Automatic Gobfuscator Deobfuscation with EKANS Ransomware

gogglehead edhacker.com/blog/post/22

Jacob Pimental March 17, 2021

17 March 2021

A few months ago I saw an article by Netlab 360 describing the malware BlackRota, specifically the obfuscation method used known as gobuscate. I noticed that a deobfuscator was made for this using Binary Ninja's API, so I decided to take a crack at developing a plugin for Cutter. To demonstrate the tool I created, I will also give a brief analysis of another malware sample that uses gobfuscate, Ekans.

How Gobfuscate Works

Package Renaming

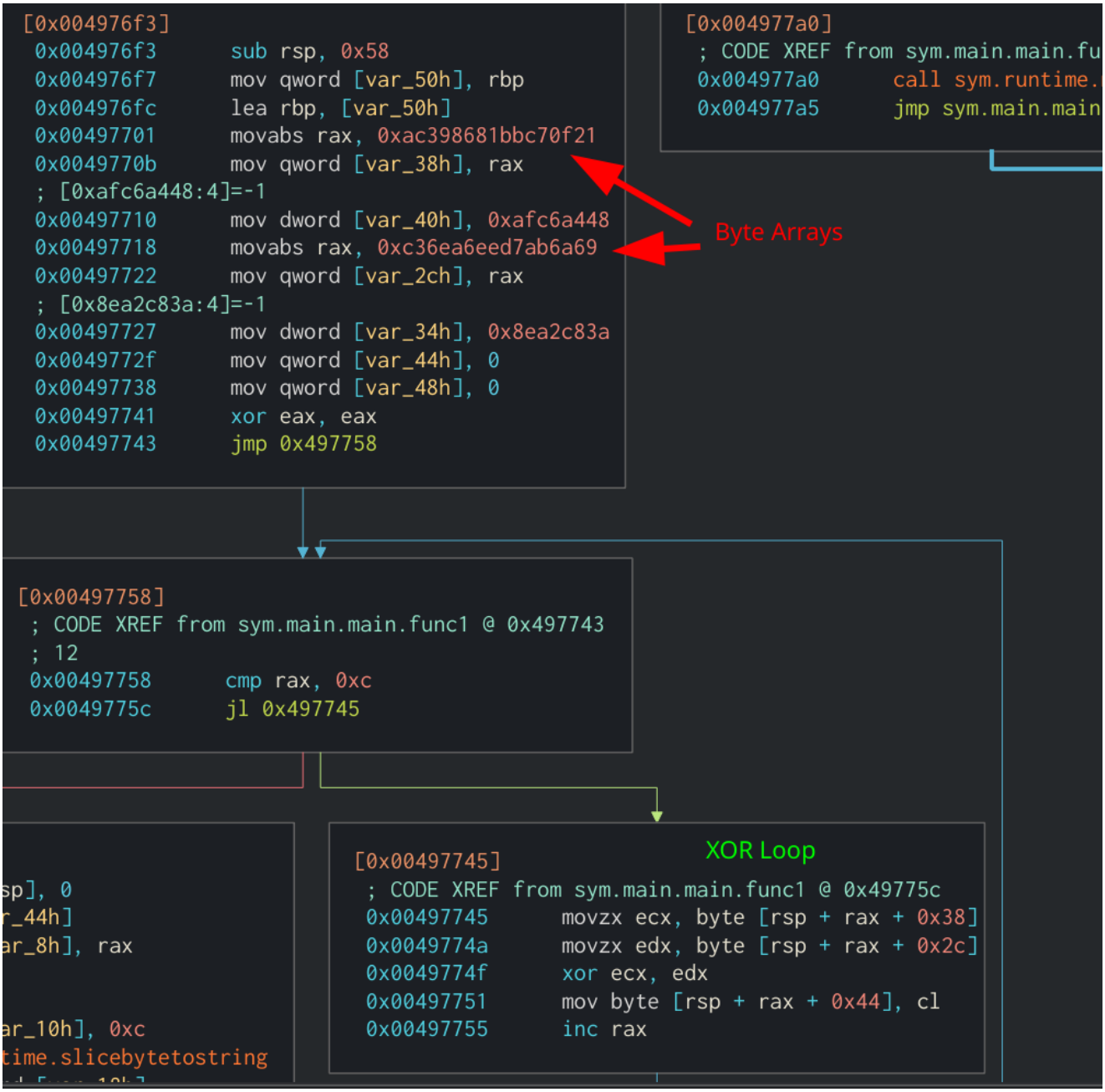
One of the things gobfuscate will do is rename package names to make it harder for analysts to identify them. It does this by taking the package name, hashing it using sha256, and replacing any numbers in the hash with letters using the algorithm:

```
'g' + (x - '0') # x is the current character
```

