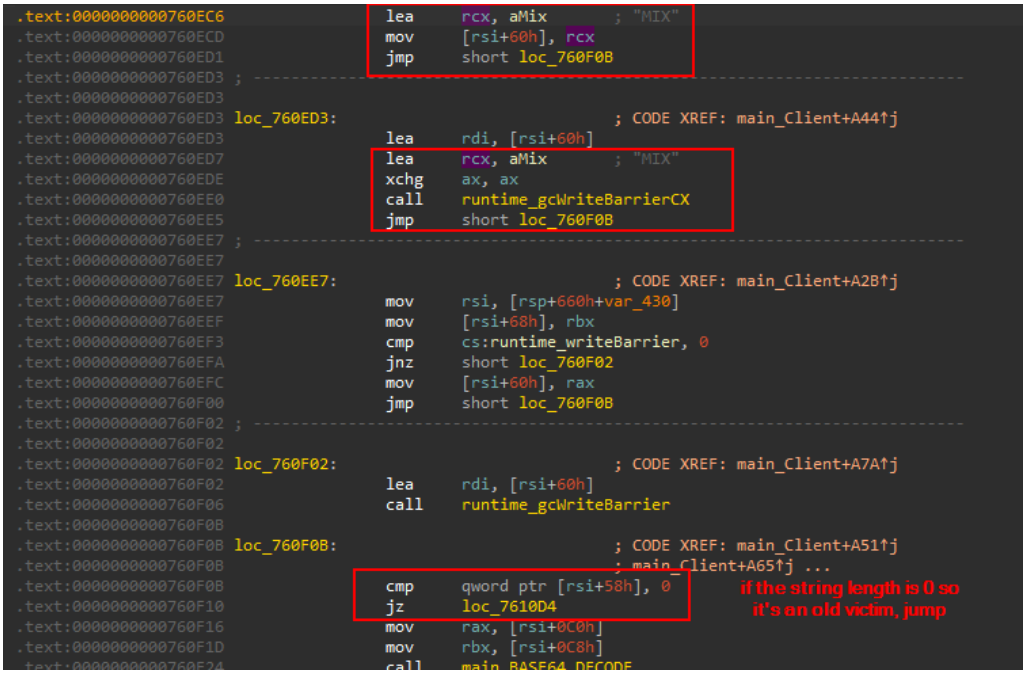
`main_GetGeo` is then called. If we remember, the loaded JSON string was referenced in this function.



It parses the string IP to convert to IP to a Go IP type which is a decimal dotted IP address.

Then it goes through a very large loaded JSON string that contains every IP range associated to each region all over the world.

The new victims will have an identifier is the string `MIX` that is checked to handle the new victims



If the victim is new, it will store the screenshot with `_ACTUAL` tag as discussed before but there is no old one to delete.

At the very end of the function, a call to `main_Registration` is made. This function just adds a new entry to the victims' list and gets the geolocation of the victim.

Main web server

At the beginning of the function `main_Server` there was a goroutine that I missed initially. It calls `main_web` before the call to `net.Listen` .

`main_web` initializes the web interface of the builder and the dashboard with all of its functionality. the server starts at port `8181` .

The function follows the same pattern to set the methods of the handler for APIs:

The following table contains all available APIs with their associated handlers:

API	APIHandler name	APIHandler address	Description
getbots	main_web_func1	0x7635A0	List all the victims by walking through main_BOT_CONN map
callback	main_web_func2	0x763800	get the callback message of each victim through the main_BOT_LASTMESSAGE or Queriyng the raw query of the connection address and get the message associated with victim IP

