BleachGap Revamped

labs.k7computing.com/index.php/bleachgap-revamped/

By Gaurav Yadav August 25, 2022

BleachGap ransomware was first reported in Feb 2021 by a researcher named Petrovic on Twitter. This **ransomware variant** that we have analysed was reported on Twitter in June 2022. This variant got us curious to get into the nuances of it because it was tagged as a **stealer and all the code was compiled in a single executable thereby not needing any supporting .bat or PowerShell scripts to execute**, most probably done for evasion and to be less noisy in comparison to the variant found in 2021, which needed the supporting .bat and .exe that it dropped for execution.Though there are not many cases reported in the wild, this blog has been written **to let the cyber community know that threat actors are modifying the attack techniques of this malware for a possible major attack that might be planned in the future.** Lets now get into the details.

Why a Stealer?

When this ransomware executes, the first step is to get the username and generate a **Unique ID (UID)** and **Password** for that particular victim. By the first look, it seems it is stealing the password from the user but after multiple executions we identified that the password is different every time and after reversing the sample we found that the ransomware is using a function (shown in Figure 1) to randomly generate the UID and the same function is being called again to generate the password which is always 32 byte long.

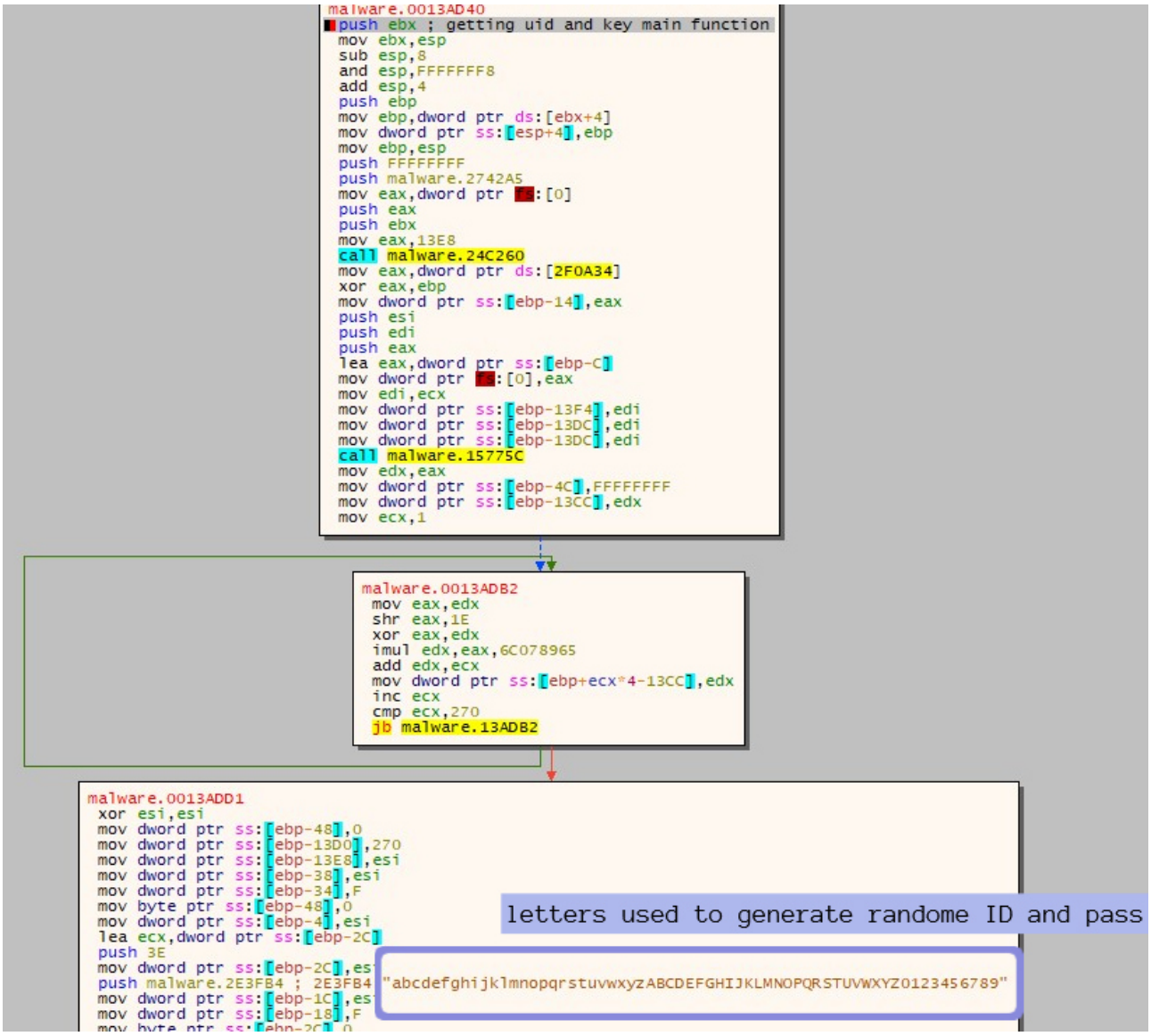

Figure 1: Function used to generate random ID and Pass

After generating the UID and Password, it gets the Username of the current user using the environment variable. It first moves the encoded bytes to memory which are already hardcoded in the executable and then decodes those to the **'username'** and then uses it as an argument to get environment variable data related to the username as shown in Figure 2.