This means that the package name contains only the characters `a-p` and is irreversible. The example that the gobfuscate GitHub page gives is that the package `github.com/unixpickle/deleteme` becomes `jiikegpkifenppiphdhi/igijfdokiaecdkihheha/jhiofoppieegdaif`.

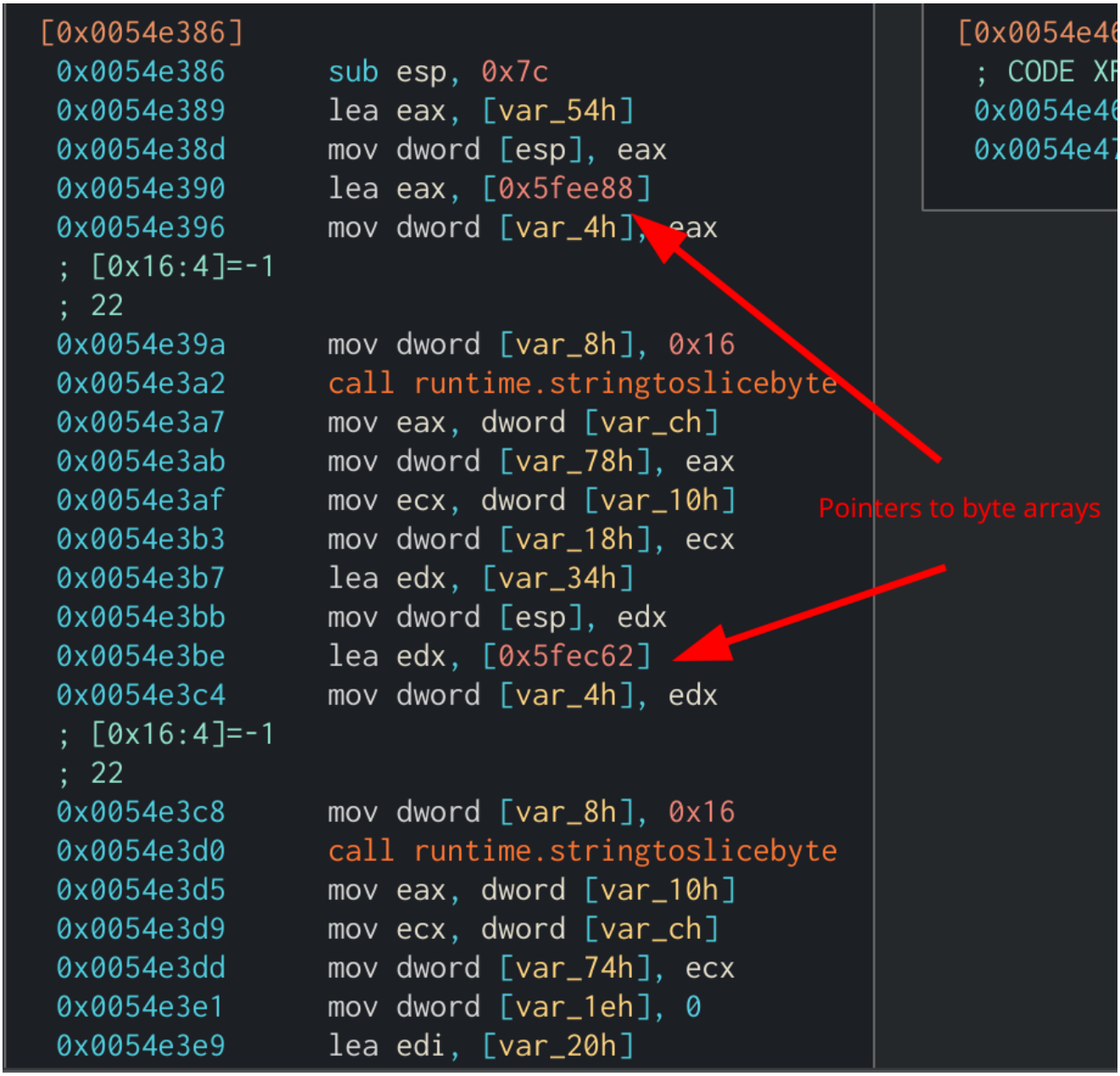
String Encryption

Each string in the binary is replaced by a function call. Each function contains two byte-arrays that are Xor'd together to return the original string. There are a few different ways that the byte-arrays are stored after the binary is compiled. The first way was through a hardcoded array.



Normal byte-array XOR Loop

The byte-arrays can also be stored in pointers, which are run through the function `stringslicetobyte` and XOR'd together.