API	APIHandler name	APIHandler address	Description
callback_STR	main_web_func3	0x763A00	get the callback message string for each victim stored at main_BOT_LASTMESSAGE_STRING
callback_ps	main_web_func4	0x763C00	get the PowerShell response of each victim through main_BOT_POWERSHELL_MESSAGE or Queriyng the raw query of the connection address and get the PowerShell message.
Statistic	main_web_func5	0x763E00	shows statistics about the victims stored in .Aurora file in ./bots/ folder and redirects to web/statistic.html html template. The statistics show all the users with their IP addresses and geolocation
send_pw	main_web_func6	0x764428	sends a base64 encoded PowerShell command to the victim using the json format. The associated key in the query is argument string
GiveMeBuild	main_web_func7	0x7648E0	checks\builds the executable file of the stealer .The build file is stored in .\build it first checks if it exists on the system. if exists, tries to read it. If read is not successfully done, it exits. If not, the author prepared the file to be sent as an attachment for another remote system. it's sent in the Content-deposition as follows: Content-Desposition: attachment = .exe
send	main_web_func8	0x764E60	sends cmd \ PowerShell commands to the victims. They are sent through the argument key in the URL raw query
sftp_stop_reverse	main_web_func9	0x7655A0	closes the SFTP connection with the victims and closes the associated port forwarding functionality. Also, it deletes the entry associated with the deleted victim's SFTP connection in main_BOT_CLIENT_SFTP map
sftp_reverse	main_web_func10	0x765820	start a SFTP server with the victim. the connection is done through port 7273 . The successful connection is indicated by WORK string. the configuration and data about the connection in the associated maps main_BOT_CLIENT_SFTP , main_BOT_LASTMESSAGE . This reverse shell is then used to host the stealer. The infected Bots can be used in DoS attacks too.
screenshot	main_web_func11	0x766540	Takes a screenshot of the victim, it first checks if it's active. SHA1 hash is calculated to the png file to see if the screenshot is the same as the stored or not before updating the database of the victims. the process is identified by Bad or Good statement.
bot	main_web_func12	0x766C00	displays the status of the bots and all information , online boots its geo location, SFTP connected bots in the web/bot.html html template page. it also reads the content of ./core/scr_n_f.png but I don't see any use of it. It encodes the data in it and then redirect to bot.html
logout	main_web_func13	0x767680	Logs out!
auth	main_web_func14	0x767780	Authenticate the access of the client. It uses the file ./cache/Auth.Aurora to compare its content with the newly calculated hashes as discussed before.

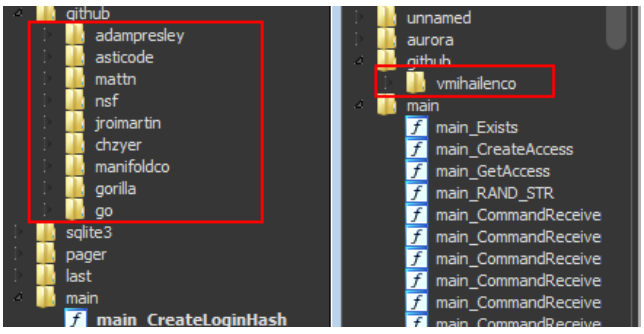
API	APIHandler name	APIHandler address	Description
dashboard	main_web_func15	0x767BA0	The dashboard of the stealer, which shows some data about the active and offline Bots.
del_cmd	main_web_func16	0x768220	deletes a registered command from the main_CMD_QUEUE assigned to the victim
commands	main_web_func17	0x768380	display the command selection interface in the web/commands.html html template
AddCommand	main_web_func18	0x768840	add a new command to the victim commands list, it reads the assigned commands JSON data and adds a new command to it buy calling main_AddCommand that updates main_CMD_QUEUE map assigned to the victim.
AddLoaderCommand	main_web_func19	0x768B60	add loader command. reads the response of the Client.Get() method and then the associated JSON data and base64 encode it. There are some strings used in the identification like EXTERNAL_RUN_PE_X64 . the data then stored in the associated map (main_CMD_QUEUE) and the victims DB

- `net.Query` in Go parses the raw query and returns the values.

```
 u, err := url.Parse("https://example.com/?a=1&b=2")
q := u.Query()
// q will have the values associated to a & b
fmt.Println(q.Get("a")) // print 1
fmt.Println(q.Get("b")) // print 2
```