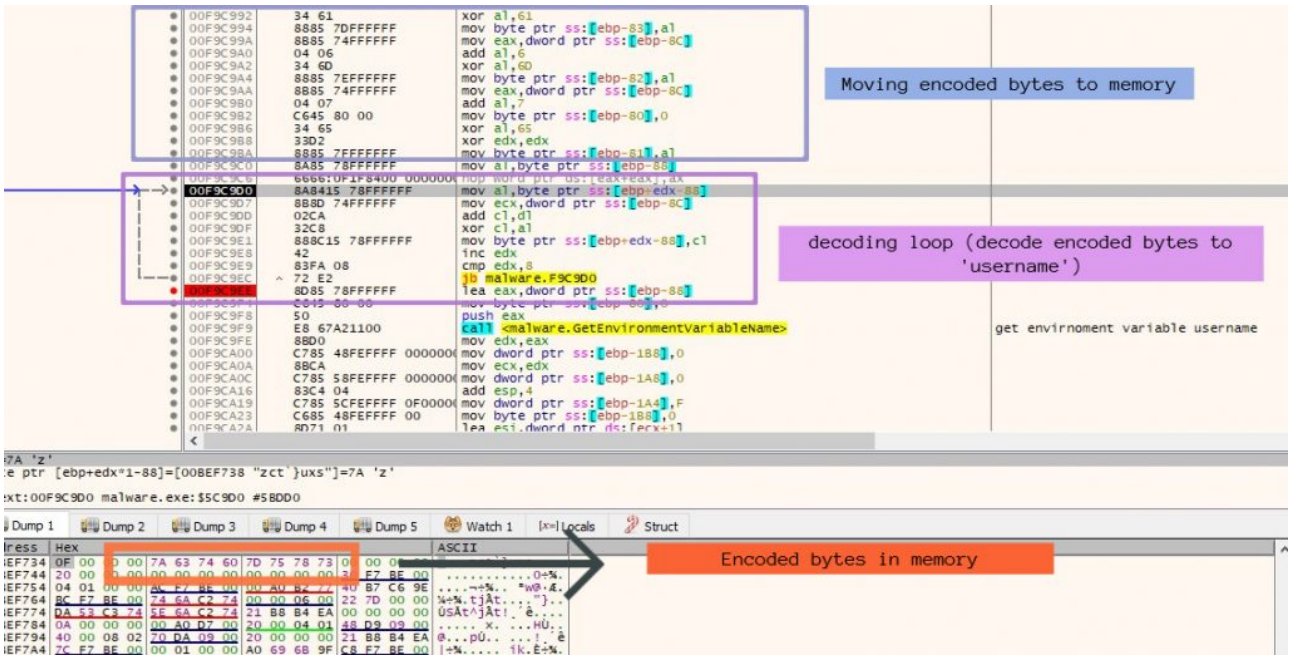

Figure 2: Getting username using environment variables

Instead of hardcoding the useful strings into executables directly, so as to evade detection, this ransomware has used a similar method of moving encoded strings into memory and then decoding them at runtime for different purposes.We will see similar method being used later in this ransomware. After getting the Username, it forms a huge string using the same method of decoding the encoded bytes which includes UID, Password, Username as shown in Figure 3.

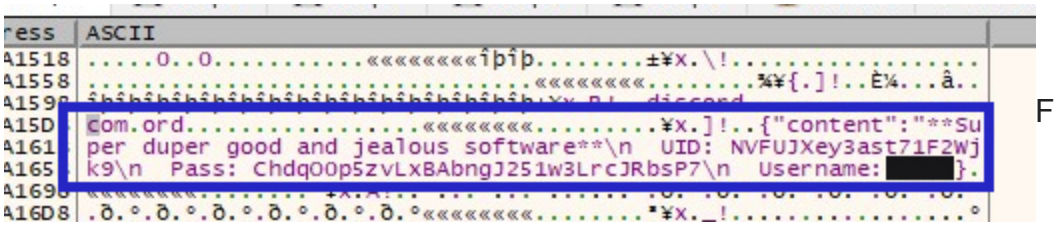

Figure 3: Large decoded string

After further analysis we learn that this ransomware sends the large decoded string shown in Figure 3 as a Post request to the Discord Webhook API which has been highlighted in Figure 4 and 5.

