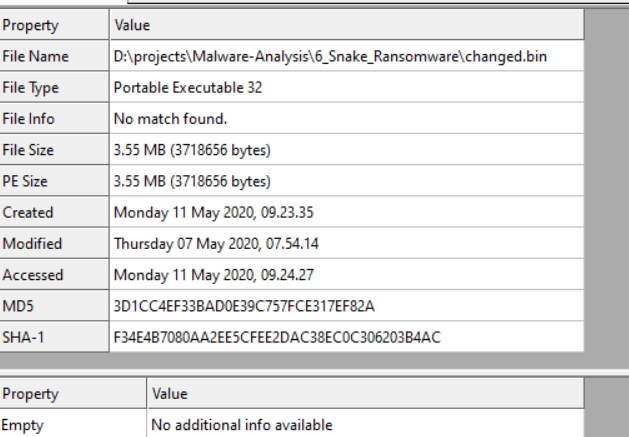
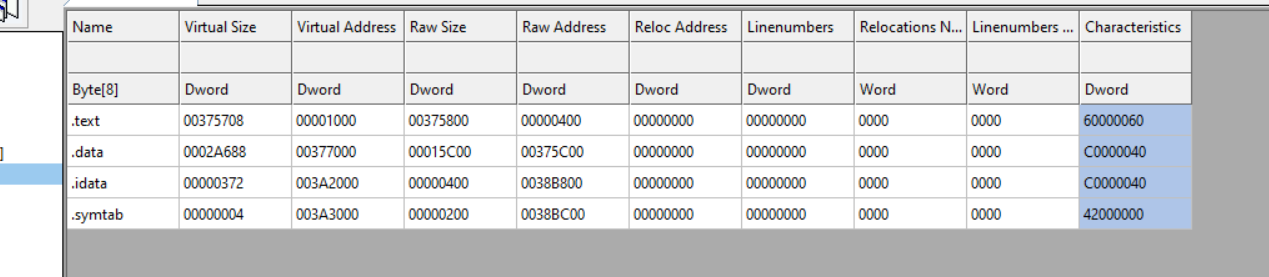
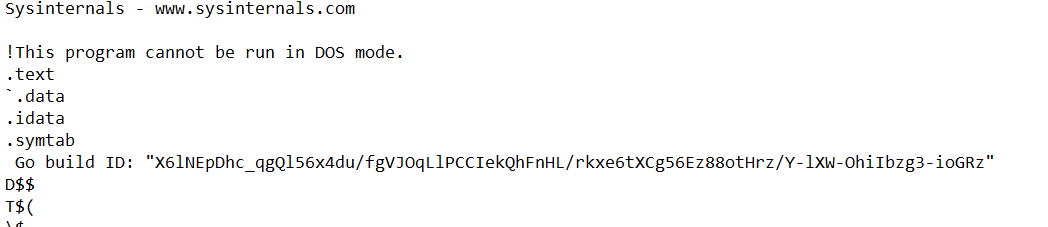
Basic info