There's another sample provided to me, executable
hash33fc61e81efa609df51277aef261623bb291e2dd5359362d50070f7a441df0ad

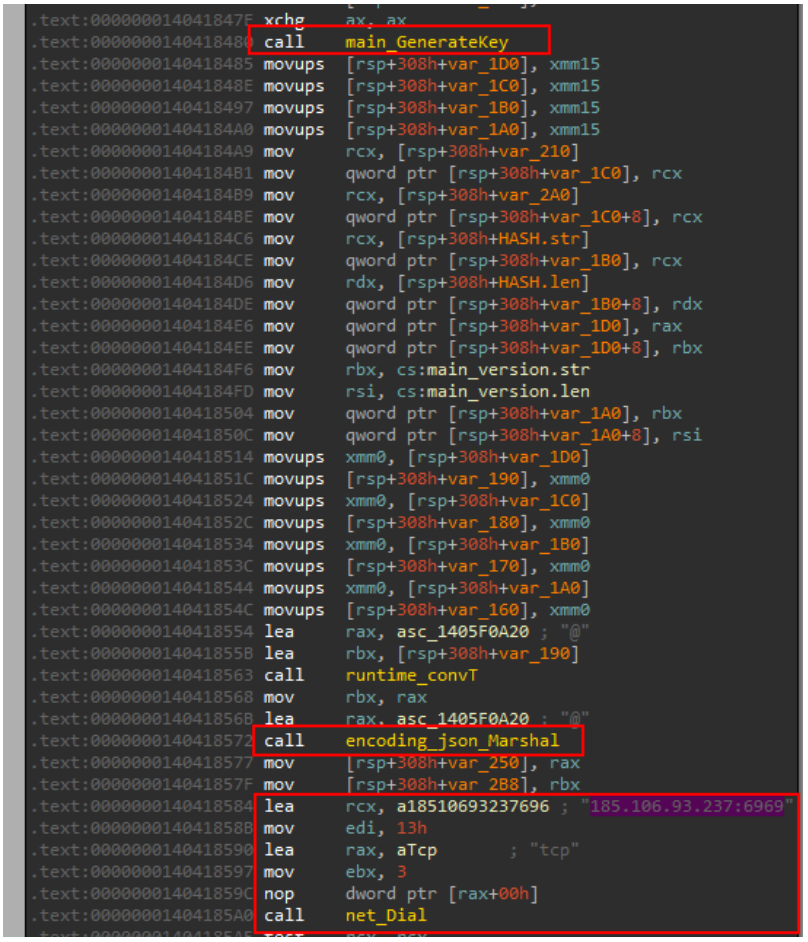
This sample looks like it was one of the first trials of the author to create a stealer in Go. It depends on so many additional legitimate packages from GitHub to create the server and handle the database manipulation and some other things. In the newer builder, it seems like he got more familiar with the Go Language and didn't rely on the packages from GitHub.