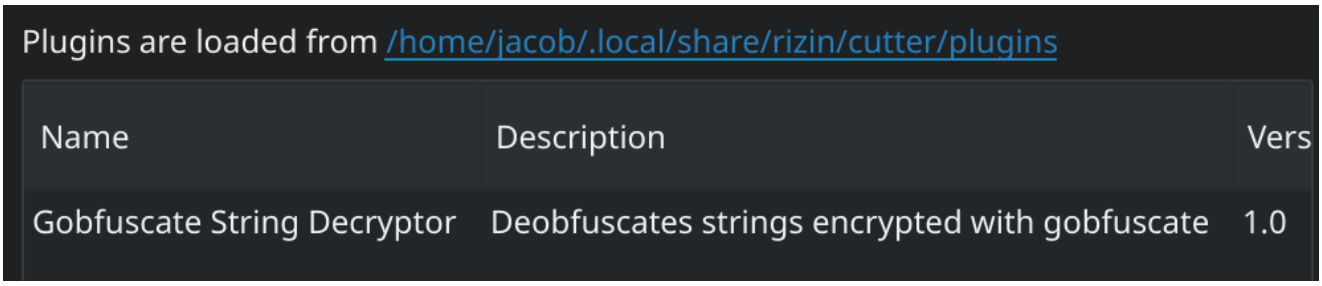
Byte-arrays being stored in pointers

These differentiations were noted when designing the deobfuscator, as not all functions will be the same. The names for the string decryption functions always contain `funcN` at the end, where `N` is an integer value. This makes them easy to spot and write a decryptor for.

How the Deobfuscator Works

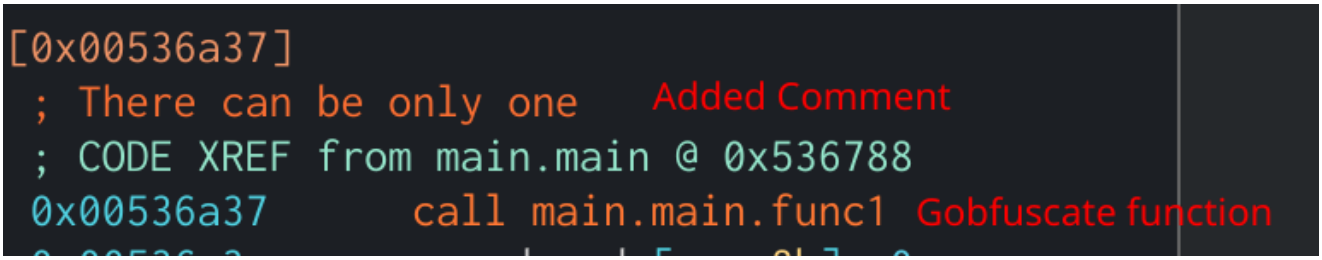
Using Cutter's API I was able to create a plugin that will either deobfuscate the string encryption function that the cursor is on or bulk deobfuscate all strings in the current method. To install the deobfuscator you will need to know the location in which Cutter stores plugins.

You can find this by going to `Edit -> Preferences -> Plugins` in Cutter.