FIgure 4: Post request to Discord API

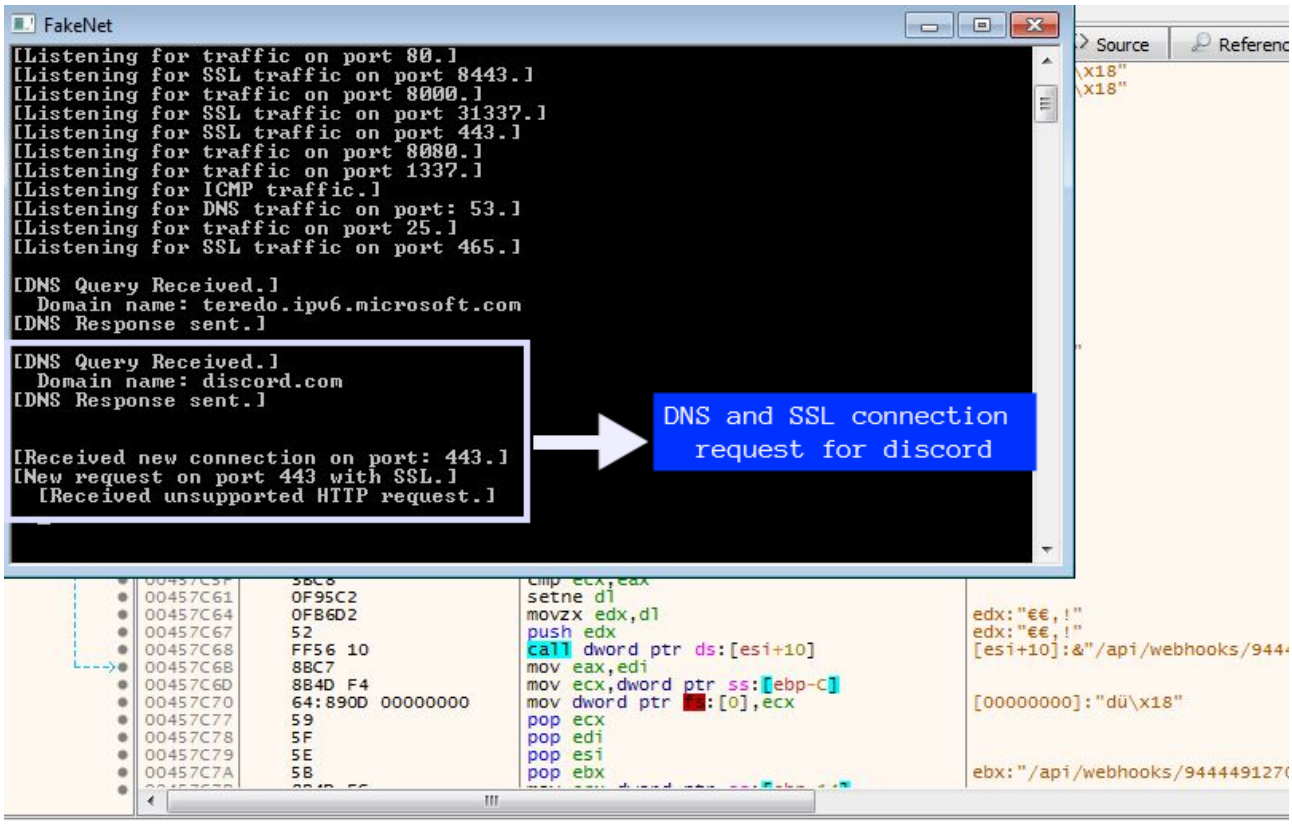

Figure 5: Fakenet output for DNS and SSL connection

Disabling Tools to Work

After sending information to the Discord API, the ransomware tries to disable tools like command prompt (CMD), Task Manager and Registry Editor so that the user is not able to make changes and stop the ransomware execution. Disabling the mentioned tools happens with the help of the registry. Ransomware first copies the encoded registry key into memory and then decodes the key using XOR loop and then does the same for key value and then calls the function **RegCreateKey** using the decoded key and value as arguments. Figure 6 shows the encoded registry key.