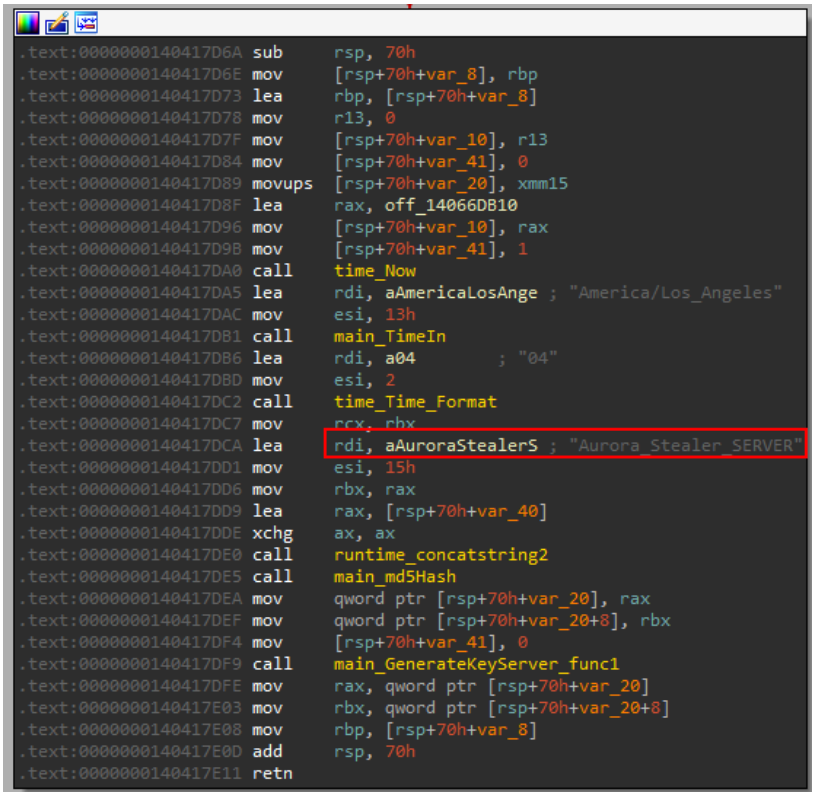
The package used to grab the favicon (from the first GitHub account), create the GUI web application (the second account), provide sqlite3 interface and provide a library like ReadLine in C.

The repositories are in the following table:

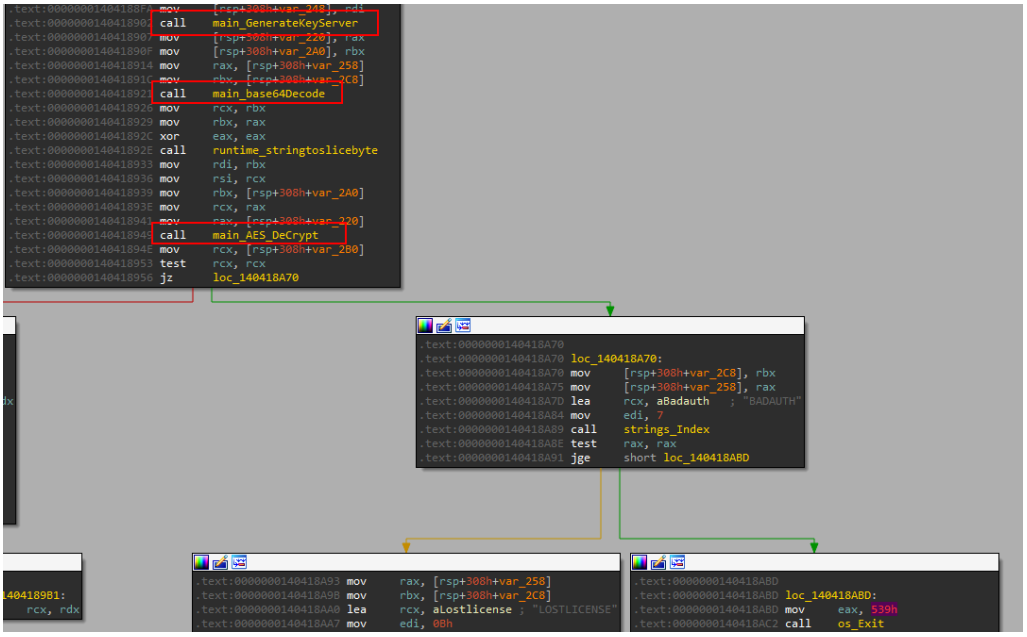
The old sample has some functions that were described before, which were extended in the 2023 version. The hash calculation method and dynamic key but instead of `Aurora_Stealer_2023` it is `Aurora_Stealer_2022` . Then it connects to the remote server to authenticate the user data, to the IP `185.106.93.237:6969` using TCP protocol.



Another dynamic key is used to authenticate with the server, based on the current time too however in the old sample the string `Aurora_Stealer_SERVER` is used.



This key is sent to the remote server and calculated later in the following code to verify the user access and the dynamic key to make sure there is no debugging session started.



If the keys do not match, the function breaks and the program is terminated.

