

# BleachGap Revamped

---

 [labs.k7computing.com/index.php/bleachgap-revamped/](https://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.

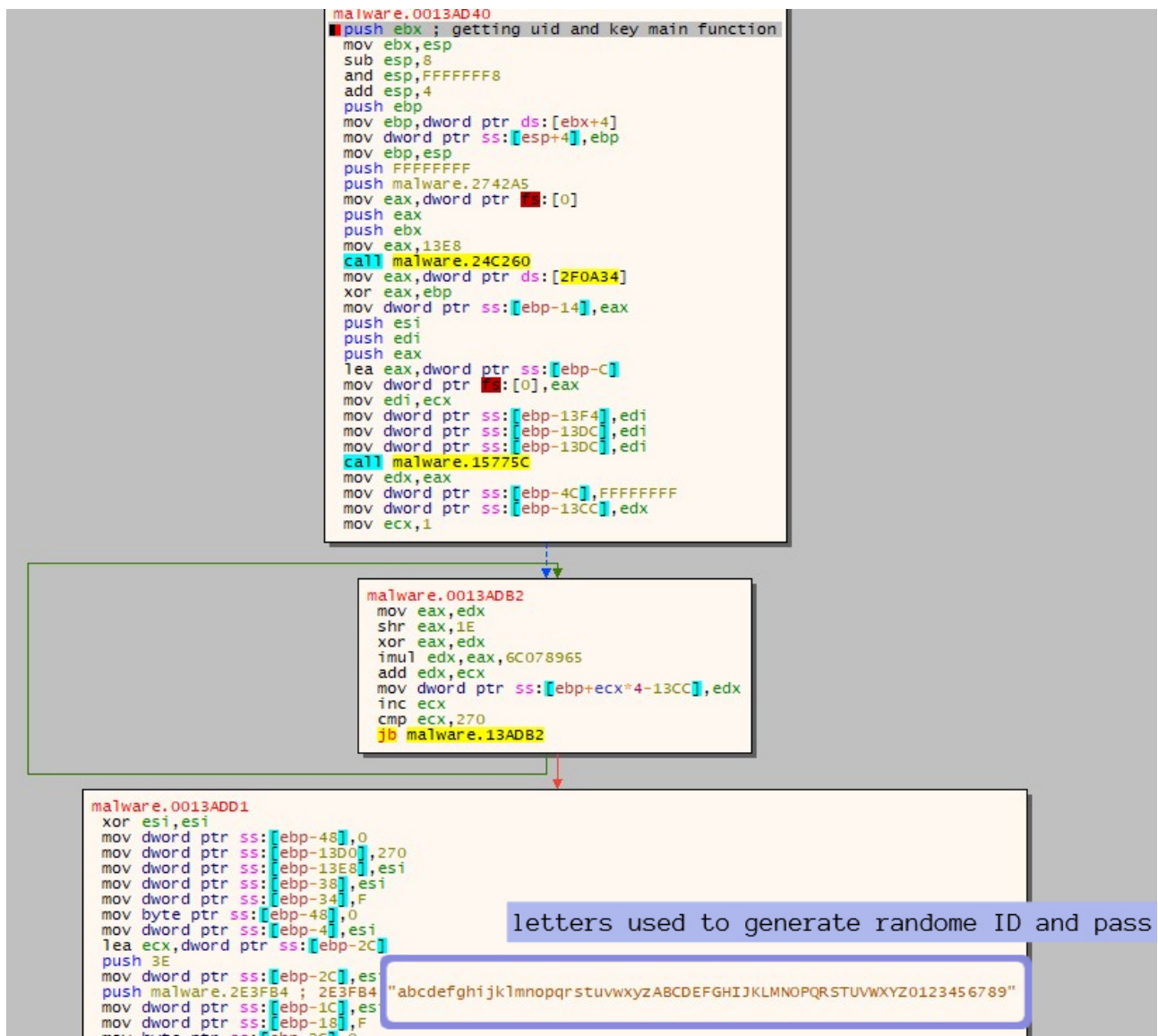


Figure 1: Function used to generate random ID and Pass

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.