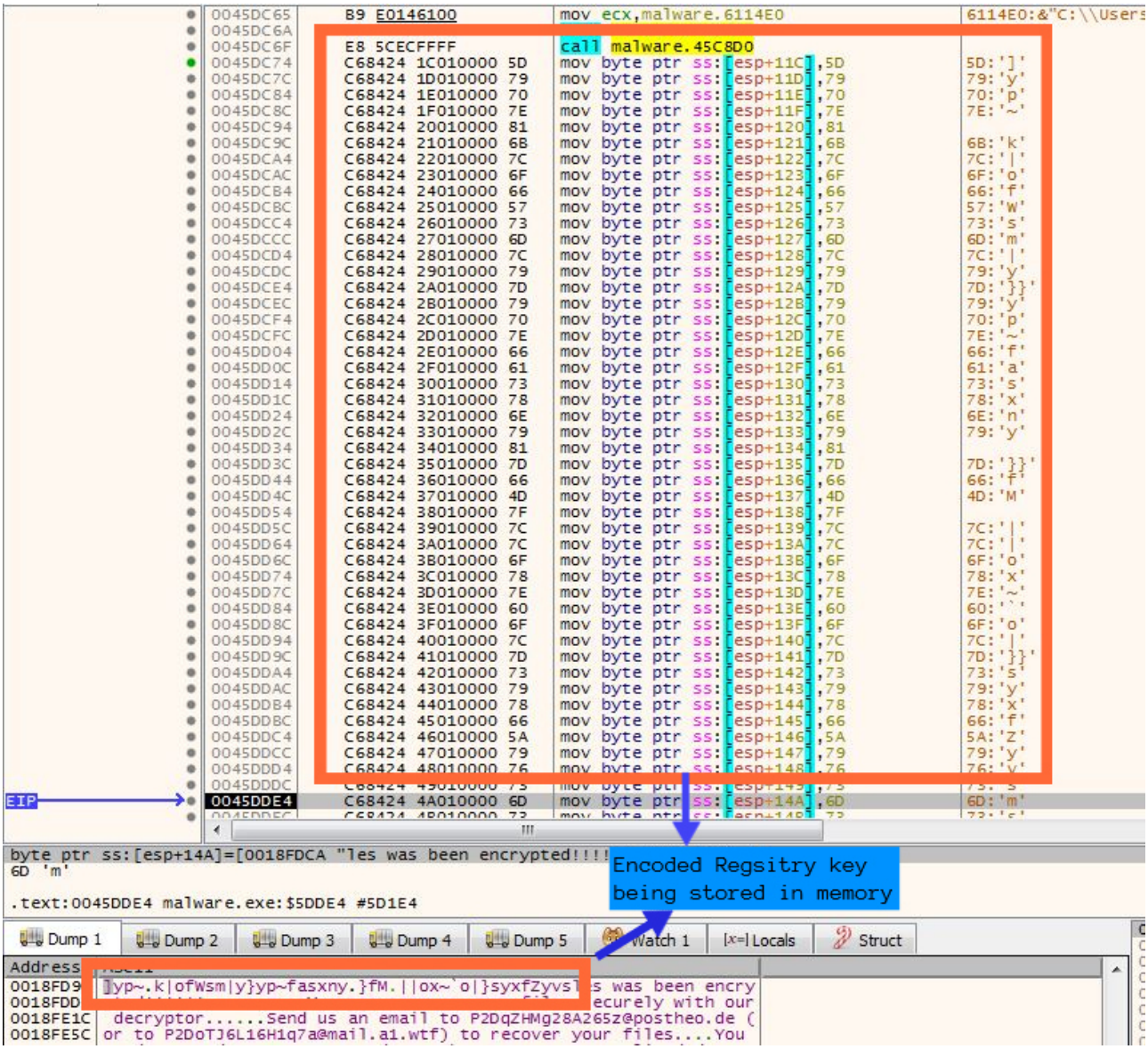

Figure 6: Encoded Registry key

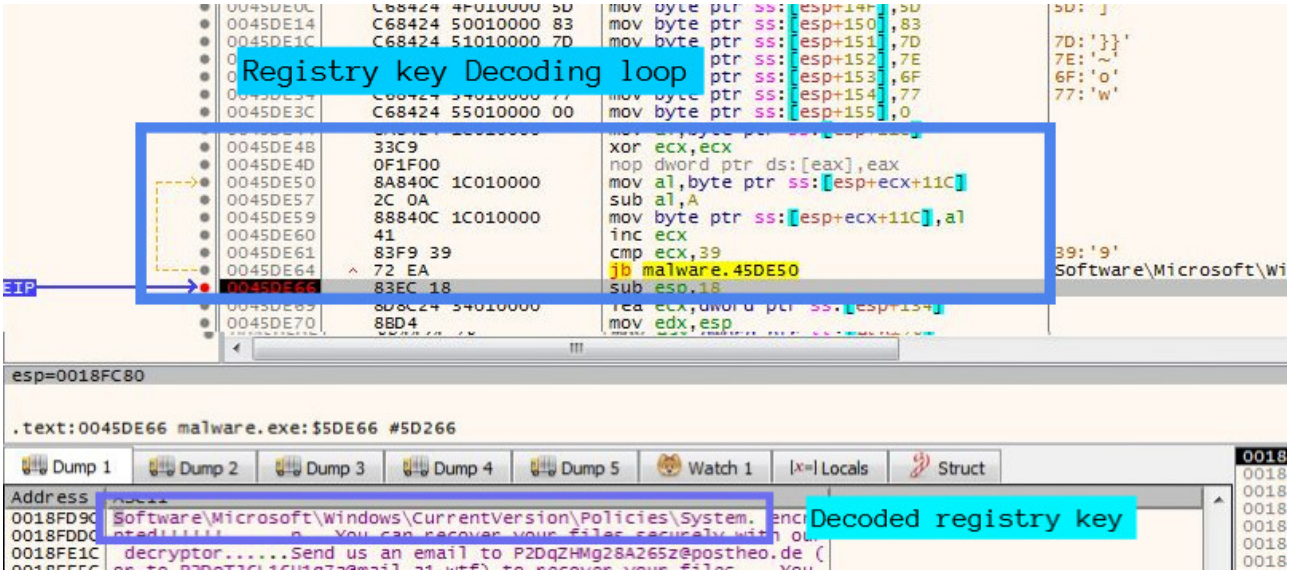

Figure 7: Decoding encoded registry key

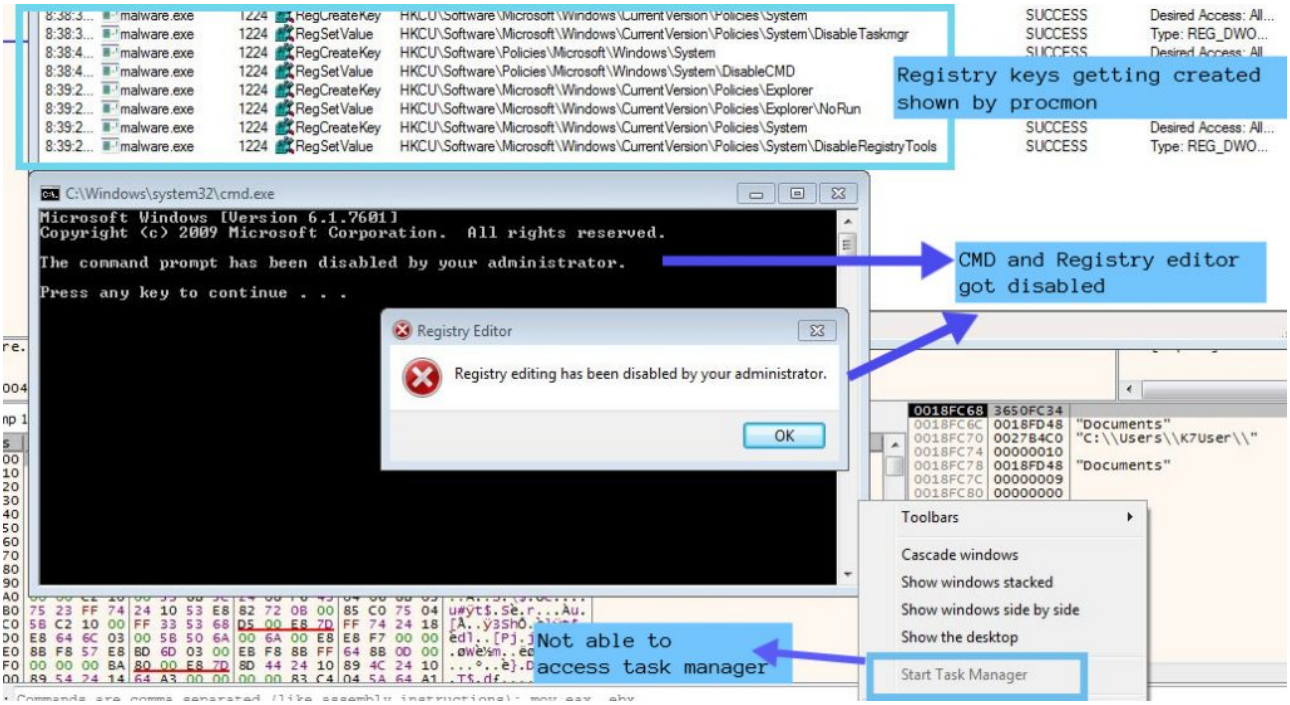

Figure 8: Tools getting disabled

After disabling the tools, the ransomware decodes the different folder names which includes Desktop, Documents, Downloads, Pictures, Music ,Public and adds it to the string **C:\Users\%username%** so that it can enumerate these folders first and encrypt the files stored inside.

Figure 9: Function adding folder names after decoding

After getting all the folder names, the ransomware starts to enumerate them using **FindFirstFileExW** and **FindNextFileW** and then uses **ReadFile** to read the existing file into a buffer for encryption.