Another dynamic key is calculated but this time for the client, it uses the string `Aurora_Stealer_2033` with the same timing method of calculation discussed.

The hashes are stored then in `ATX.Aurora` in `./cache` folder.

It then checks the existence of some files: `./cache/ATX.Aurora` , `./cache/telegram.Aurora` , `./cache/Config.Aurora` and `./cache/Trash` .

`./cache/Trash` contains older Aurora executables, the older executables are auto-moved to this folder using PowerShell command, and the new version, which is expected to be in `.zip` format with the name `Update.zip` , is then unzipped and replaces the older version. The program is then restarted using PowerShell. This is all done in `main_AutoUpdate` function.

The function `main_ReadTGData` reads telegram data from the file `./cache/telegram.Aurora` which is AES encrypted. The authentication is done using a telegram bot through the telegram API. This authentication method is removed from the new version, where everything is done through communicating with the remote server.

The old builder additionally contains an important function called `main_LoadStealer` . This function calls two other goroutines. both two functions execute PowerShell commands that configure the firewall to allow it to receive incoming TCP connections through Port 80 and 8081.


```
 #function main_LoadStealer_func2 allow it on local port 80
netsh advfirewall firewall add rule name="Port 80 dir=in action=allow protocol=TCP localport=80
#function main_LoadStealer_func2 allow it on local port 80
netsh advfirewall firewall add rule name="Port 8081 dir=in action=allow protocol=TCP localport=8081
```

At the end of the main function, it creates a new hidden instance of CMD and starts the Web service of the stealer. using the function `main_StartWeb`

This function starts the web service on localhost `http://127.0.0.1/dashboard` . It has a different set of APIs and different associated handlers then the newer version.

- The command strings are highlighted.

API	APIHandler name	APIHandler address	Description
receive	main_StartWeb_func1	0x140421B00	It receives the incoming commands and connects to the remote server 185.106.93.237:6969 to get match the stored hashes with the calculated one in form of *Aurora*<PASSWORD .this function has a lot of other functionality. it reads the command from the response of the server. It allows the user to delete a directory Delete, remove file grabber RemoveG, or remove the loader RemoveL.GEO_URL to get the geolocation of all victims. AddDmen Add a new domain name received from the server.BuildGen builds a new version of the stealer and the ability to increase the file size PumbMB.DeleteTG , AddTelegram delete\add telegram configuration.DeleteAll Delete all the configs.ChangePassword , change password and download all logs files Download_AllLogs. Download_OnlyCrypto downloads the crypto wallet information only.
api.exe	main_StartWeb_func2	0x140421B60	adds a new telegram API key to the stealer and adds an icon using resource hacker cmd command ./resource/ResourceHacker.exe -open ./builds/<STEALER_NAME>.exe -save ./builds/<STEALER_NAME>.exe -action addskip -res ./resource/main.ico -mask ICONGROUP,MAIN .
dashboard/{id: [0-9]+}	main_productsHandler	0x14041D080	display the main window of the web service displays information about a specific victim ID: Cookies, passwords, the Geolocation, and crypto wallet information. Logs are stored in ./logs/ folder contain passwords in passwords.txt , cookies in folder Cookies . All the information is shown through the HTML template ./gui/Dashboard.html
download_geo	main_StartWeb_func3	0x140422100	retrieves the geolocation information, the same as the new one.
download_l	main_StartWeb_func4	0x1404222A0	gets the logs in a .zip archive, uncompresses it and deletes the archive. the logs contain all the stolen data
api/get-log-build	main_StartWeb_func5	0x140422620	get the build logs from ./logs associated with a specific API key used
build.exe	main_StartWeb_func6	0x140422B60	gets a build executable of the stealer stored at ./builds
dashboard	main_StartWeb_func7	0x140422EA0	display the dashboard of the stealer, and shows some statistics about the infected system. IPs,