Plugin location for Cutter

Then download the python script from the GitHub repository and move it into the `Python` folder under `plugins`. Cutter will need to be reloaded after this. To use the plugin, right-click on a gobfuscate function then select either `Plugins -> DeGobfuscate` or `Plugins -> Bulk DeGobfuscate`. The decrypted string is added as a comment above the function. If the comment doesn't appear right away, go to `View -> Refresh Contents` to refresh the screen, which should show the comment.



Example of encrypted string function

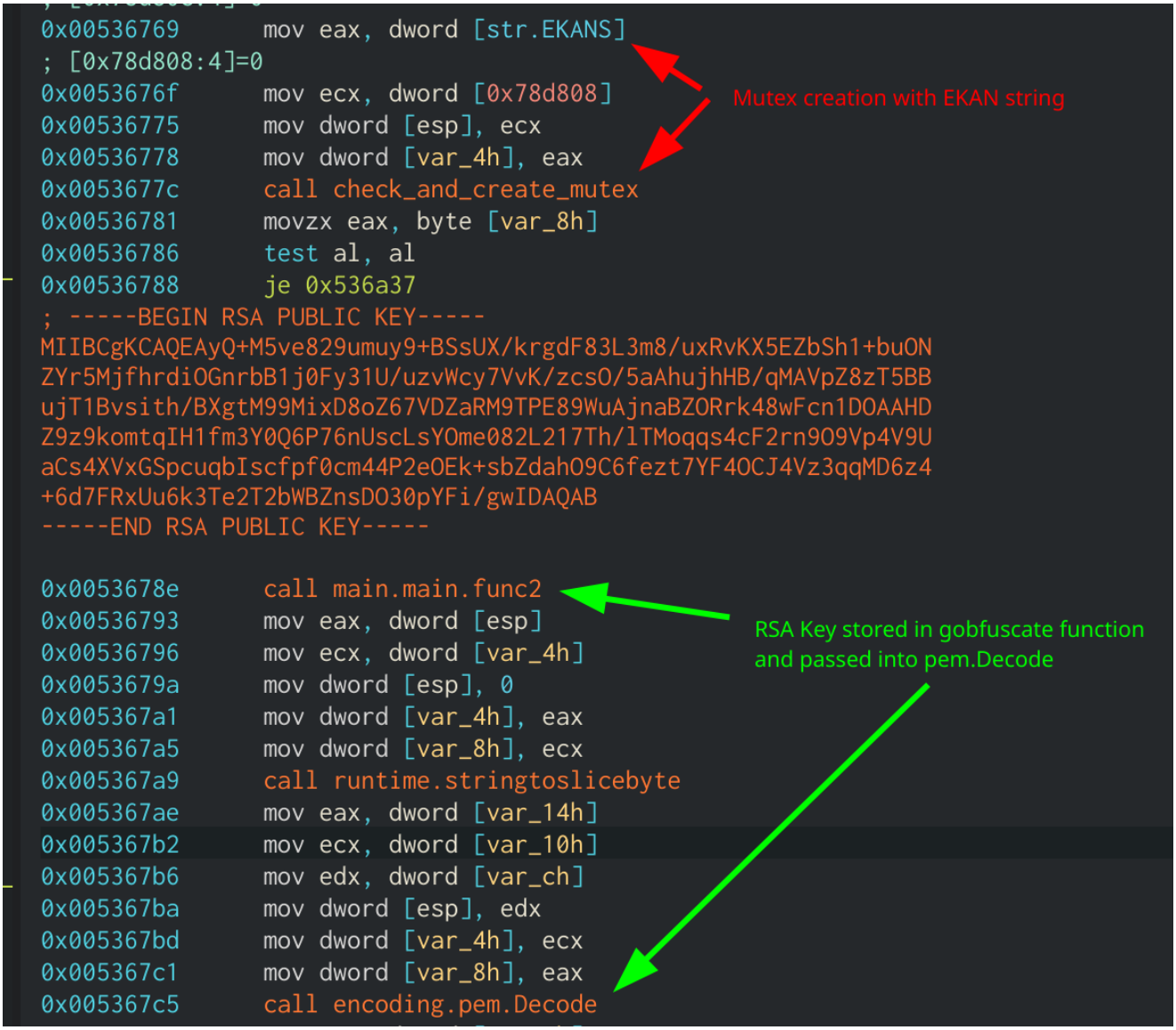
The deobfuscator utilizes Cutter's API to loop through the assembly code in the function and grab the two byte-arrays that are present. It will then XOR these together and create a comment at the location. It also checks to see if the arrays are stored in either a pointer or are hardcoded into the function.