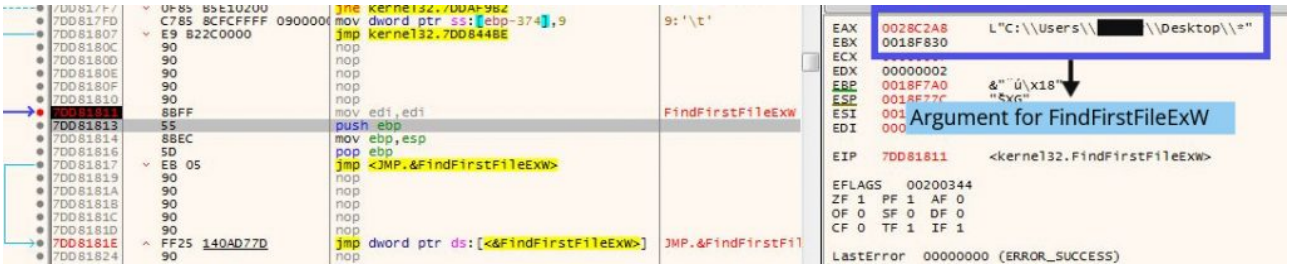

Figure 10: Using **FindFirstFileExW**

Figure 11: Using **FindNextFileW**

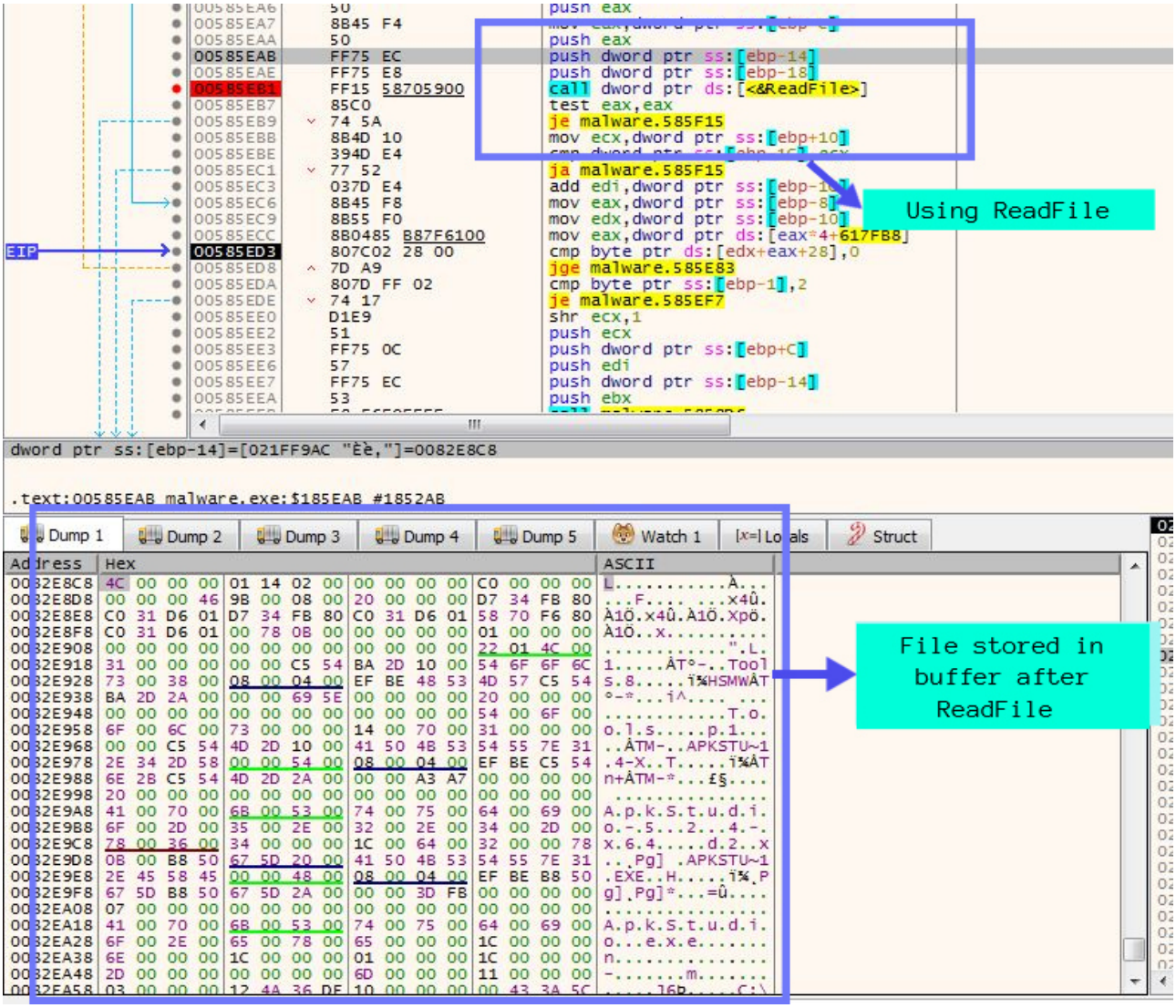

Figure 12: Using **ReadFile** to read **ApkStudio.exe.lnk**

Encrypting Files

When analysing the sample after ReadFile we observed that the sample doesn't use any common encryption related Windows APIs like **CryptAcquireContextA, CryptReleaseContext, CryptGenKey, CryptExportKey,** etc**.** On further digging, we found that the whole encryption routine is implemented inside the ransomware. We identified this when a hardware breakpoint on the randomly generated password (described at the start of the blog) was hit after the ReadFile API. There were some calculations happening with the password and some bytes were also present in the .**rdata** section.

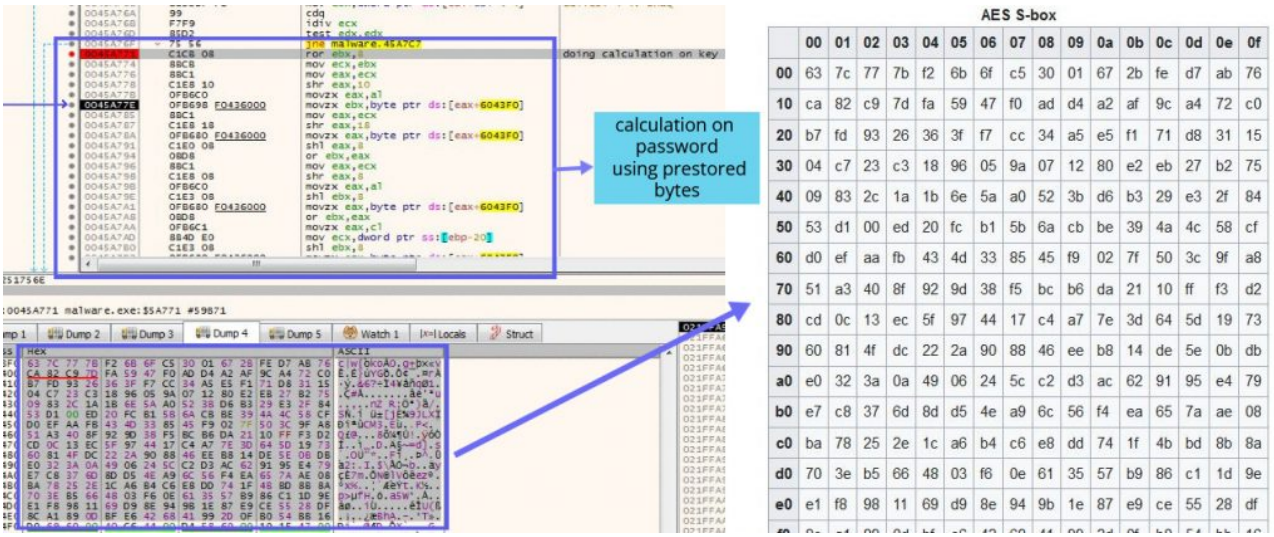

Figure 13: Using AES S-block for key expansion

After checking those prestored bytes we identified that it is an S-Block used in the AES Algorithm during the Key Expansion phase. So ransomware is using the AES algorithm to encrypt the files using the password (key) that was randomly generated and sent to discord webhook. On further analysis, we got the functions which were responsible for encrypting the file bytes and writing it to the memory 16 bytes at a time as in Figure 14 and Figure 15.