API	APIHandler name	APIHandler address	Description
			geo-location and the stolen information
loader	main_StartWeb_func8	0x140422FE0	display information about the Loader and file grabber. the threat actor can use this section to configure the loader and specify the target file to grab. file ./config/telegram.txt is used to extract the telegram connection configuration. The information is viewed by executing gui/Loader.html HTML template.
setting	main_StartWeb_func9	0x1404234A0	builder settings, display information about the subscribed plan and change the password and telegram configuration and API. and shows the used domains
auth	main_StartWeb_func10	0000000140423A40	the AUTH page that the user signs in to where the used credentials and AUTH cache file in ./cache/AuthHash.Aurora are checked. Whenever the user navigates, the credentials and hashes are checked. if not valid, will be redirected to this page
builder	main_StartWeb_func11	0x140423CC0	creates a new build through it. the build target architecture victims group is chosen.
checker	main_StartWeb_func12	0x140424380	checks the wanted information from the victim DB. check the build used and get the geolocation of the victim specified.

then the server is started on port 80

In function `main_AddNewClient` , the victim entries on the data based are created by calling `main_CreateDB` data stored about the user in `UserInformation.txt` :

- HWID
- Build ID
- Log date
- IP
- Country
- Region
- City
- PC INFORMATION
  - CPU
  - Screen Size
  - Screen Size
  - RAM
  - Display Device (GPU)

in addition to the stolen information the following credentials are received:

- Steam
- Passwords
- cookies
- crypto wallets -stored in subdirectory `/wallets`
- Telegram info
- screenshots
- grabbed files -stored in subdirectory `./FileGrabber`
- Cards information

Browser cookies are stored in `.db` files in `./cache` to be decrypted and the extracted data is stored in `.txt` file.

The end of the packet is checked by `END_PACKET_ALL_SEND` sentence. And the last packet sent to the victim is `Thanks` , then, the data are zipped and sent to the telegram account configured.

The function `main_DecryptLog_Card` is used to decrypt the credit card information collected. It uses the following sqlite3 query to achieve that:

```
select name_on_card, expiration_month, expiration_year, card_number_encrypted, date_modified, use_date, use_count, nickname from
```

You can find screenshots of the HTML templates in this tweet.

all the rules can be found here.

new builder version

```
rule aurora_stealer_builder_new{
   meta:
   malware = "Aurora stealer Builder new version 2023"
   hash = "ebd1368979b5adb9586ce512b63876985a497e1727ffbd54732cd42eef992b81"
   reference = "https://d01a.github.io/"
   Author = "d01a"
   description = "detect Aurora stealer Builder new version 2023"

   strings:
   $is_go = "Go build" ascii

   $s1 = "_Aurora_2023_Technology_"    ascii
   $s2 = "AURORA_TECHNOLOGY"  ascii
   $s3 = "scr_n_f.png" ascii
   $s4 = "EXTERNAL_RUN_PE_X64" ascii
   $s5 = "[Aurora]" ascii //log messages begin with [Aurora] __LOGMSG__

   $fun1 = "main.Server" ascii
   $fun2 = "main.GetAcess" ascii
   $fun3 = "main.AddCommand" ascii
   $fun4 = "main.GetGeoList" ascii
   $fun5 = "main.GiveMeBuild" ascii

   condition:
   uint16(0) == 0x5a4d and ( $is_go and (2 of ($s*)) and (2 of ($fun*)) )
}
```

old builder version

```
rule aurora_stealer_builder_old{
   meta:
   malware = "Aurora stealer Builder old version 2022"
   hash1 = "33fc61e81efa609df51277aef261623bb291e2dd5359362d50070f7a441df0ad"
   reference = "https://d01a.github.io/"
   Author = "d01a"
   description = "detect Aurora stealer Builder old version 2022"

   strings:
   $is_go = "Go build" ascii

   $s1 = "ATX.Aurora"    ascii
   $s2 = "Aurora_Stealer_2033"  ascii
   $s3 = "Aurora_Stealer_SERVER" ascii
   $s4 = "[Aurora Stealer]" //log messages
```

```
16
17          $fun1 = "main.DecryptLog" ascii
18          $fun2 = "main.CreateDB" ascii
19          $fun3 = "main.GenerateKey" ascii
20          $fun4 = "main.TGParce" ascii
21
22          condition:
23          uint16(0) == 0x5a4d and ( $is_go and (2 of ($s*)) and (2 of ($fun*)) )
24      }
```