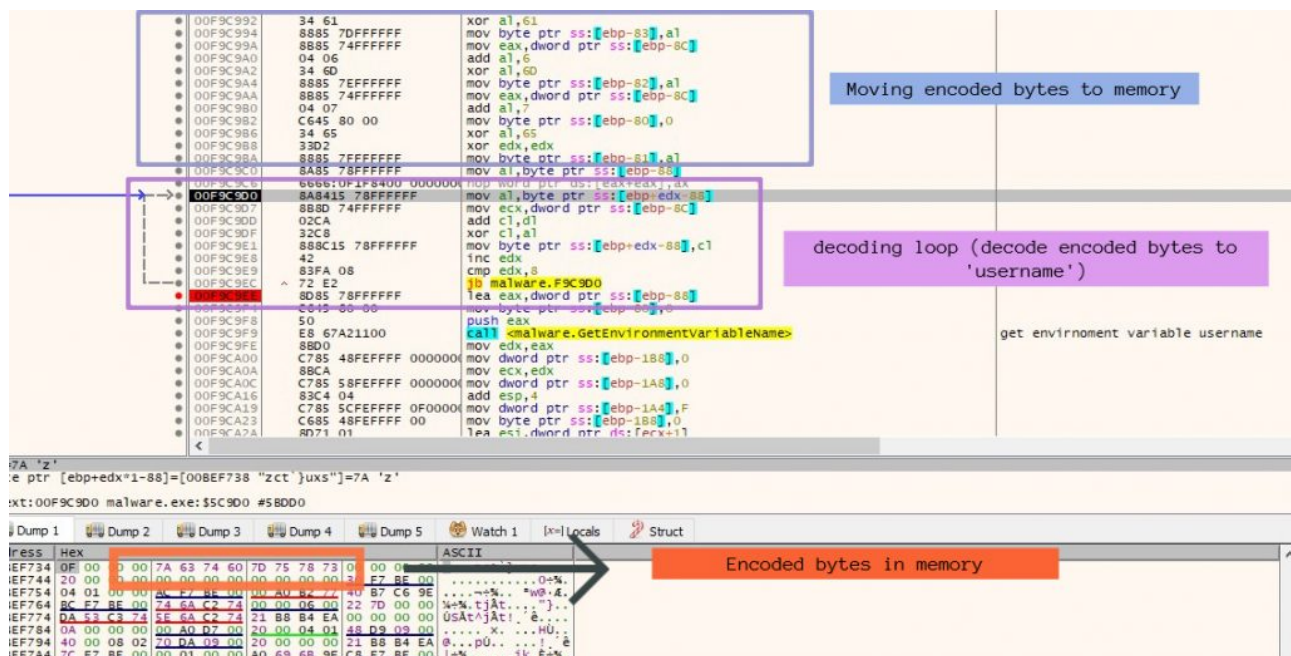
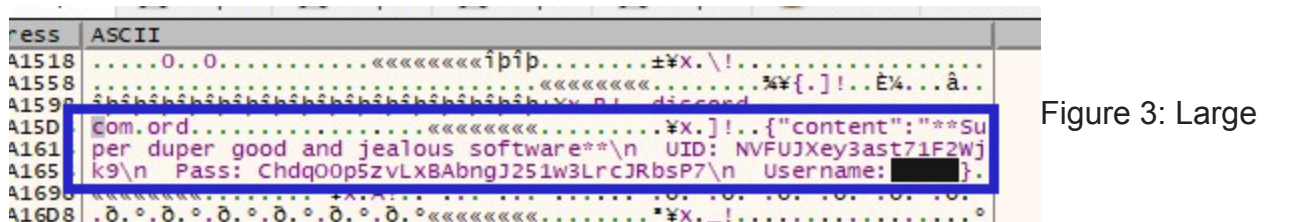


Figure 2: Getting username using environment variables

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.