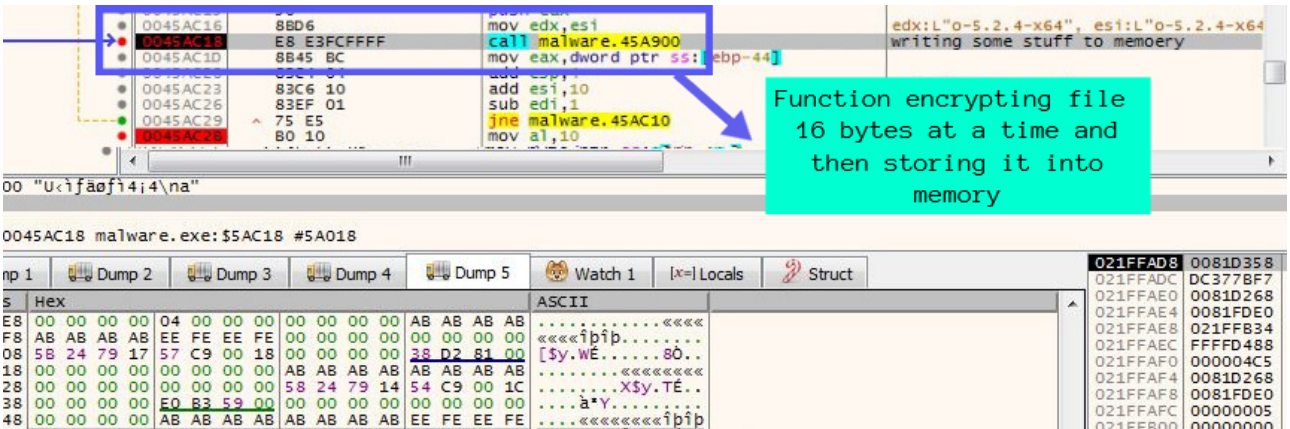

Figure 14: **ApkStudio.exe.lnk** File getting encrypted

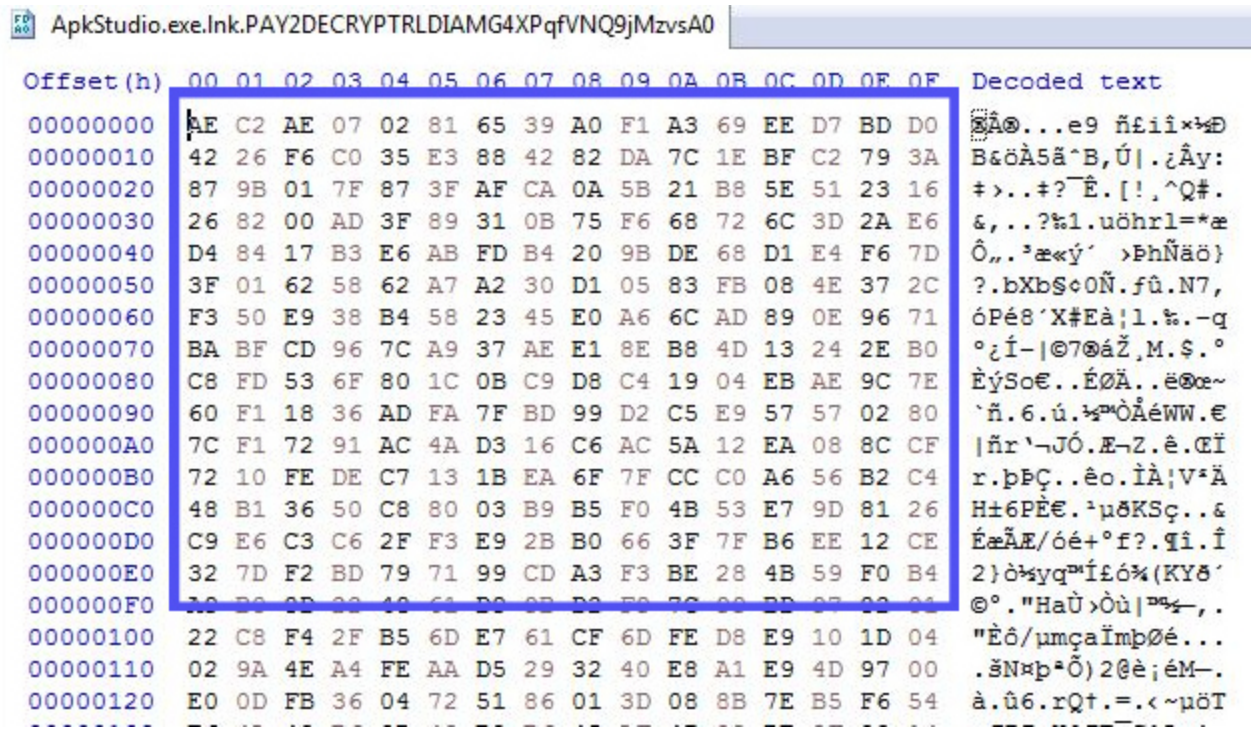

Figure 15: Encrypted File **ApkStudio.exe.lnk**

Ransomware creates a new file with the same name and writes the encrypted bytes into that file and then renames the file with the extension **PAY2DECRYPT+UID.** After encrypting the files, it puts 100 ransom notes on the Desktop. This ransomware encrypts executables as well. It changes its own name to encrypted file extension but the file remains as-is and not encrypted.