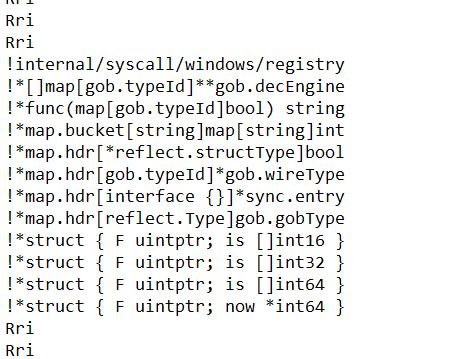
Symtab section, go binary has those



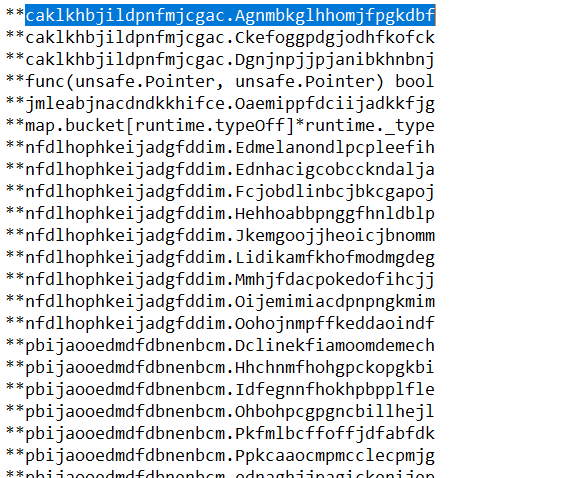
Strings shows its go binary



I guess golang function names



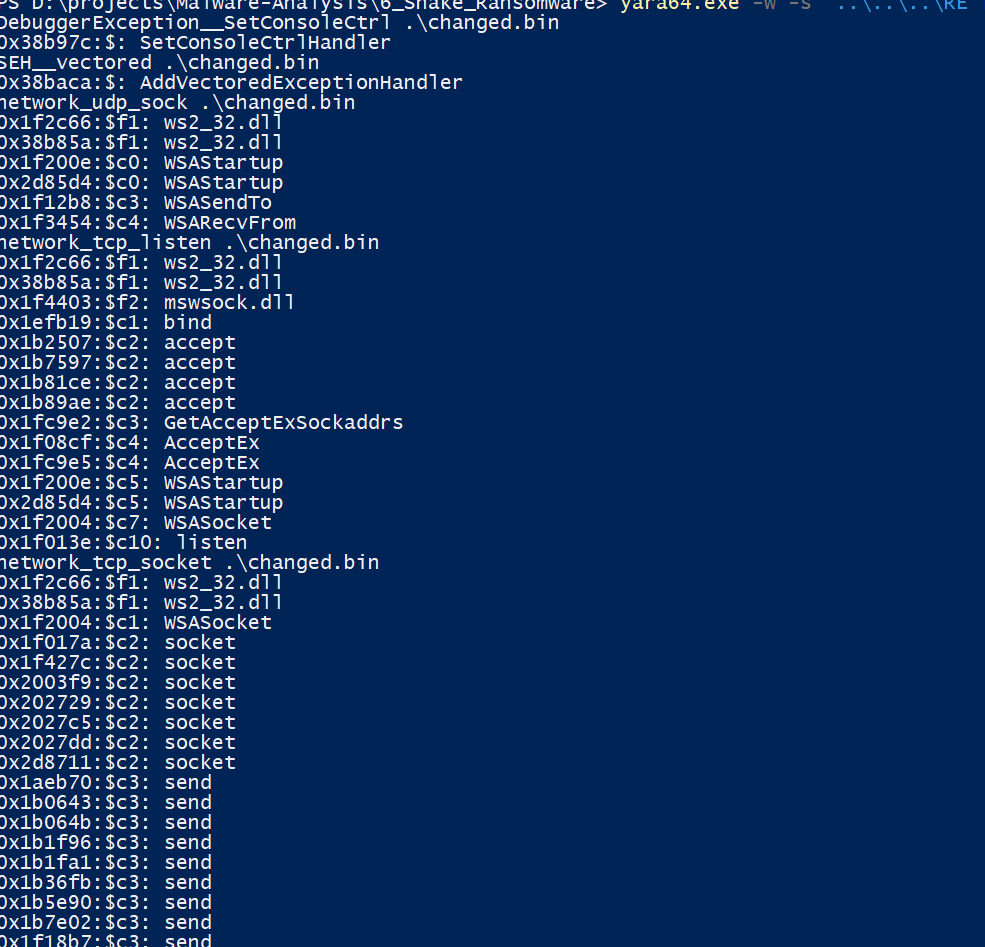
Some more

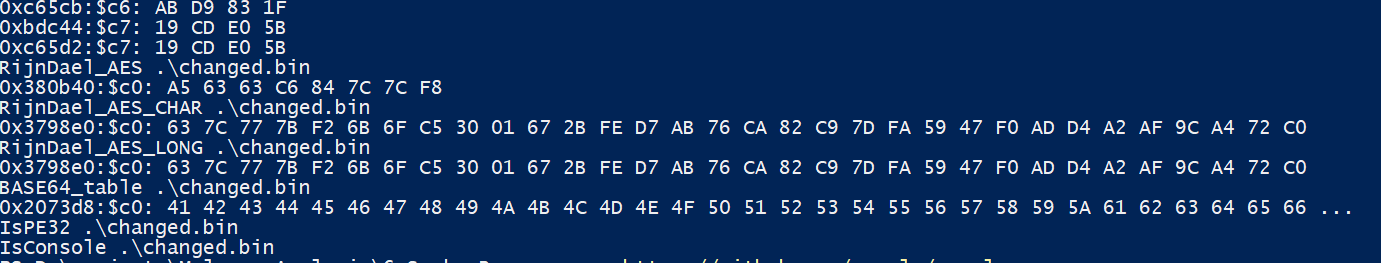


Strange stuff as well



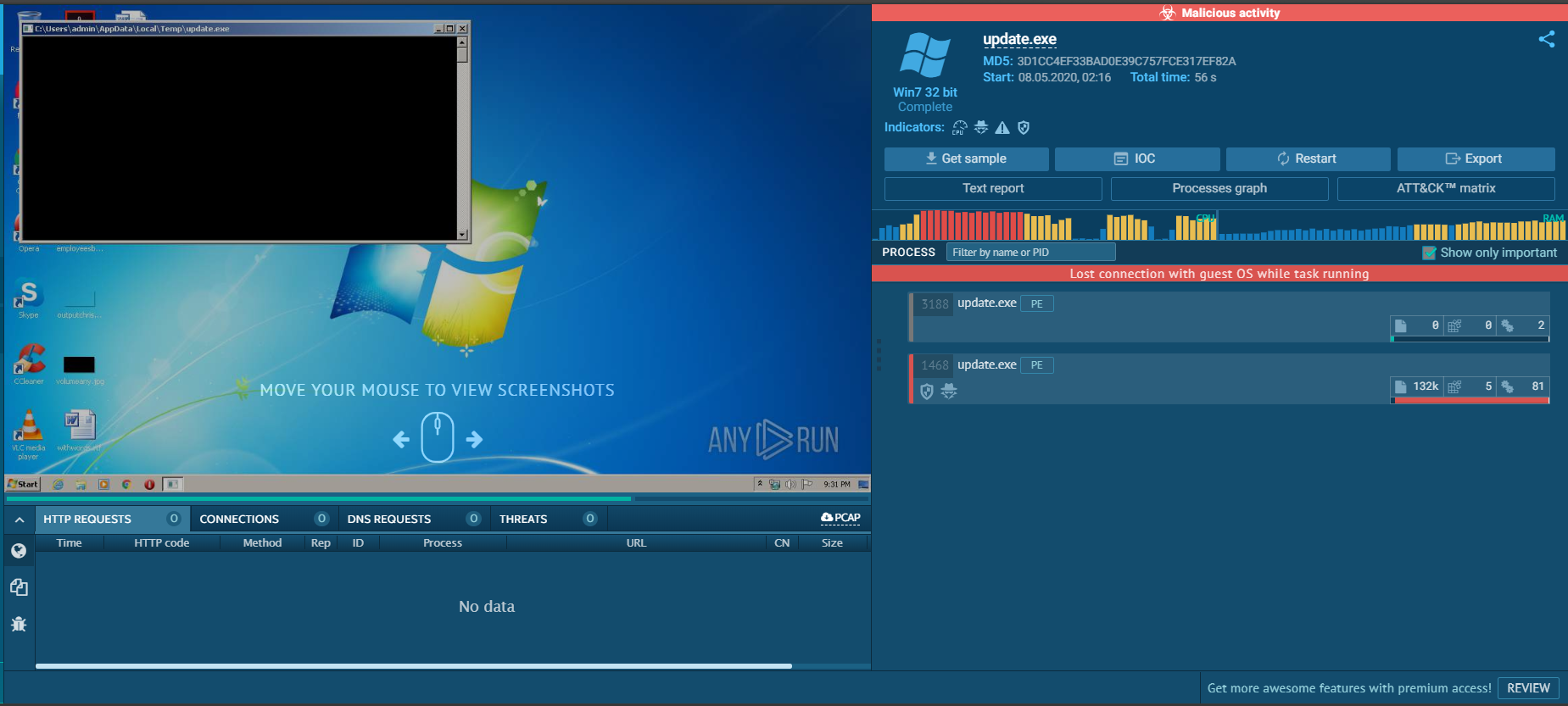
Go compiler adds metadata about the directory during coding. WIN1 was the username. Also the malware is using opensource projects like go-ole <https://github.com/go-ole/go-ole> Running yara