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

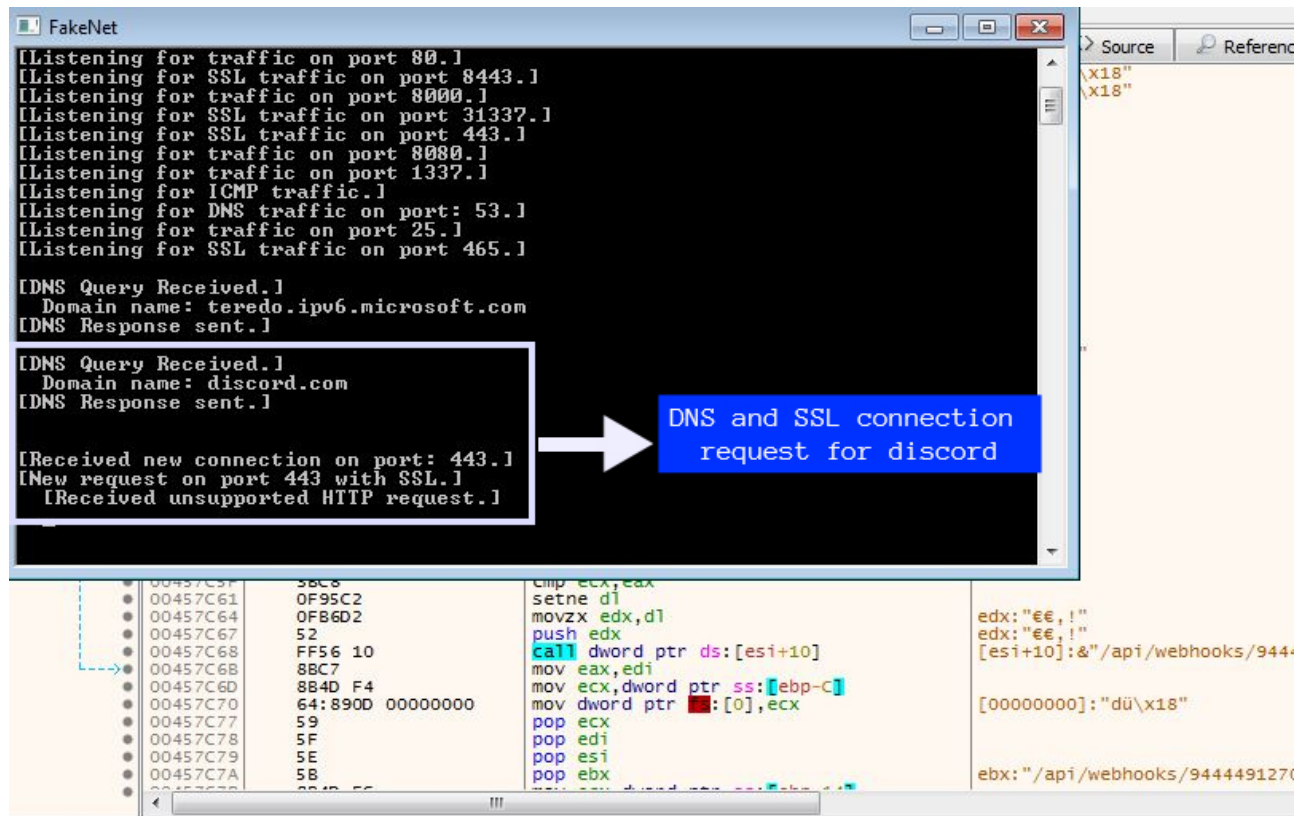


Figure 5: Fakenet output for DNS and SSL connection

## 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.

Assembly code snippet (addresses 0045DC65 to 0045DDDC):

```

B9 E0146100 mov ecx,malware.6114E0 6114E0:&"C:\Users
E8 5CECFFFF call malware.45C8D0
C68424 1C010000 5D mov byte ptr ss:[esp+11C],5D
C68424 1D010000 79 mov byte ptr ss:[esp+11D],79
C68424 1E010000 70 mov byte ptr ss:[esp+11E],70
C68424 1F010000 7E mov byte ptr ss:[esp+11F],7E
C68424 20010000 81 mov byte ptr ss:[esp+120],81
C68424 21010000 68 mov byte ptr ss:[esp+121],68
C68424 22010000 7C mov byte ptr ss:[esp+122],7C
C68424 23010000 6F mov byte ptr ss:[esp+123],6F
C68424 24010000 66 mov byte ptr ss:[esp+124],66
C68424 25010000 57 mov byte ptr ss:[esp+125],57
C68424 26010000 73 mov byte ptr ss:[esp+126],73
C68424 27010000 6D mov byte ptr ss:[esp+127],6D
C68424 28010000 7C mov byte ptr ss:[esp+128],7C
C68424 29010000 79 mov byte ptr ss:[esp+129],79
C68424 2A010000 7D mov byte ptr ss:[esp+12A],7D
C68424 2B010000 79 mov byte ptr ss:[esp+12B],79
C68424 2C010000 70 mov byte ptr ss:[esp+12C],70
C68424 2D010000 7E mov byte ptr ss:[esp+12D],7E
C68424 2E010000 66 mov byte ptr ss:[esp+12E],66
C68424 2F010000 61 mov byte ptr ss:[esp+12F],61
C68424 30010000 73 mov byte ptr ss:[esp+130],73
C68424 31010000 78 mov byte ptr ss:[esp+131],78
C68424 32010000 6E mov byte ptr ss:[esp+132],6E
C68424 33010000 79 mov byte ptr ss:[esp+133],79
C68424 34010000 81 mov byte ptr ss:[esp+134],81
C68424 35010000 7D mov byte ptr ss:[esp+135],7D
C68424 36010000 66 mov byte ptr ss:[esp+136],66
C68424 37010000 4D mov byte ptr ss:[esp+137],4D
C68424 38010000 7F mov byte ptr ss:[esp+138],7F
C68424 39010000 7C mov byte ptr ss:[esp+139],7C
C68424 3A010000 7C mov byte ptr ss:[esp+13A],7C
C68424 3B010000 6F mov byte ptr ss:[esp+13B],6F
C68424 3C010000 78 mov byte ptr ss:[esp+13C],78
C68424 3D010000 7E mov byte ptr ss:[esp+13D],7E
C68424 3E010000 60 mov byte ptr ss:[esp+13E],60
C68424 3F010000 6F mov byte ptr ss:[esp+13F],6F
C68424 40010000 7C mov byte ptr ss:[esp+140],7C
C68424 41010000 7D mov byte ptr ss:[esp+141],7D
C68424 42010000 73 mov byte ptr ss:[esp+142],73
C68424 43010000 79 mov byte ptr ss:[esp+143],79
C68424 44010000 78 mov byte ptr ss:[esp+144],78
C68424 45010000 66 mov byte ptr ss:[esp+145],66
C68424 46010000 5A mov byte ptr ss:[esp+146],5A
C68424 47010000 79 mov byte ptr ss:[esp+147],79
C68424 48010000 76 mov byte ptr ss:[esp+148],76
C68424 49010000 75 mov byte ptr ss:[esp+149],75
C68424 4A010000 6D mov byte ptr ss:[esp+14A],6D
C68424 4B010000 73 mov byte ptr ss:[esp+14B],73