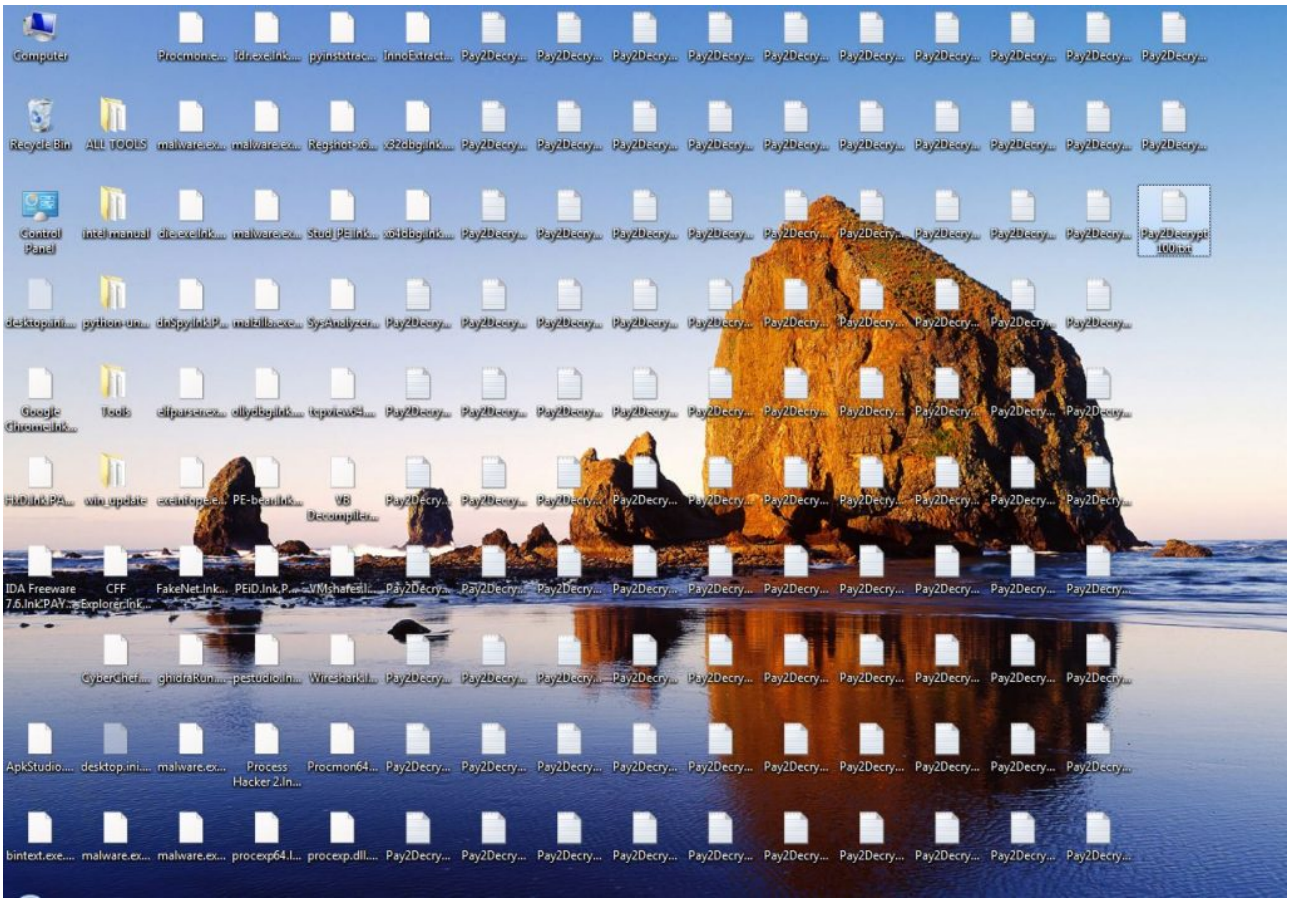

Figure 16: 100 ransom notes on Desktop

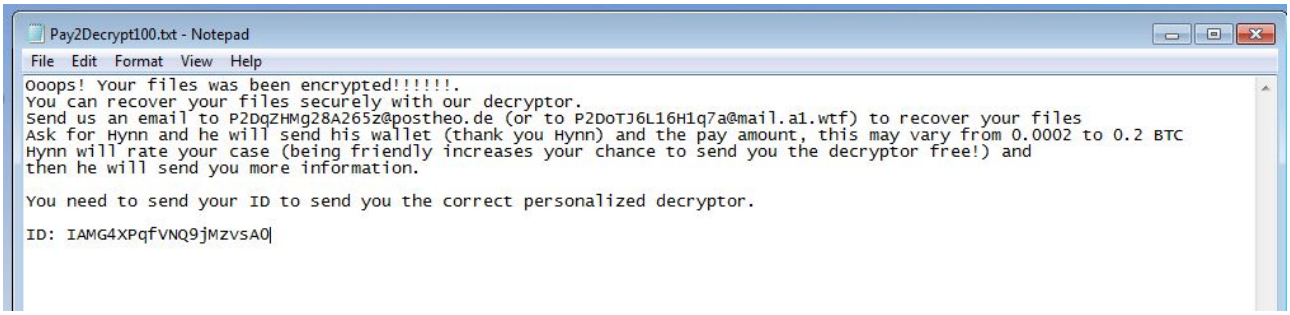

Figure 17: Ransom Note

We at K7 Labs provide detection for BleachGap ransomware and all the latest threats. Users are advised to use a reliable security product such as **“K7 Total Security”** and keep it up-to-date to safeguard their devices.

Indicators of Compromise (IOCs)

File Name	Hash	Detection Name
ransomito.exe	bfe289c6f91ffcda97c207f3c1c525a9	Riskware (00584baa1)

References

https://www.goggleheadedhacker.com/blog/post/reversing-crypto-functions-aes

https://twitter.com/Finch39487976/status/1533126802159304705

https://www.reversingsecurity.com/blog/pay2decrypt-bleachgap-analysis