Ekans Analysis

The Ekans ransomware has been associated with attacks on Industrial Control Systems (ICS). Ekans does not rely on outside resources to perform its functions. Everything is stored within the binary itself, mostly using the gobfuscate string encryption functions. This makes it an ideal candidate for testing the degobfuscate plugin. You can find this specific sample on Hybrid Analysis. The first step in this analysis will be to use rizin-gohelper to recover the function names from the gopclntab.

The first thing the ransomware will do is attempt to create a Mutex `Global\EKANS`. If that Mutex already exists then execution will end. It will then create the public key object that it will use to encrypt files using RSA. The public key is stored in a string in the `main.main` function, which was encrypted by gobfuscate. After running the deobfuscator over this, the public key is shown in a comment above the decryption function. It is best to view multi-line

comments in the disassembly view in Cutter since the graph view only shows the first line. This string will then be passed to Golang's `pem.Decode` function and later the `ParsePKCS1PublicKey` function.



Creation of Public Key

After this, the ransomware will create an array of objects to whitelist. This includes file extensions, file names, directories, and a regex statement. The lists are:

- File extensions:
  - .docx
  - .dll
  - .exe
  - .sys
  - .mui
  - .tmp
  - .lnk
  - .config
  - .manifest
  - .tlb
  - .olb
  - .blf
  - .ico
  - .regtrans-ms
  - .devicemetadata-ms
  - .settingcontent-ms
  - .bat
  - .cmd
  - .ps1

- File names:
  - desktop.ini
  - iconcache.db
  - ntuser.dat
  - ntuser.ini
  - ntuser.dat.log1
  - ntuser.dat.log2
  - usrclass.dat
  - usrclass.dat.log1
  - usrclass.dat.log2
  - bootmgr
  - bootnxt
  - windir
  - SystemDrive
  - ntldr
  - NTDETECT.COM
  - boot.ini
  - bootfont.bin
  - bootsect.bak
  - desktop.ini
  - ctfmon.exe
  - iconcache.db
  - ntuser.dat
- Directories:
  - `:\\$Recycle.Bin`
  - `:\\ProgramData`
  - `:\\Users\\All Users`
  - `:\\Program Files`
  - `:\\Local Settings`
  - `:\\Boot`
  - `:\\System Volume Information`
  - `:\\Recovery`
  - `\\AppData\\`
- Regex:

```
.+\\Microsoft\\(User Account Pictures|Windows\\
(Explorer|Caches)|Device Stage\\Device|Windows)\\
```

All of these strings were encrypted via gobfuscate, which is why the “bulk” option exists. Ekans will then enumerate drives and grab a list of all files that do not match the whitelists. This new file list will later be passed to worker threads for encryption.