 Debugger exception

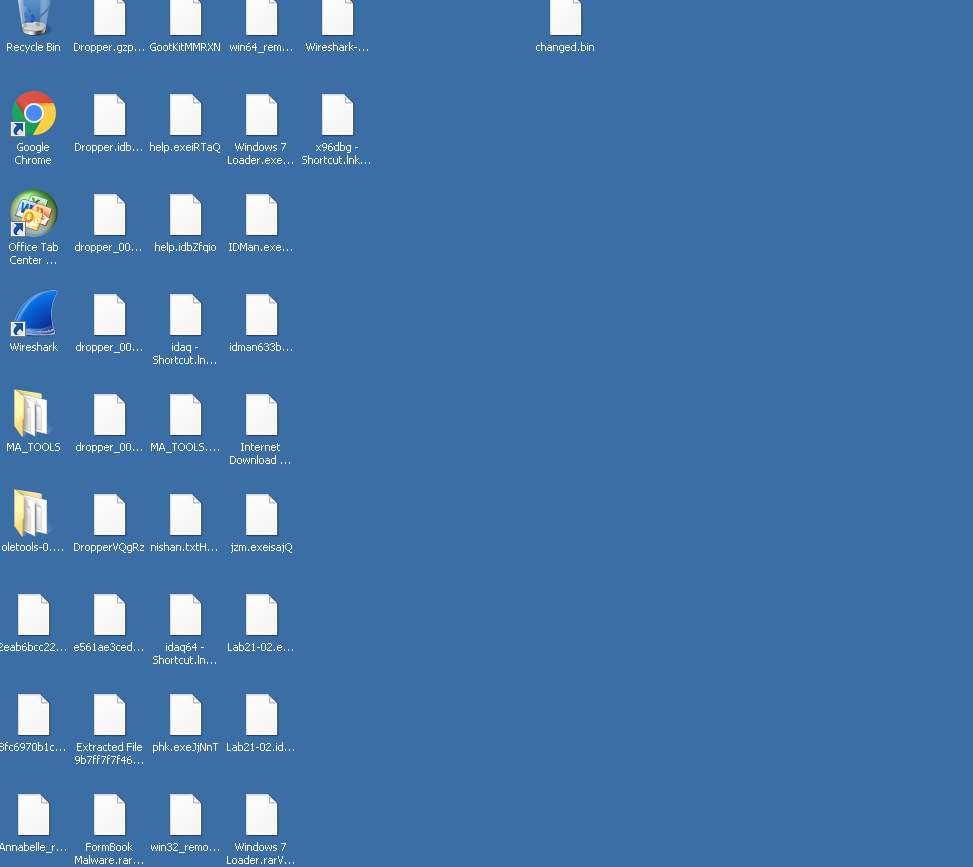


AES stuffs

Running in Sandbox, Just runs a copy of itself



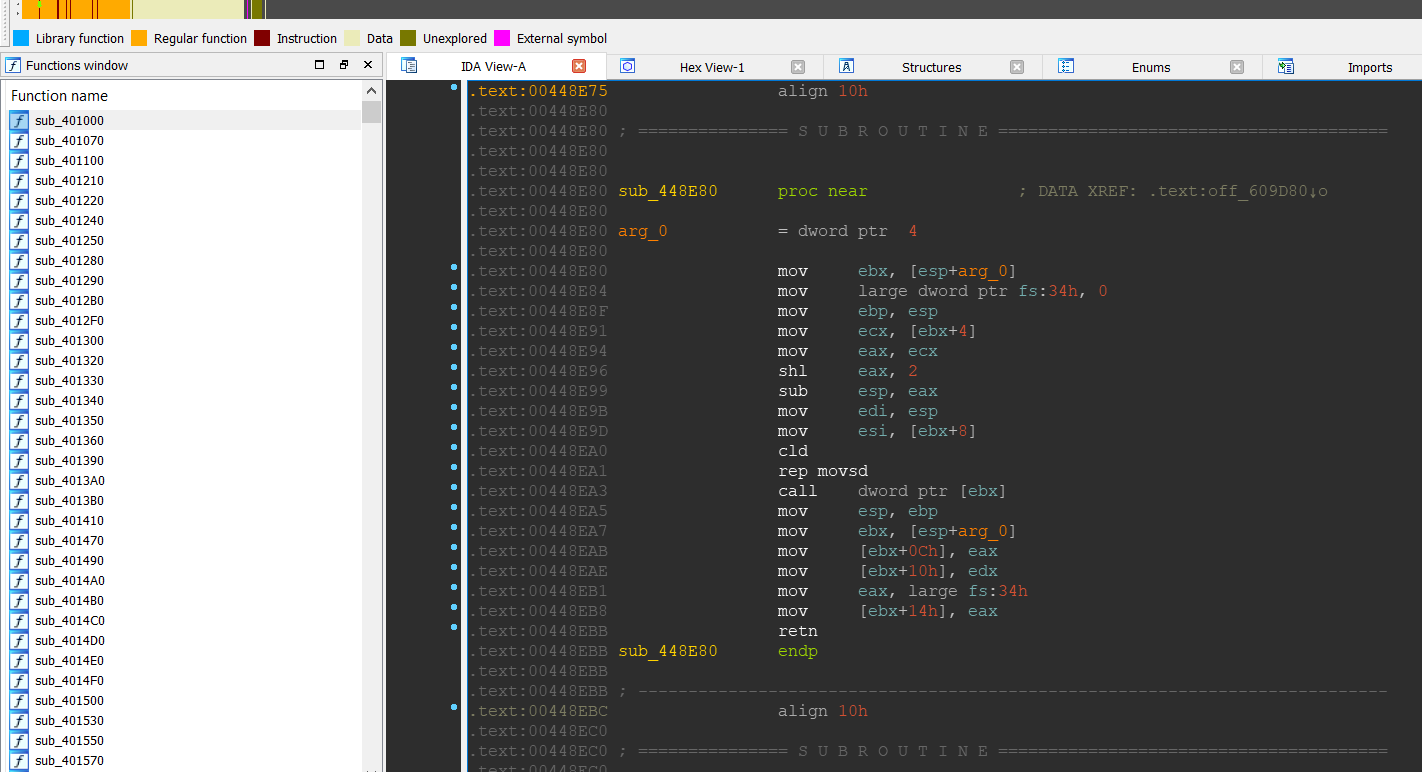
Running in VM



But no advice file

# Code Reversing

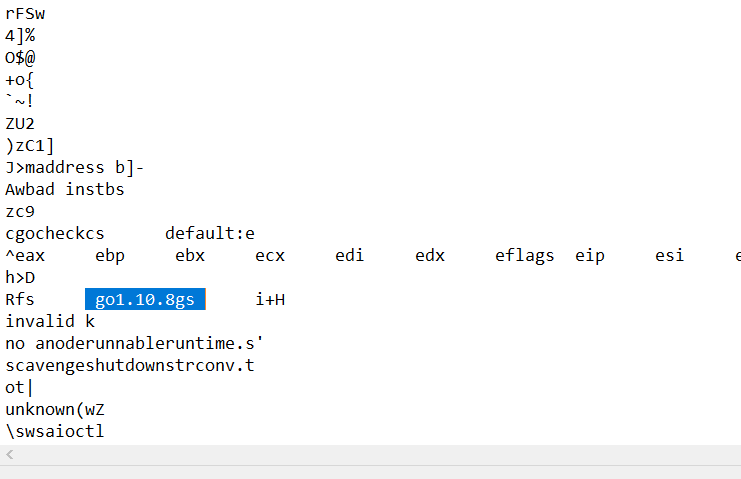
Unlike regular golang binaries, there were no regular function names,

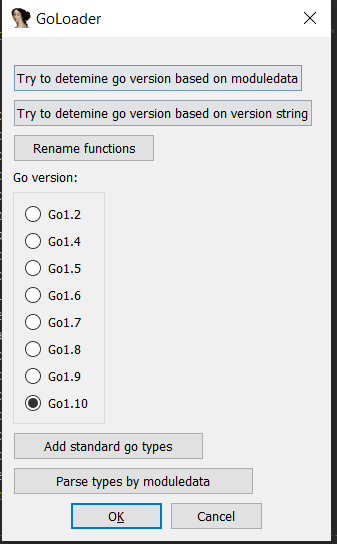


so tool used idagolang helper

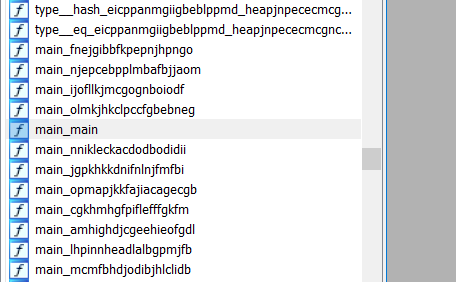
<https://github.com/sibears/IDAGolangHelper>

used strings to find the golang version

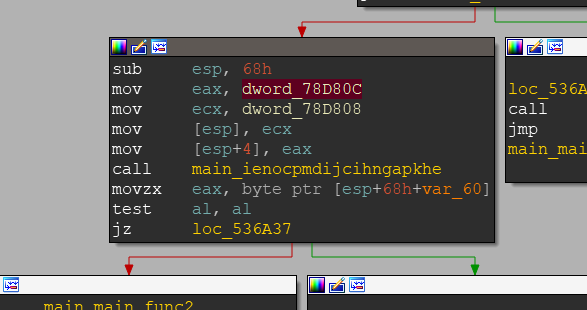




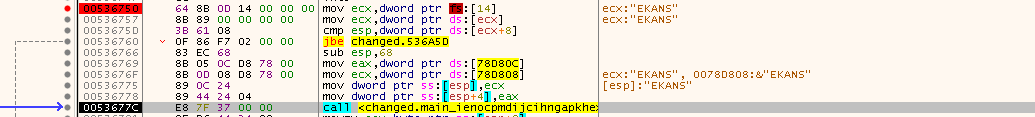
Now we getting somewhere, but looking at function I was trying to get a hint, but the function names itself is weird.



Lets analyze main

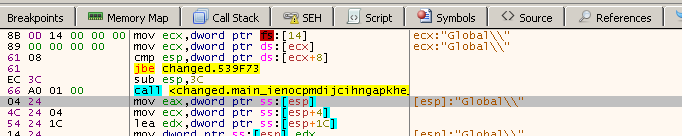


Dword values are not during static analysis, so I analyze dynamically



Debugging main\_ienocpmdijcihngapkhe

**main\_ienocpmdijcihngapkhe\_func1** takes alphabets, and returned [esp]:"Global\\"



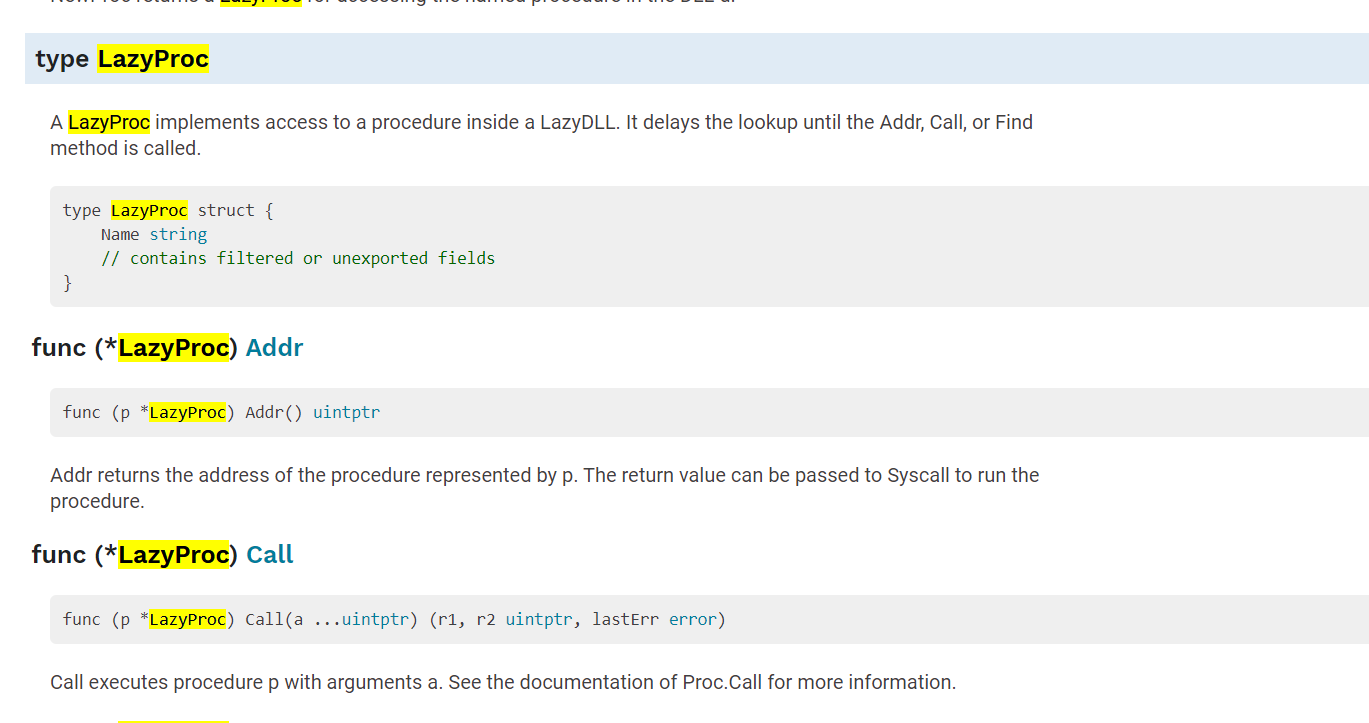
Renamed to main\_ienocpmdijcihngapkhe\_func1\_takes\_alphabet\_first\_Returned\_global



Lets see the GOs documentation

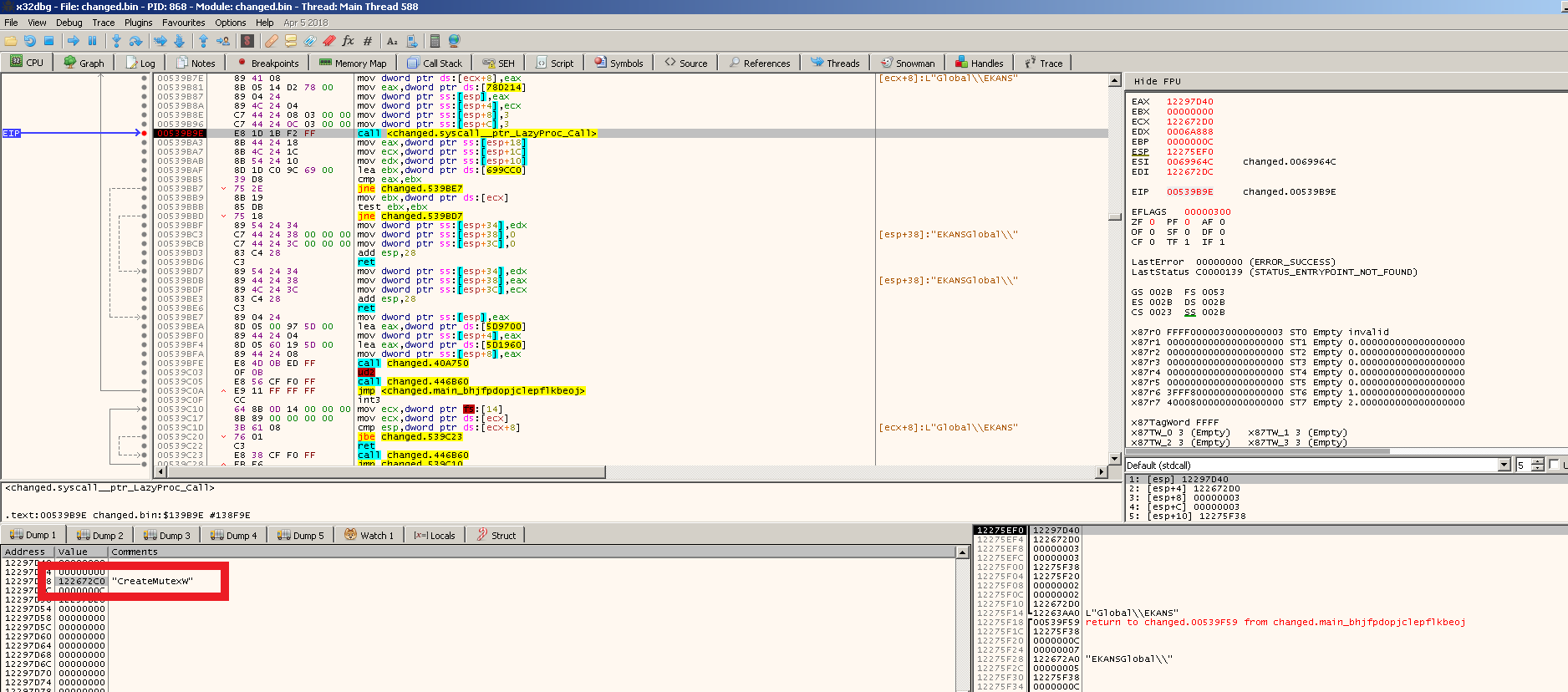
<https://golang.org/pkg/syscall/?GOOS=windows>

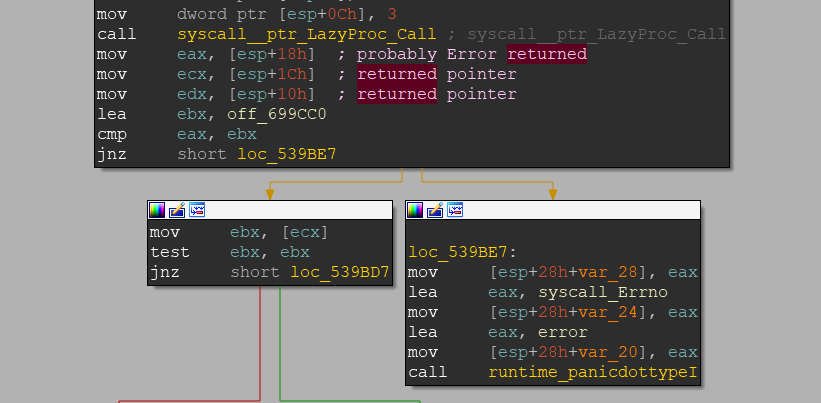
So the argument passed dword\_78D214 is unrecognized during static analysis. During dynamic analysis. Set a breakpoing, if it runs



Guess calling a function from dll

In golang the arguments are passed via stack, since we know that a pointer to function is being passed, and a Name must be the name of the function being called, when dynamically executed. We see CreateMutex being passed

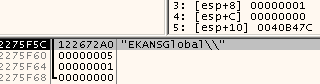


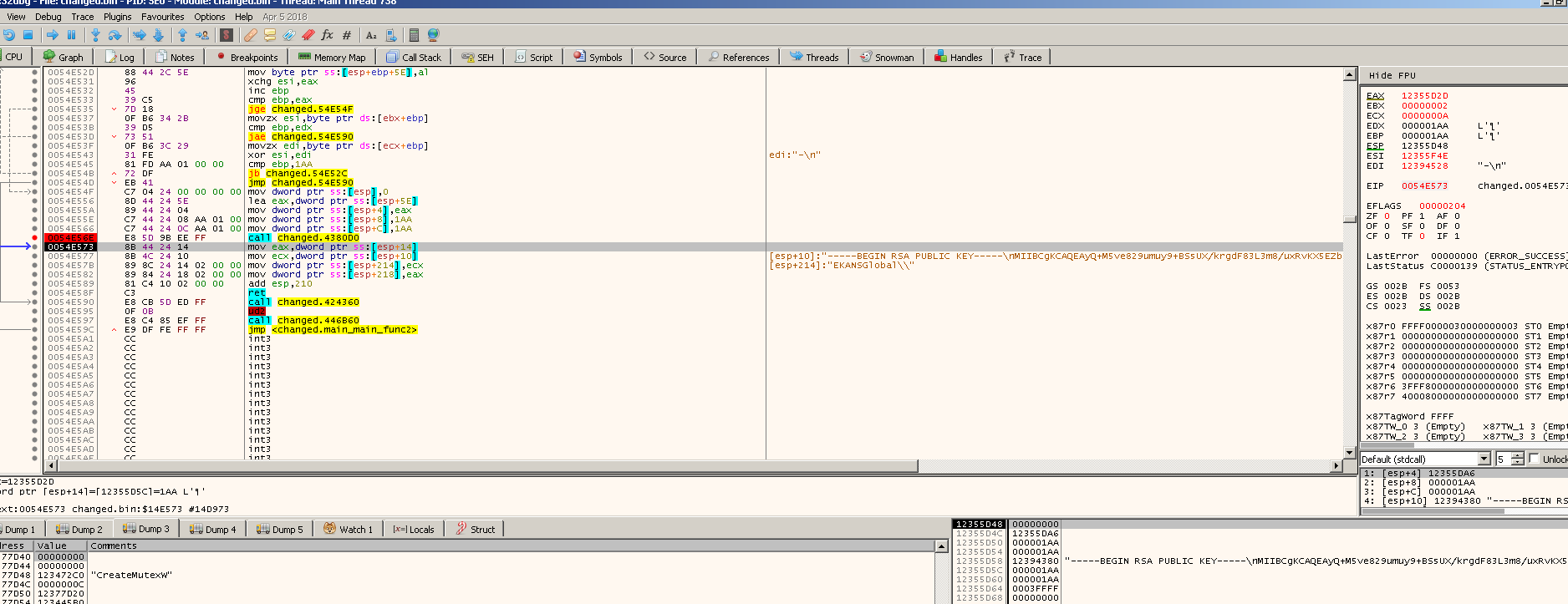


By observing that the value esp+0x18 being used in error condition check we can assume that it is the lastErr value and the rest is the returned pointer. Seeing this we can safely assume, that the function main\_bhjfpdopjclepflkbeoj is responsible for calling the windows function CreateMutexW and returning its value. Renaming the function to sycall\_function

Renamed main\_ienocpmdijcihngapkhe to calls\_function\_Creating\_global\_and\_sycall\_create\_mutex.

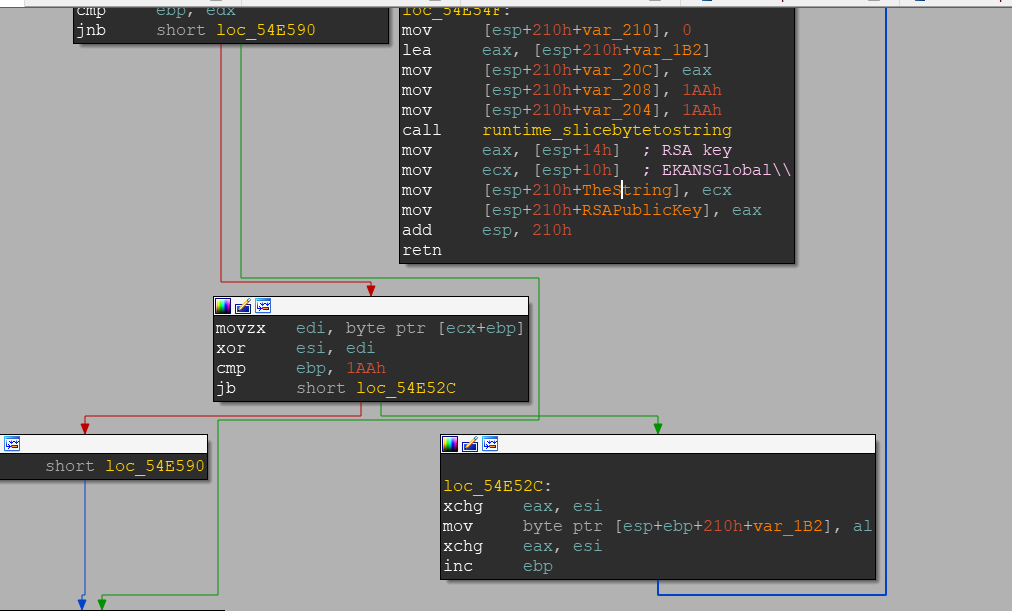
Analyzing the function main\_main\_func2. Observing the arguments being passed.





Function at 0054E480 was responsible for key extraction

"-----BEGIN RSA PUBLIC KEY-----\nMIIBCgKCAQEAyQ+M5ve829umuy9+BSsUX/krgdF83L3m8/uxRvKX5EZbSh1+buON\nZYr5MjfhrdiOGnrbB1j0Fy31U/uzvWcy7VvK/zcsO/5aAhujhHB/qMAVpZ8zT5BB\nujT1Bvsith/BXgtM99MixD8oZ67VDZaRM9TPE89WuAjnaBZORrk48wFcn1DOAAHD\nZ9z9komtqIH1fm3Y0Q6P76nUscLsYOme082L217Th/lTMoqqs4cF2rn9O9Vp4V9U\naCs4XVxGSpcuqbIscfpf0cm44P2eOEk+sbZdahO9C6fezt7YF4OCJ4Vz3qqMD6z4\n+6d7FRxUu6k3Te2T2bWBZnsDO30pYFi/gwIDAQAB\n-----END RSA PUBLIC KEY-----\n"

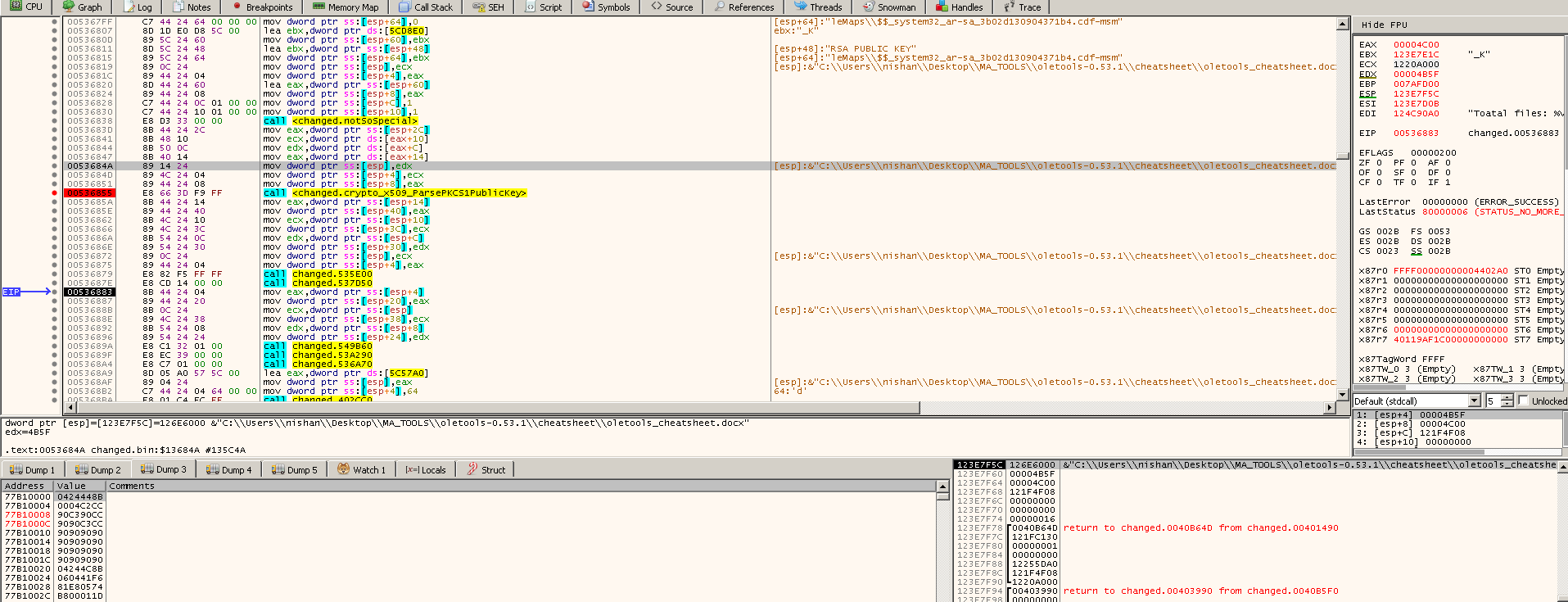


We rename variables accordingly

After indepth analysis it seems

main\_ffhlhdmhalodcojcaeok\_\*\*\*

function just slice to byte and byte to slice back, not useful at all



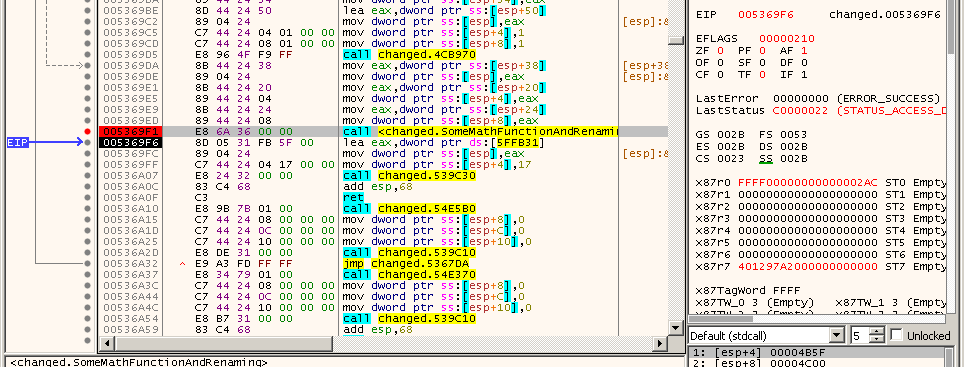
We see location to files being loaded by function.

0053687E | E8 CD 14 00 00 | call changed.537D50 |

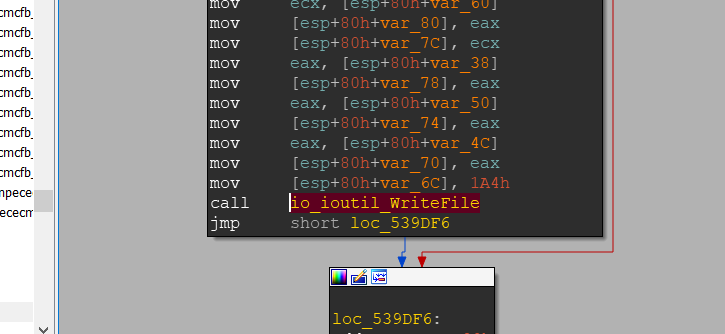
When Statically observing, many recursive function calls

Rather lets find the function that actually encrypts it

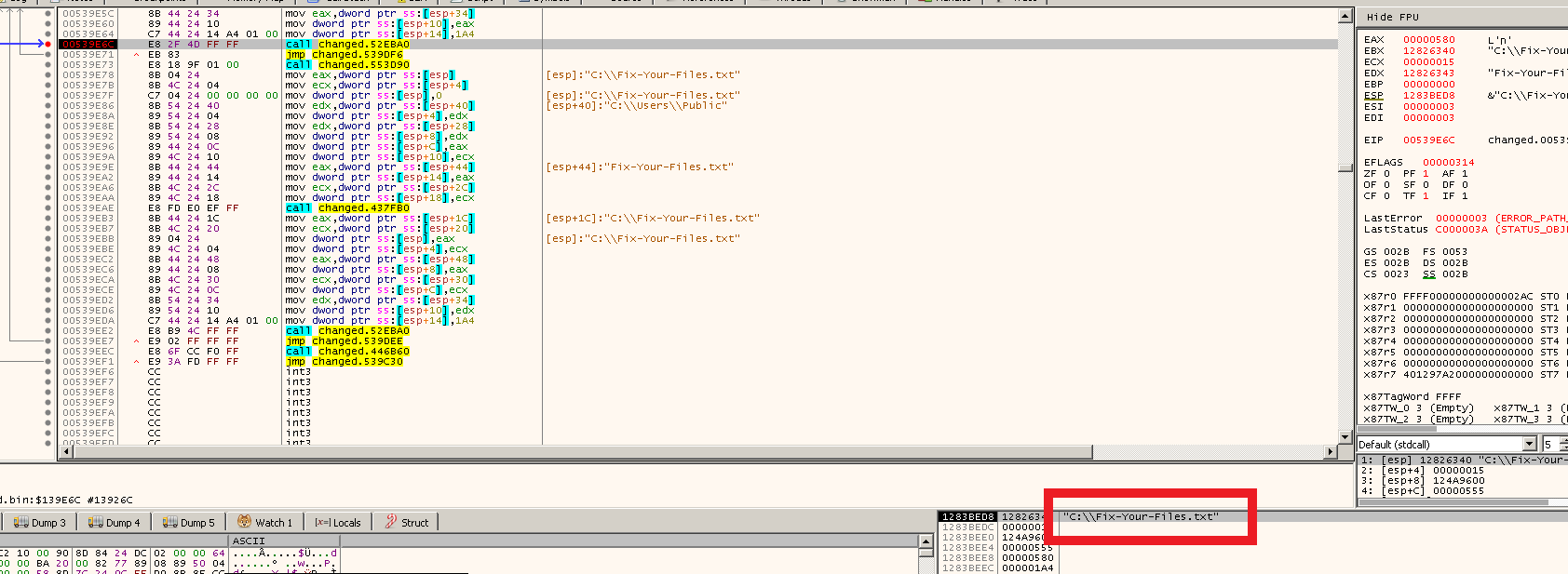
After further digging down, we see that



Function sub\_0053A060 is responsible for looping through each files and encrypting them

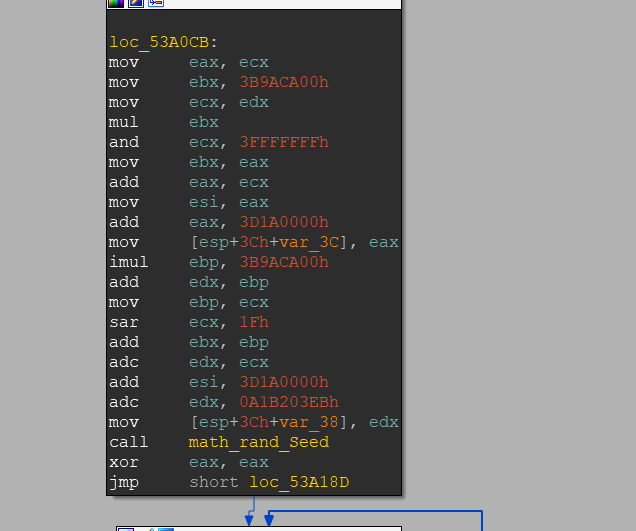
And finally the function main\_amgmhoeegchpnnaljano 00539C30 eventually calls io\_ioutil\_WriteFile, 

Possible help file, so we run dynamically.



The file is at C:\\Fix-Your-Files.txt. but I could not find the file

Lets digger into the function 0053A060, which I renamed loopThroughEverythingAndEncrypt.



We see a call to math\_rand\_seedFor Yara

.symtab

Go build ID

C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/pdmnjmklbkflhigdnplp/oleutil.go