ebd1368979b5adb9586ce512b63876985a497e1727ffbd54732cd42eef992b81	**aurora.exe (2023 version)**
e7aa0529d4412a8cee5c20c4b7c817337fabb1598b44efbf639f4a7dac4292ad	builder archive (2023 version)
33fc61e81efa609df51277aef261623bb291e2dd5359362d50070f7a441df0ad	aurora.exe (2022 version)
33b61eb5f84cb65f1744bd08d09ac2535fe5f9b087eef37826612b5016e21990	geo.Aurora
1def6bdec3073990955e917f1da2339f1c18095d31cc12452b40da0bd8afd431	ds.html
f1ba92ae32fcaeea8148298f4869aef9bcd4e85781586b69c83a830b213d3d3c	statistic.html
8b1abbb51594b6f1d4e4681204ed97371bd3d60f093e38b80b8035058116ef1d	bot.html
e9cf3e7d2826fa488e7803d0d19240a23f93a7f007d66377beb1849c5d51c0af	commands.html
d7829f17583b91fb1e8326e1c80c07fc29e0608f1ba836738d2c86df336ea771	rergister.html
1b88624936d149ecdea6af9147ff8b2d8423125db511bdf1296401033c08b532	settings.html
185.106.93.237:56763	Aurora server -version 2023- used in user account verification
185.106.93.237:6969	Aurora server -version 2022- used in user account verification
Auth.aurora	locally created for each Aurora panel user and used in account verification
scr_n_f.png	contains config information
screenshot/	a local folder that contains victims' screenshots
<*>_ACTUAL.png	screenshot of current state of online bots
<>_<>.png	custom screenshots format

The following go files were identified in the binary, all starting with the path: "C:/Users/SixSixSix/Desktop/Botnet 2023/26.01.2023/new/"

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
```

```
 auth.go
crypt.go
command.go
compressor.go
core.go
geo.go
main.go
pfor.go
port.go
web.go

core/statistics/window.go
core/statistics/winfuns.go
core/statistics/queue.go
```

```
core/monitor/monitor.go
core/common/copy.go
core/common/udpconn.go
core/common/util.go
core/logger/logger.go
core/schema/monitor.go
core/schema/util.go
core/server/client.go
core/server/client_handlers.go
core/server/server.go
core/server/server_handlers.go
```

There are similar files identified in the old version of the builder/panel.

The common path for this older sample is: “C:/Users/SixSixSix/Desktop/Aurora 2022/server”

```
 auth.go
compressor.go
config.go
cryptography.go
favicon.go
geo.go
gui.go
main.go
notify.go
other.go
server.go
telegram.go
zip.go
```

To create the Yara rules, the following strings were used. Those are all present in the builder:

```
 127.0.0.1:7273

POWR

WORK

PORT_FORWARD

FTP_RUN - REVESRE START

_*Aurora_2023_Technology_*

AURORA_TECHNOLOGY

./cache/Auth.aurora

_ACTUAL

./bots/screenshot/

./core/scr_n_f.png

EXTERNAL_RUN_PE_X64

[Aurora] Botnet - SERVER - RUN

- old sample.
```

```
./cache/Config.Aurora

./cache/Aurora.Aurora

./cache/telegram.Aurora

./cache/ATX.Aurora

Aurora_Stealer_2033

Aurora_Stealer_SERVER

Aurora_Stealer_2022

https://api.telegram.org/bot%s/%s

./cache/AuthHash.Aurora

[Aurora Stealer]: Yes i am work!
```

@gi7w0rm for providing me with the samples and helping me formatting the article to make it better.

Source: https://d01a.github.io/aurora-stealer-builder/