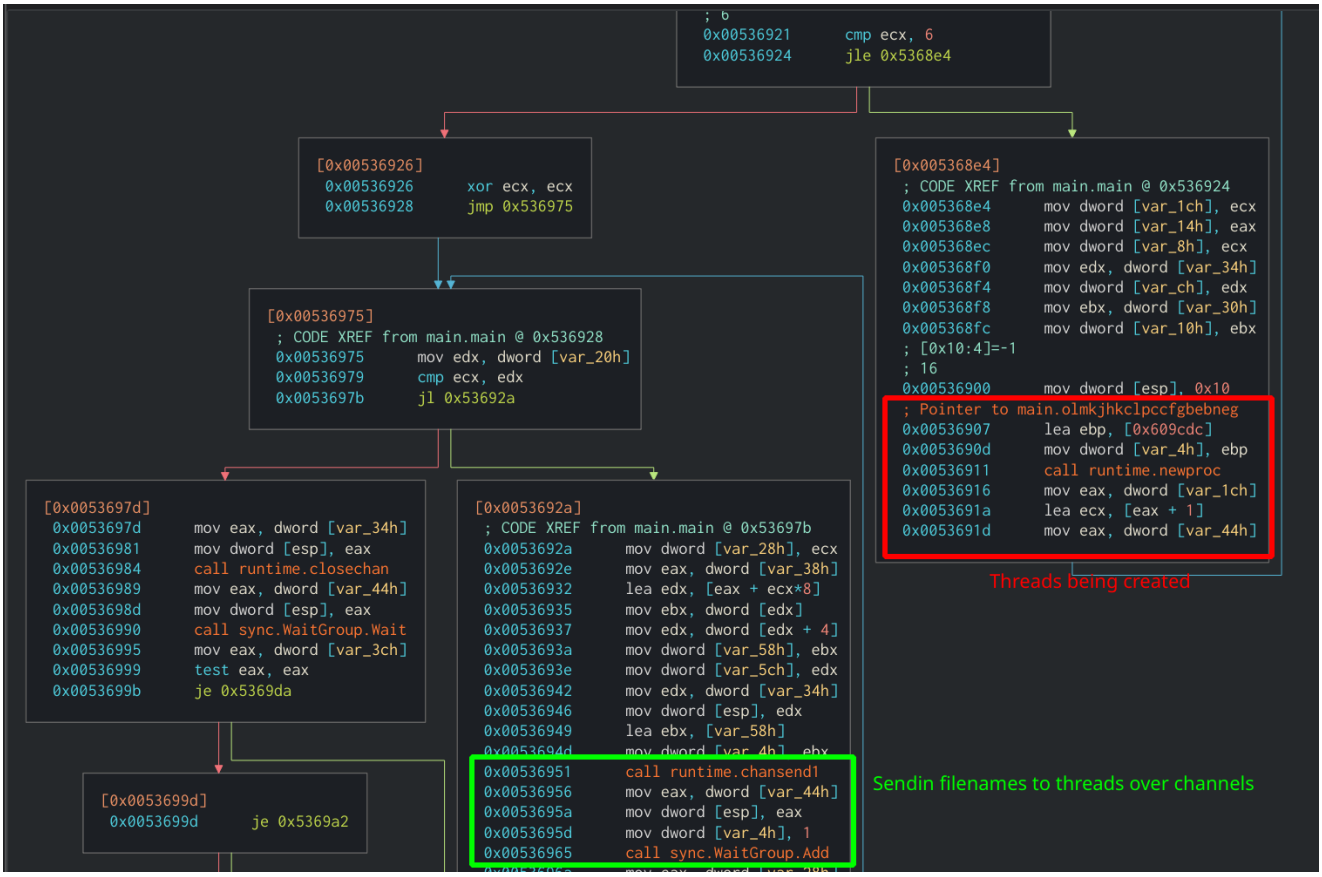
```
; .dll
; CODE XREF from create_whitelist @ 0x539ac0
0x00537de8      call main.mcmfbhdjodibjhlclidb.func2
0x00537ded      mov eax, dword [esp]
0x00537df0      mov dword [var_190h], eax
0x00537df7      mov ecx, dword [var_4h]
0x00537dfb      mov dword [var_b4h], ecx
; .exe
0x00537e02      call main.mcmfbhdjodibjhlclidb.func3
0x00537e07      mov eax, dword [esp]
0x00537e0a      mov dword [var_18ch], eax
0x00537e11      mov ecx, dword [var_4h]
0x00537e15      mov dword [var_b0h], ecx
; .sys
0x00537e1c      call main.mcmfbhdjodibjhlclidb.func4
0x00537e21      mov eax, dword [var_4h]
0x00537e25      mov dword [var_ach], eax
0x00537e2c      mov ecx, dword [esp]
0x00537e2f      mov dword [var_188h], ecx
; .mui
0x00537e36      call main.mcmfbhdjodibjhlclidb.func5
0x00537e3b      mov eax, dword [esp]
0x00537e3e      mov dword [var_184h], eax
0x00537e45      mov ecx, dword [var_4h]
0x00537e49      mov dword [var_a8h], ecx
; .tmp
0x00537e50      call main.mcmfbhdjodibjhlclidb.func6
```

Whitelist creation function

The ransomware will then kill a list of 288 hard-coded services. Instead of listing all of the services in this article here, you can find them here. The Ekans process will then kill a list of 1118 processes, which are also included in the linked repository. Ekans will then delete shadow copies using a `WbemScripting.SWbemLocator` object with the following WMI query:

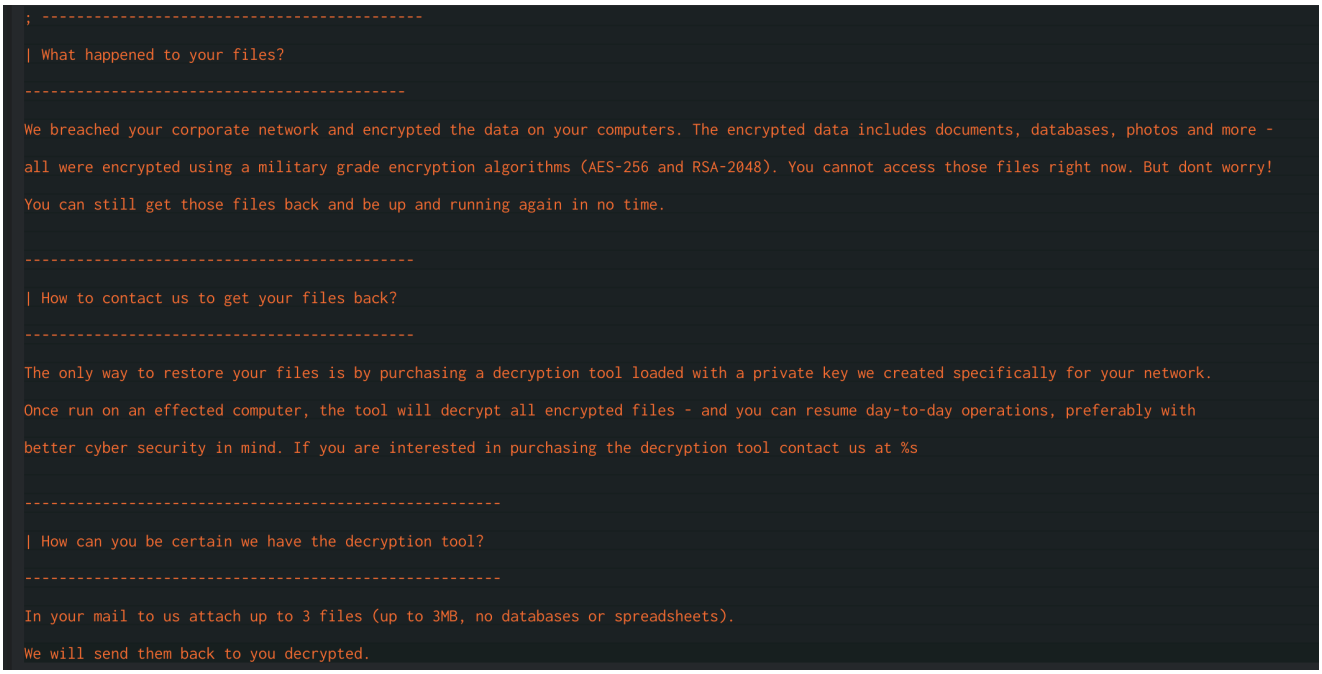
```
SELECT * FROM Win32_ShadowCopy
```

After this, the ransomware will create several threads and pass in the filenames to these via GolLang's `channel` functions. The threads will take the filenames, encrypt the files, and write them back to disk.



Loop used to create encryption threads

Finally, the ransom note is dropped to the file `Fix-Your-Files.txt`. The note itself is hard-coded and uses the `sprintf` function with the ransomware author's email to format the note, which in this case is `bapcocrypt@ctemplar.com`.



Unformatted Ransom Note

Conclusion

I did not want to delve too deep into the Ekans ransomware analysis as this was to demonstrate the usefulness of the degobfuscator plugin. This was my first attempt at making a plugin for Cutter and I enjoyed the challenge very much. I am excited to see what Cutter has in store for the future and will continue to make plugins for it to aid other analysts. As always, if you have any questions feel free to reach out to me on my Twitter or LinkedIn.

Thanks for reading and happy reversing!

Malware Analysis, GoLang, Cutter, Ekans, Ransomware

More Content Like This: