Both binaries modify their own code so EIP throws an error when debugging

* Queries some DLL’s that were suppose to come with the malware
  + WINSPOOL.DRV
  + OLEACC.dll
  + OLEACCRC.dll
  + SXS.DLL
  + CRYPTSP.dll
  + RpcRtRemote.dll
  + ieproxy.dll? looks suspicious
  + Looks at wbemcomm, rsaenh, lsa system controlset, fips algorithm
  + Queries some Microsoft Crypto stuff
  + Keeps querying stdole2 and mshtml

c3281d – – win32 kryptik malware , called movedie.exe

* queries language register key as well as a UX
* lot of RPC
* Main calls
  + CoInitialize
  + Has string (“Frontf %d dif %dQue”)
  + Gets environment variable casema with GetEnvironmentVariable
    - Environment Variable stored from offset unk\_13DB7A, lpAddress is close to our memory
  + CoUnintialize
  + Calls function that crashes
    - Inside 2nd function, Virtual Protect
      * Variable names
      * lpaddress
      * flOldProtect
    - Then function that crashes
      * Calls esi which is set by [ebp+var\_C]
      * last time is set is by eax
      * eax was set before by [ebp +arg0]
      * where arg\_0 = dword ptr 8
      * arg\_0 aka eax’s value is unk\_29B7A0
      * Call is fine, then 0xe2cd4 referenecs e61c0 which can’t be read (wellDrop.exe)
      * Priveleged Instruction (c0000096)
* Enters welldrop.exe with the esi call at the end
* Normal Program - .data section
  + first edi call
    - ntdll\_NlsAnsiCodePage
      * **Windows code pages** are sets of characters or code pages (known as character encodings in other operating systems) used in Microsoft Windows from the 1980s and 1990s. Windows code pages were gradually superseded when Unicode was implemented in Windows, although they are still supported both within Windows and other platforms.
    - RtlGetLocaleFileMappingAddress
    - kernel32\_NlsUpdateSystemLocale+C8E
      * natural language support
    - kernel32\_LoadLibraryExA
      * loads the specified module into the address space of the calling process
    - VirtualFree
  + 2nd Call Edi
    - ntdll\_NlsAnsiCodePage
    - ntdll32\_RtlGetLocaleFileMappingAddress
    - kernel32\_NlsUpdateSystemLocale +8
    - ntdll\_RTLExitUserThread
  + VirtualAlloc (Allocates section 00100000(not always this address or these permissions) RWXD)
  + Jumps into allocated section
    - UTF-16 morocco
    - call edi
      * ntdll\_NlsAnsiCodePage
      * ntdll32\_RtlGetLocaleFileMappingAddress
      * kernel32\_NlsUpdateSystemLocale
      * kernel32\_LoadLibraryExA
      * VirtualFree
    - 2nd call edi
      * kernel32\_VirtualFree
      * ntdll32\_NlsAnsiCodePage
      * ntdll32\_KiIntSystemCall(iteration)
      * ntdll\_RtlExitUserThread
    - VirtualAlloc (Allocates another section 000110000 in a RWXD section) (nothing seems to be put there)
    - VirtualAlloc malware (00210000 has RW)
    - VirtualFree the malware section (is encrypted!)
    - kernel32, kernel32\_flsfree
    - VirtualProtect on (OWNSECTION)
    - VirtualProtect on OWNSECTION again
    - VirtualProtect ( .text section of welldrop.exe)
    - VirtualProtect (another different parts of the .text section of welldrop.exe) x4
    - This is where it adds something to the malware
    - VirtualFree (00100000, RW) and frees the malware
  + Jumps to .text section of welldrop again (crahshes cause of the virtualprotect on program sections called)
    - HeapCreate
    - GetCommandLineW
    - VirtualProtect (RWD)
    - RTL Leavecriticalsection
    - RTL Acquire SRW Lock Exclusive
    - ntdll32\_fltused
    - ntdll32\_NtContinuel

FOUND PE\_malware file

* Heapcreate
* GetModuleHandleA
* GetCommandLineW
* Subfunction
  + open\_process
    - CreateEventA
      * lpEventAttributes(0) – NULL means can’t be inherited by child process
      * bManualRest(1) – if true creates a manual reset event object
      * bInitialState(0) – initial state of the event object is signaled
      * lpName(0) – name
    - GetVersion
    - GetCurrentProcessID (3880)
      * Returns value of processor calling the method
    - OpenProcess (100)
      * dwDesiredAccess – access to the process object
        + has permission to delete
        + query info
        + process\_dup\_handle,process\_vm\_write n read
        + vm\_opperation, process\_create thread, process\_terminate
      * bInheritHandle – processes created by this process will inherit the handle if True
      * dwProcessId – identifier of the local process
  + Subfunction Virtual Protect
    - Pushes date, March 11 2019
    - Subfun
      * VirtualProtect(RW, removes D) on
    - Loaded in that segment VirtualProtectChange are NTDLL library call names, suspsicious ones such as CreateProcess and LoadLibrary
  + Subfun
    - GetModuleHandleA (Kernell32.dll)
    - offset ProcName “IsWow”
    - GetProcAddress(isWow64 address)
    - calls wow64, pointer set to TRUE if the process is running under WOW64 on an Intel64 or x64 processorSubfunction
  + Subfun
    - GetModuleHandleA - ntdll
      * SEH\_pro\_log
    - lstrlenA
    - ntAllocateVirtualMemory
  + GetLongPathName
    - GetModuleFileName(26 characters)
    - Gets the longpath of our own executable,
    - lpstring2 is this path
  + GetCurrentProcessID
  + wsprintFW - Writes formatted data to the specified buffer
  + lstrlenW
  + CreateFileMappings - Creates or opens a named or unnamed file mapping object for a specified file
  + MapViewOfFile – creates a pointer to a segment with RW with content (unreadable) that we just created
  + Copies directory string of 2nd\_formatted and unmapViewOfFile
  + GetCurrentThreadId
  + GetCurrentThread
  + Subfunction
    - Mathy- maybe decodes something in a memory address allocated previously (not the data segment)
    - NTCreateSector - A section object represents a section of memory that can be shared. A process can use a section object to share parts of its memory address space (memory sections) with other processe
    - MapSector - maps a view of a section into the virtual address space of a subject process, (could make to a different process)
    - CreatesFiles
      * GetModuleHandle - NTDLL.32
      * createfile
        + Loads module, has MZ in memory
        + getting\_module\_filename – SysWOW64/ntdll.dll
        + CreateFile with same name
        + Points file to a part of ntdll.dll
        + reads a part then closes handle
      * CreateFile for LdrLoadDll
      * CreateFile for LdrGetProcedureAddress
      * CreateFile for NtProtectVirtualMemory
    - AllocateVirtualMemory
    - GetContextThread
    - Errors Method
      * Calls LDR load DLL
        + ADVAPI32.dll
        + SHLWAPI.dll
        + OLE32.dll
        + USER32.dll
        + SHELL32.dll
      * Calls LDR GetProcAddress (cycles through process addresses)
      * Calls Zw ProtectiveVirtualMemory
      * Runs through main of the program again
      * CoInitializeEx
      * Main loops is LoadLibrary followed by GetprocAddress and then using the function, in order
        + GetUserName

mal

* + - * + GetComputerName

mal-lab-win7x64

* + - * + StrToIntEx
        + New Text In Memory

wrladolph.city

rsf58.city

subaldodd.email

3245.12.10291029JSJUYNHG.10.20.0

constitution.org/usdeclar.txt

d2ca.com

ru.org

* + - * + StrToIntExA x4

Converts a string representing a decimal or hexadecimal number to an integer

* + - * + StrChrA x6
        + New Text In Memory

StrstrA

StrChrW

StrChrA

StrToIntExA

GetClipboardData

OpenClipboard

wsprintfw

CloseClipboard

ShellExecuteExW

CocreateInstance

CoSetProxy

* + - * + StrtrimA

Removes specified leading and trailing characters from a string

* + - * + Command Text in Memory

Copies some info

Pings

deletes rundll32, shell32.dll

runs powershell iex([System.Text.Encoding]::ASCII.GetString((Get-ItemProperty ‘HKCU:\%S’).%s)) Software\Microsoft\Internet Explorer\Main Check\_Associations.no

* + - * + StrChrA x4

Searches a string for the first occurrence of a character that matches the specified character

* + - * + CoCreateInstance
        + StrstrIW

Finds the first occurrence of a substring within a string

* + - * + Memset
        + push root\\default
        + wbemprox\_dll
        + cosetproxyblanket
        + **DllCanUnloadNow**
        + **CwbemDataPacket::IsValid**
        + stdregprov

manipulate system registry keys and values

**StdRegProv** is preinstalled in the WMI namespaces root\default and root\cimv2

* + - * + IE10RunOnceLastShown\_TIMESTAMP

sValueName

aSubKeyName

* + - * + IE8RunOnceLastShown\_TIMESTAMP
        + set software\\microsoft\\internetexplorer
        + Check\_Associations

<https://docs.microsoft.com/en-us/previous-versions/windows/desktop/regprov/checkaccess-method-in-class-stdregprov>

**HKEY\_CLASSES\_ROOT**

**return 0 = is successful**

* + - * + no
        + fastprox cwbemclass\_get
        + fastprox cwbemclass\_put
        + sys allocstring
        + StrstrIW
        + builds file path from /images/asdf/asf/asdf.avi
        + mbstowcs
        + CoCreateInstance
        + RPCRT4
        + IEProxy\_getProxyInfo
        + IEProxy\_getProxyDLLInfo
        + IEProxy\_DLLGetClassObject
* Launches Internet Explorer

3dd5d, generic banker malware called materialcarry.exe

* RPC
* Looks at terminal servers
* querying ddl’s
* Attempts to look at DLLSearch options in the sessino manager
* CLSID
* look at createfil dlls
* Stays alive in the task manager
  + Looks for RegEnumKeys every so often by recreating threads
    - Enumerates the subkeys of the specified open registry key. The function retrieves the name of one subkey each time it is called.
  + HKCR\TypeLib\{}
* Main calls
  + Coinitialize
  + print(Stand %d bank %d)
  + getModuleFileName
  + allmul – multiples two longs
* Has a lot more requests, behavior tab in virus total. HTTP requests and DNS resolutions as well as a process injection