```

Memory dump (Address 0018FD9 to 0018FE5C):

```

0018FD9 0018FDD 0018FE1C 0018FE5C
|yp~.k|ofwsm|y|yp~fasxny.}fM.||ox~'o|}syxfZyvs|
is was been encry
securely with our
decryptor.....Send us an email to P2DqZHmG28A265z@posttheo.de (
or to P2DoTj6L16H1q7a@mail.a1.wtf) to recover your files....You

```

Figure 6: Encoded Registry key

Assembly code snippet (addresses 0045DE14 to 0045DE66):

```

0045DE14 C68424 50010000 83 mov byte ptr ss:[esp+150],83
0045DE1C C68424 51010000 7D mov byte ptr ss:[esp+151],7D
0045DE24 C68424 52010000 7E mov byte ptr ss:[esp+152],7E
0045DE34 C68424 53010000 6F mov byte ptr ss:[esp+153],6F
0045DE3C C68424 54010000 00 mov byte ptr ss:[esp+154],0
0045DE48 33C9 xor ecx,ecx
0045DE4D 0F1F00 nop dword ptr ds:[eax],eax
0045DE50 8A840C 1C010000 mov al,byte ptr ss:[esp+ecx+11C]
0045DE57 2C 0A sub al,A
0045DE59 88840C 1C010000 mov byte ptr ss:[esp+ecx+11C],al
0045DE60 41 inc ecx
0045DE61 83F9 39 cmp ecx,39
0045DE64 72 EA jb malware.45DE50
0045DE66 83EC 18 sub esp,18
0045DE69 808C24 54010000 mov ecx,dword ptr ss:[esp+154]
0045DE70 8BD4 mov edx,esp

```

Memory dump (Address 0018FD9C to 0018FF5C):

```

0018FD9C 0018FDDC 0018FE1C 0018FF5C
Software\Microsoft\Windows\CurrentVersion\Policies\System.
You can recover your files securely with our
decryptor.....Send us an email to P2DqZHmG28A265z@posttheo.de (
or to P2DoTj6L16H1q7a@mail.a1.wtf) to recover your files....You

```

Figure 7: Decoding encoded registry key

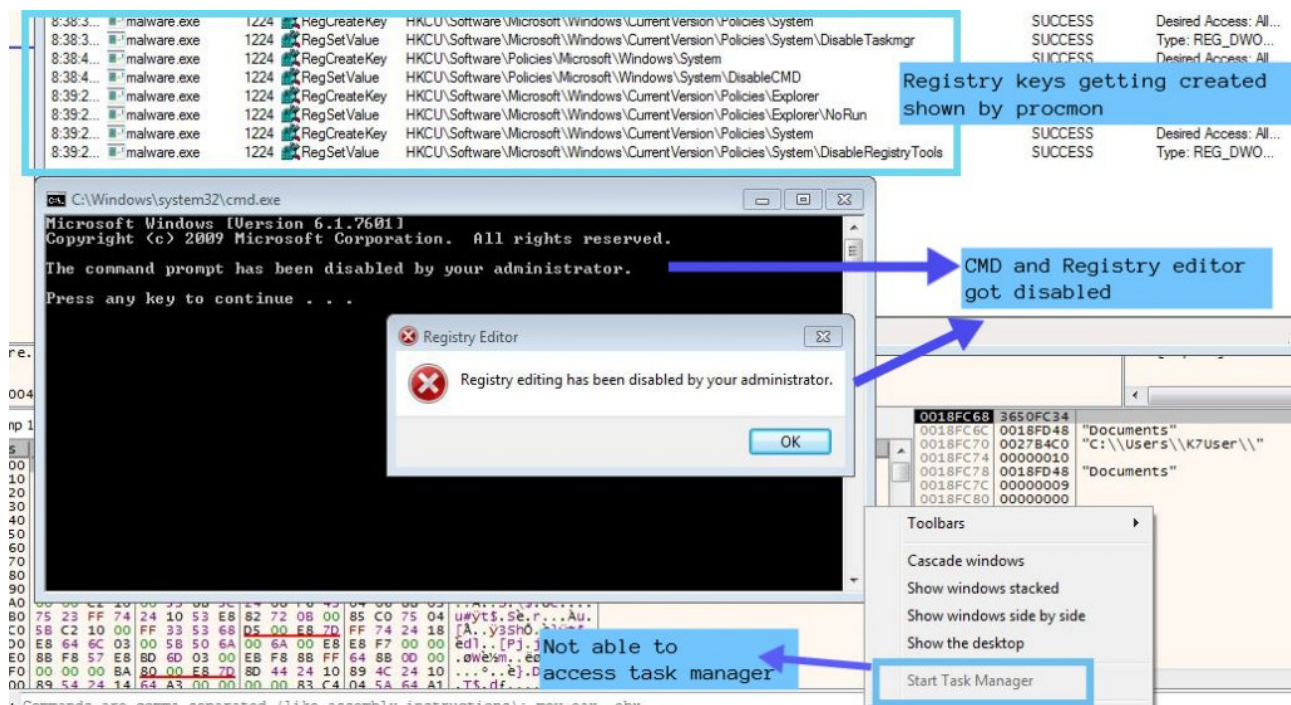


Figure 8: Tools getting disabled

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.

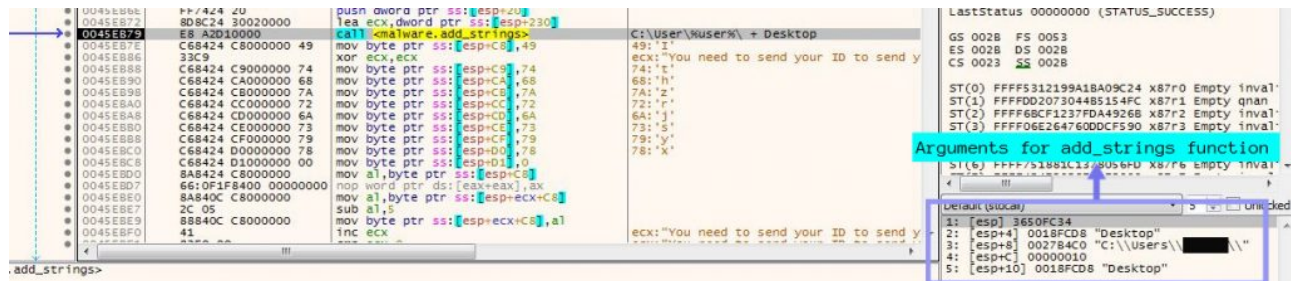


Figure 9: Function adding folder names after decoding

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.

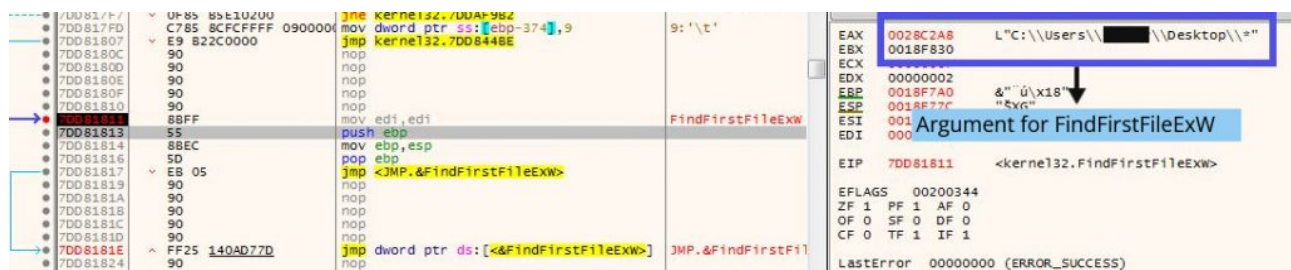


Figure 10: Using FindFirstFileExW



Figure 11: Using FindNextFileW

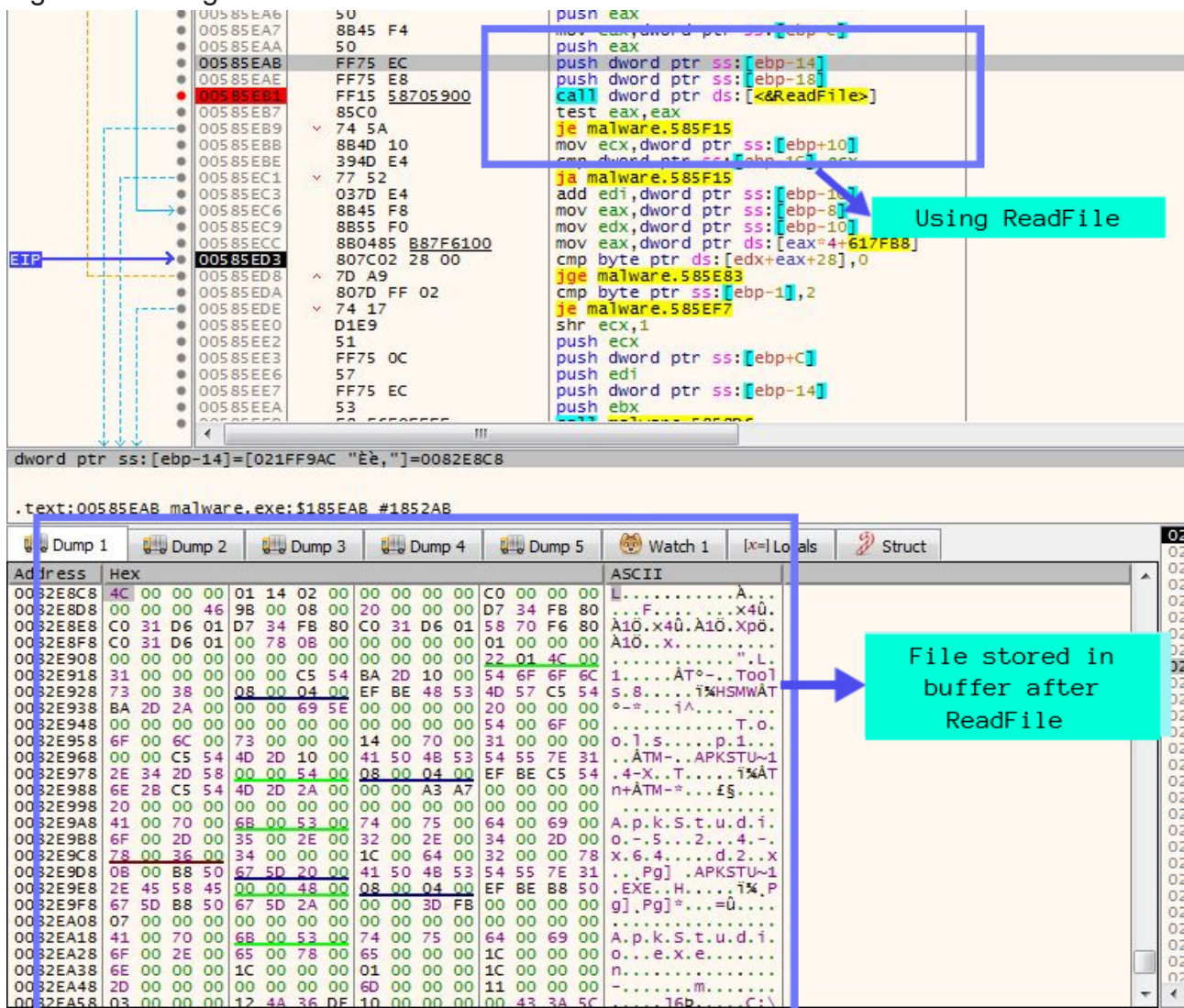


Figure 12: Using ReadFile to read ApkStudio.exe.Ink

## 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.

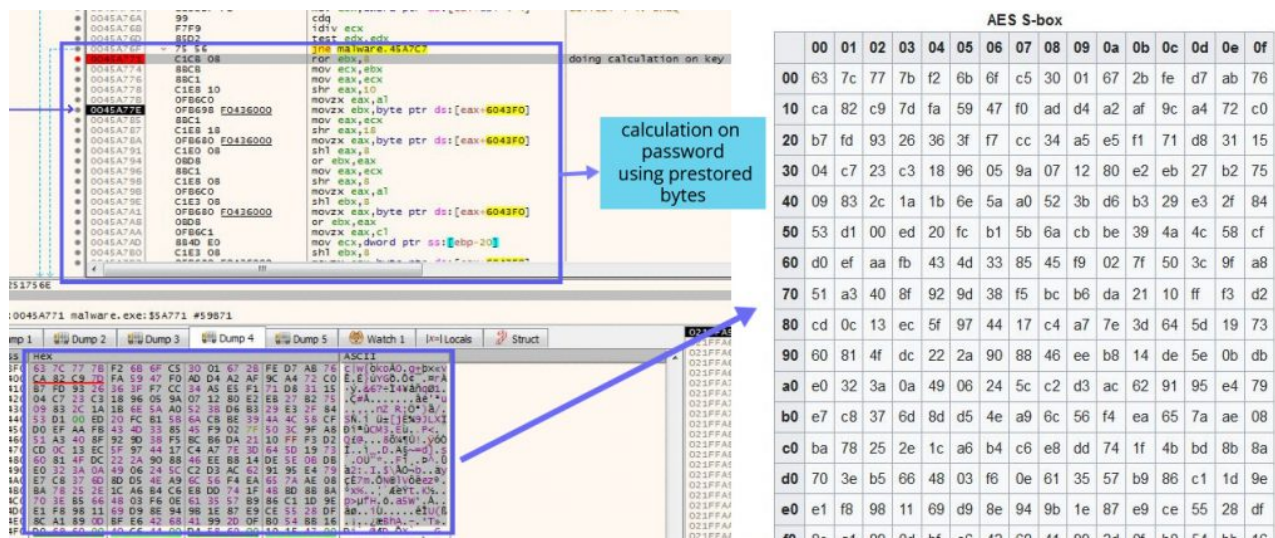


Figure 13: Using AES S-block for key expansion

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.

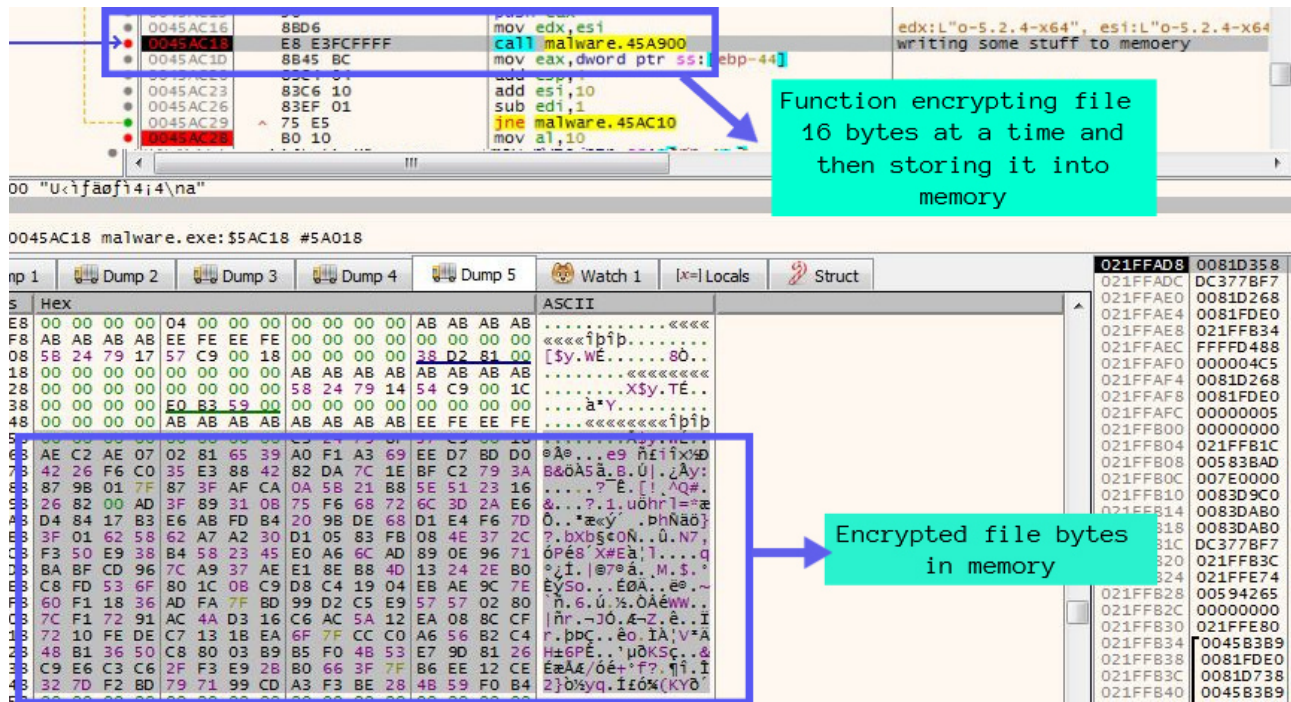


Figure 14: ApkStudio.exe.Ink File getting encrypted



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	AE	C2	AE	07	02	81	65	39	A0	F1	A3	69	EE	D7	BD	D0	ËÄ@...e9 ñfii×*@
00000010	42	26	F6	C0	35	E3	88	42	82	DA	7C	1E	BF	C2	79	3A	B&öÀ5ä`B,Ú .¿Äy:
00000020	87	9B	01	7F	87	3F	AF	CA	0A	5B	21	B8	5E	51	23	16	+>...+?`Ê.(!,^Q#.
00000030	26	82	00	AD	3F	89	31	0B	75	F6	68	72	6C	3D	2A	E6	&,..?%1.uöhr1=*æ
00000040	D4	84	17	B3	E6	AB	FD	B4	20	9B	DE	68	D1	E4	F6	7D	Ô,,.'æ«ý' >PhÑäö}
00000050	3F	01	62	58	62	A7	A2	30	D1	05	83	FB	08	4E	37	2C	?..bXb\$ç0Ñ.fû.N7,
00000060	F3	50	E9	38	B4	58	23	45	E0	A6	6C	AD	89	0E	96	71	óPé8`X#Eà;1.%.-q
00000070	BA	BF	CD	96	7C	A9	37	AE	E1	8E	B8	4D	13	24	2E	B0	°¿Í- @7@áŽ.M.\$.°
00000080	C8	FD	53	6F	80	1C	0B	C9	D8	C4	19	04	EB	AE	9C	7E	ÈýSo€..ÉøÄ..é@æ~
00000090	60	F1	18	36	AD	FA	7F	BD	99	D2	C5	E9	57	57	02	80	`ñ.6.ú.%çÒÄéWW.€
000000A0	7C	F1	72	91	AC	4A	D3	16	C6	AC	5A	12	EA	08	8C	CF	ñr`~JÓ.Æ-Z.è.€Ï
000000B0	72	10	FE	DE	C7	13	1B	EA	6F	7F	CC	C0	A6	56	B2	C4	r.pPÇ..éo.ÌÀ;V`Ä
000000C0	48	B1	36	50	C8	80	03	B9	B5	F0	4B	53	E7	9D	81	26	H±6PÉ€.²µðKSç..&
000000D0	C9	E6	C3	C6	2F	F3	E9	2B	B0	66	3F	7F	B6	EE	12	CE	ÉæÄE/óé+°f?.¶i.Î
000000E0	32	7D	F2	BD	79	71	99	CD	A3	F3	BE	28	4B	59	F0	B4	2)ð²yq²²Í£ó%(KYð'
000000F0	20	80	0D	33	40	61	D0	07	D3	F8	70	00	ED	07	02	04	@°. "HaÛ>òù ²²², .
00000100	22	C8	F4	2F	B5	6D	E7	61	CF	6D	FE	D8	E9	10	1D	04	"Èó/µmçaÏmpøé...
00000110	02	9A	4E	A4	FE	AA	D5	29	32	40	E8	A1	E9	4D	97	00	.šN²p²Ö) 2@è;ém-
00000120	E0	0D	FB	36	04	72	51	86	01	3D	08	8B	7E	B5	F6	54	à.û6.rQ†.=.<~µøT

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

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.

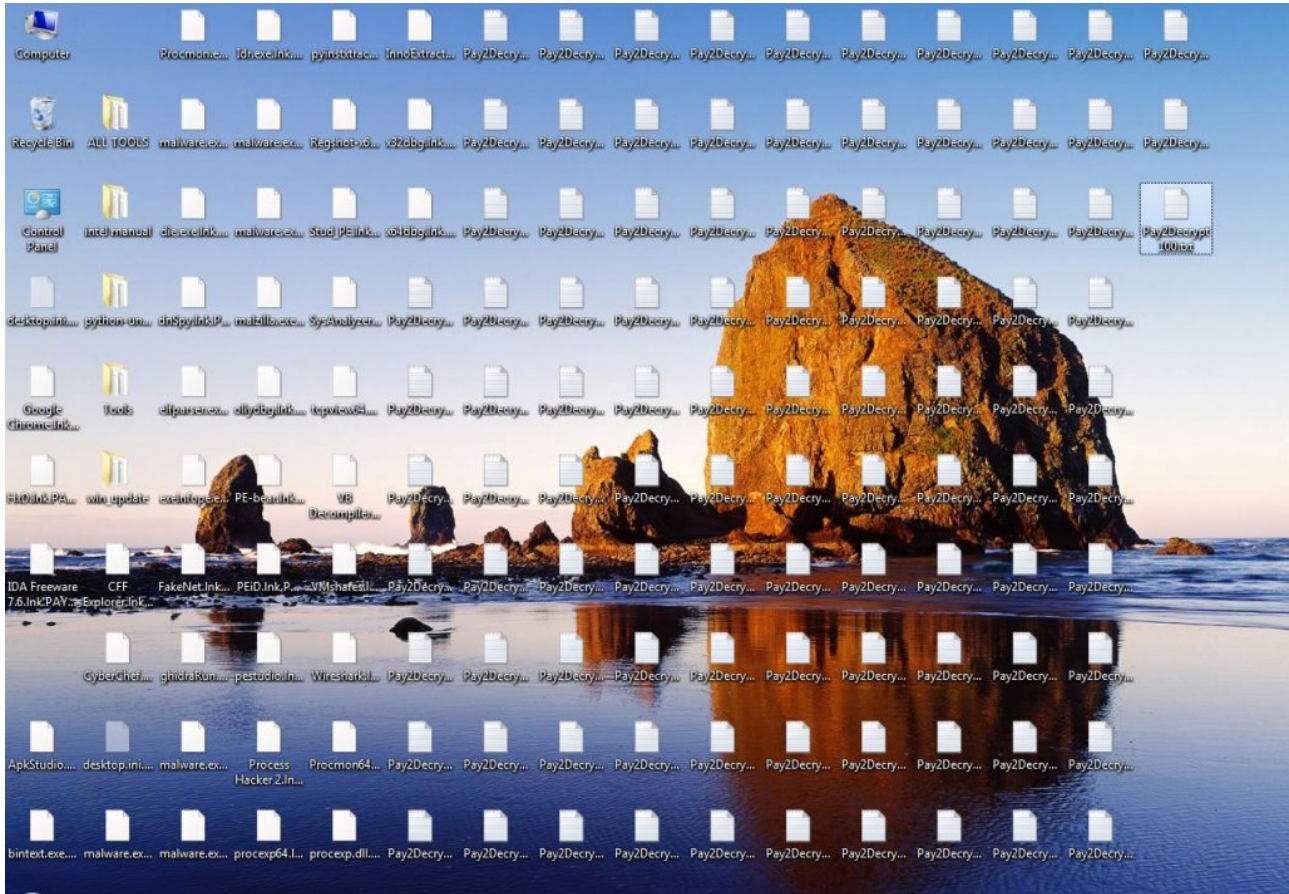


Figure 16: 100 ransom notes on Desktop

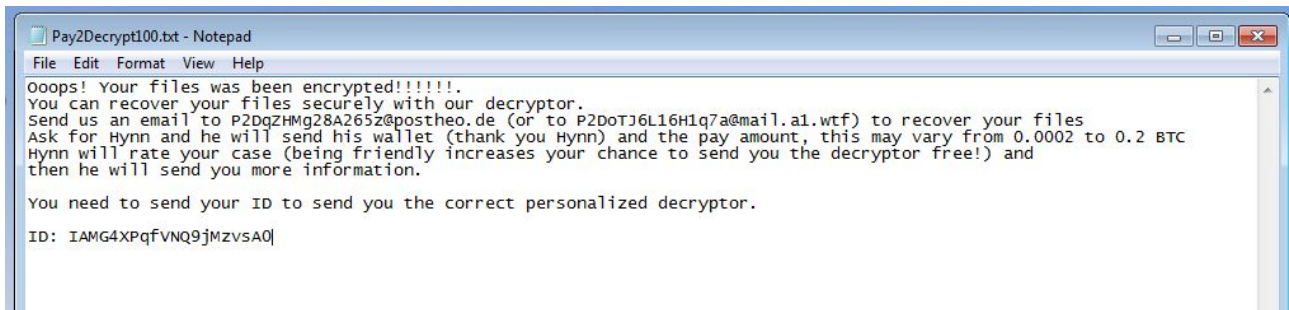


Figure 17: Ransom Note

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>