MORPHISEC LAB

Evasive Malware Campaign Hits in Inbox with Faked HM Revenue and Customs Attachment

INTRODUCTION

On **December 12, 2016** Morphisec identified and monitored a new wave of sophisticated malware delivered via targeted phishing emails with malicious macro-based documents attached. The malicious documents themselves use a clever, new social engineering technique to convince the target to enable macros. Once enabled, the document calls an unknown downloader whose evasion techniques has resemblance to the Cerber downloader, but employs new obfuscation techniques.

This report analyzes the document attack chain. It is part of a series of reports produced by Morphisec Lab focusing on the most evasive and sophisticated in-memory malware.

Still Can't Evade Morphisec

The campaign recently identified by Morphisec uses several interesting modified evasive techniques and had an extremely low detection ratio on VirusTotal of only 1/ 54.

Despite the brand-new elements and variations, Morphisec's Endpoint Threat Prevention stops this sophisticated attack at its first encounter without the need of changing any rules.

virustotal

SHA256:	d526f134d2fd0bc1bf29b8c0f684a3c518a71c0f3066a85bf54fe9c9457bf39d
File name:	Statement of Liabilities.doc
Detection ratio:	1 / 54
Analysis date:	2016-12-12 14:38:27 UTC (1 day, 22 hours ago) View latest

5 0

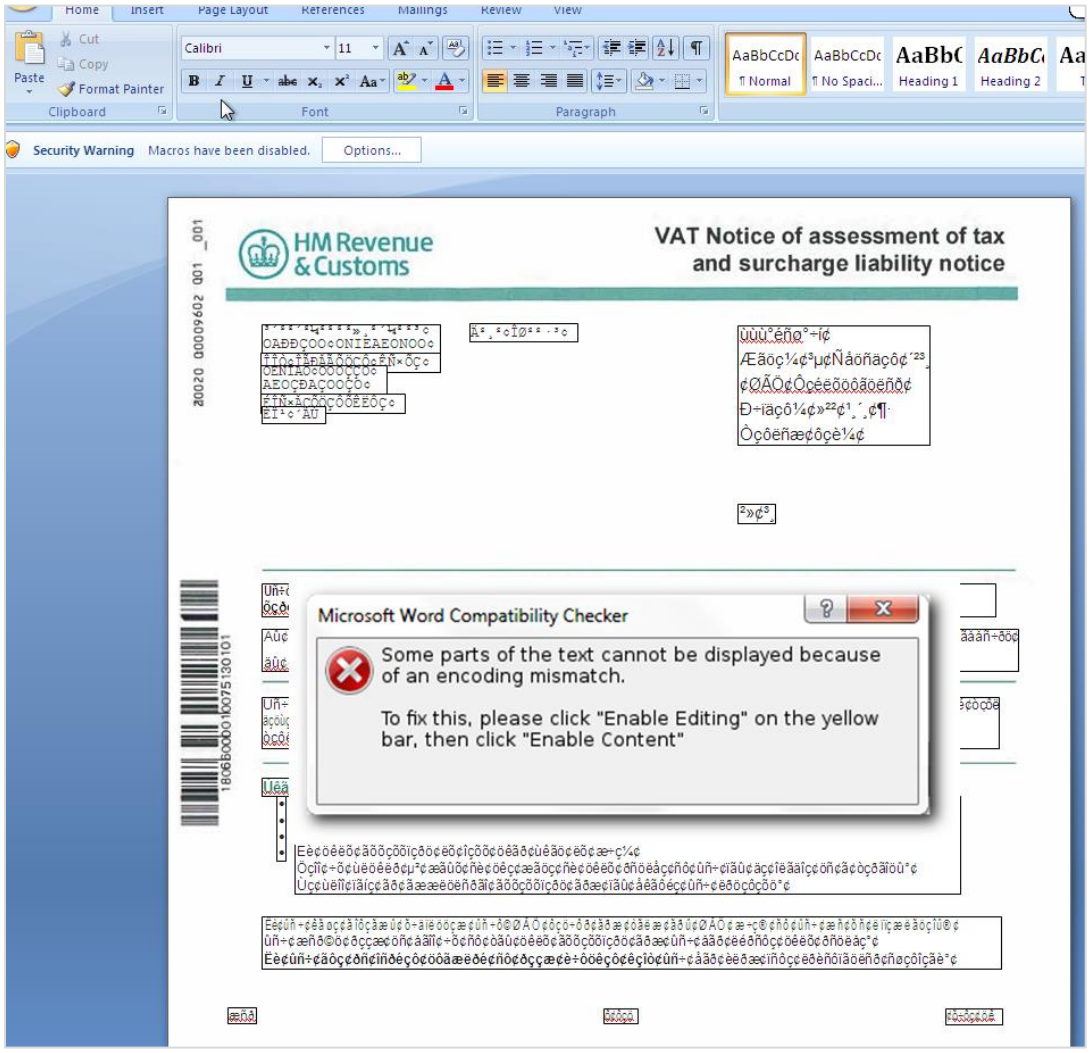
Core Characteristics

- Attempts to detect and bypass traditional defenses.
- Runs interesting EXE version downloaded from a URL and executes it
  - Resources partially taken from the AIMP audio player
- Uses Madshi (a popular productional injection and hooking library used by many security vendors – very similar to Microsoft Detours)
- Injects malicious code into explorer.exe

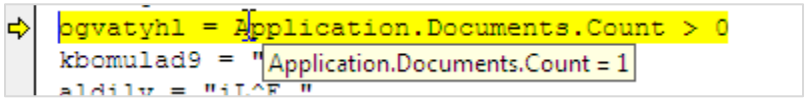
TECHNICAL ANALYSIS

1. Macro Documents

1.1. **Malicious macro WORD documents sent to the targeted victims**. Note the clever new social engineering method to enable the macro.

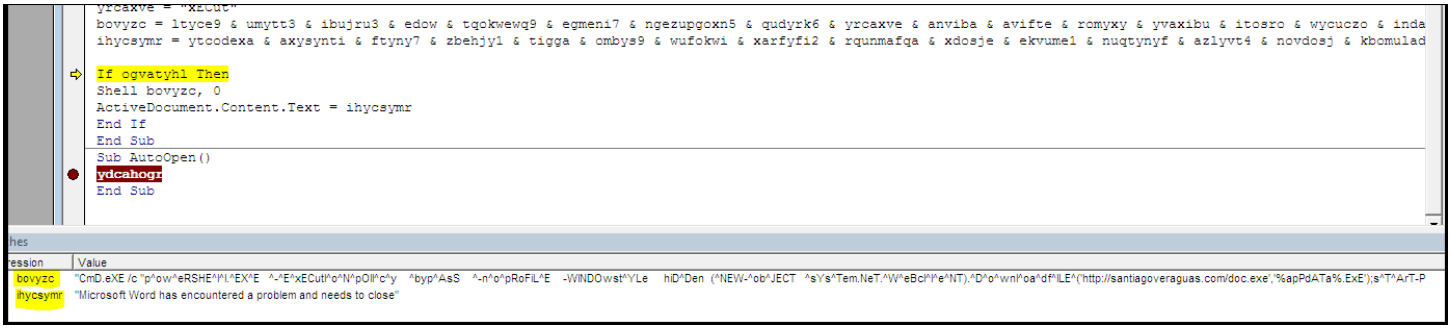


1.2. **Machine Check Anti-Sandboxing:** The malicious macro WORD document checks count of recent documents > 0 to verfy that this isn't a sandbox evirnoment.

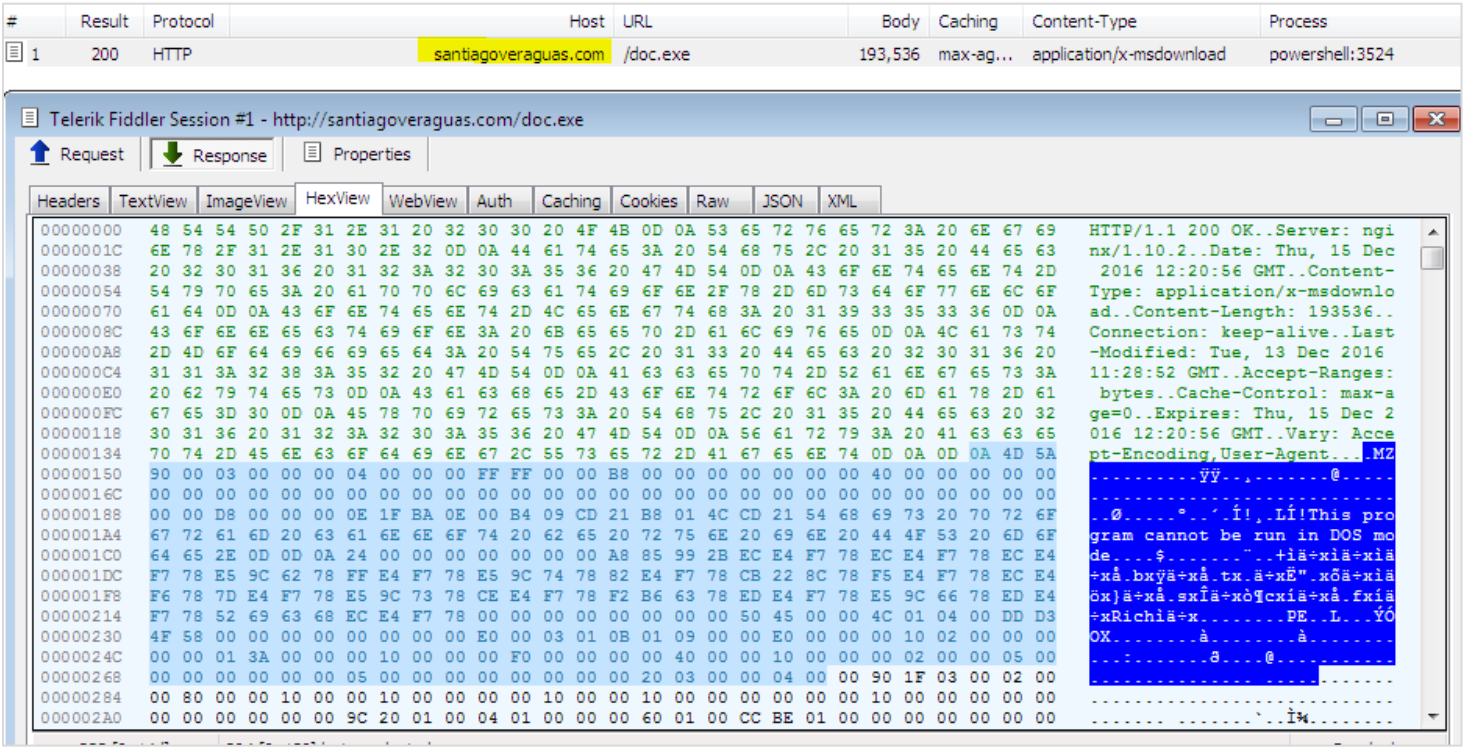


1.3. **Core capabilities in the macro**:

- First, it makes sure that it isn't in a sandbox (count of recent documents > 0)
- Second, runs the obfuscated string with special character "^" between the chars – PowerShell code downloads executable payload and runs it.
- To remove suspicious activity, the document replaces the original intro into *"Microsoft Word has encountered a problem and needs to close"*



- Finally, the downloaded executable



2. The Executable

Morphisec observed different payloads downloaded by similar documents within a very short time period. We focus here on the most interesting executable malware.

2.1. **IMPORTS: Implicit imports** reminding methods to bypass sandboxes, debugging or reversing:

- Anti-debug (Kernel32.dll):
    1. GetTickCount
    2. TerminateProcess
    3. UnhandledExceptionFilter
    4. IsDebuggerPresent
- Anonymous(Ws2_32.dll,odbc32.dll):
    1. 167
    2. 170
    3. 166
    4. 115
- Others (kernel32.dll,user32.dll,ole32.dll,secur32.dll,rpcrt4.dll):
    1. LoadLibraryW
    2. LoadLibraryA
    3. GetProcAddress
    4. CreateToohelp32Snapshot
    5. Process32First
    6. Process32Next
    7. GetEnviromentStringsW
    8. GetModuleFileNameA
    9. VirtualAlloc
    10. Sleep
    11. HeapCreate
    12. WriteFile
    13. SendMessageA
    14. CoInitialize – **use of a com objects to run a shellcode**
    15. CoCreateInstance
    16. QuearyCredentialsAttributesA
    17. EnumerateSecurityPackagesA
    18. AcquireCredentialsHandleA

2.2. **RESOURCES:**

Delphi file:

1. Type: RCData
2. Name: TAIMPOPTIONSSPAMEPLAYLISTBHHAVIOR
3. Size: 2976 byte

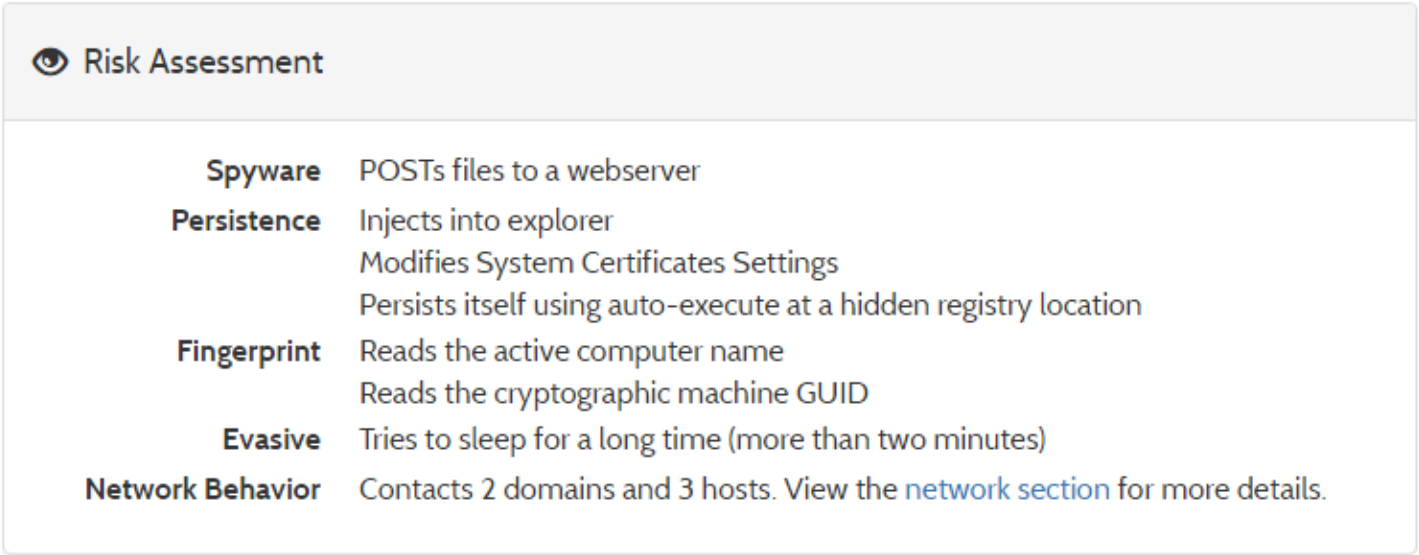
This is part of Aimp3.exe of AIMP3 developed by AIMP DevTeam. AIMP is a freeware audio and video player for windows and Android, originally developed by a Russian developer.

What makes AIMP unique is its ability to load the entire media file into the RAM of the computer.

Strings:

- GetLastActivePopup
- &Save capture
- &open
- &Exit
- &File
- Could not initialize winsock
- Unable to initialize heap
- Attempt to initialize the CRE more than once
- www.madshi.net – (open source to inject or hook)

On uploading the malicious executable to Hybrid-analysis.com, we received following risk assessment:



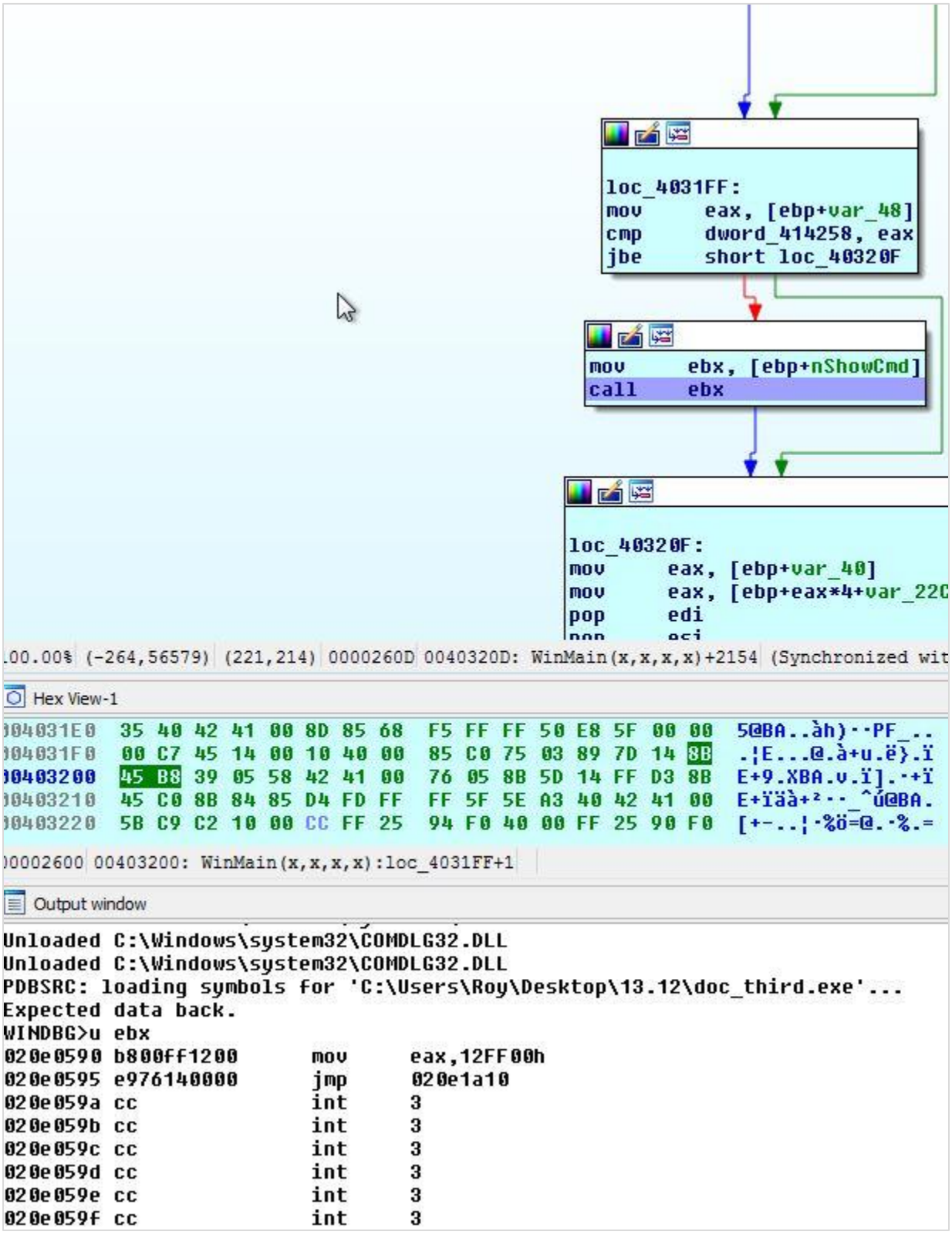
File details: Based on Kerish Products [Kerish Doctor 2015, a Windows maintenance tool]

property	value
file-type	executable
date	n/a
language	English United States
code-page	Unicode UTF-16, little endian
CompanyName	Kerish Products
FileDescription	Kerish Doctor 2015 v4.60
FileVersion	4.60.0.0
LegalCopyright	© Kerish Products
ProductName	Kerish Doctor 2015 v4.60
ProductVersion	4.60.0.0

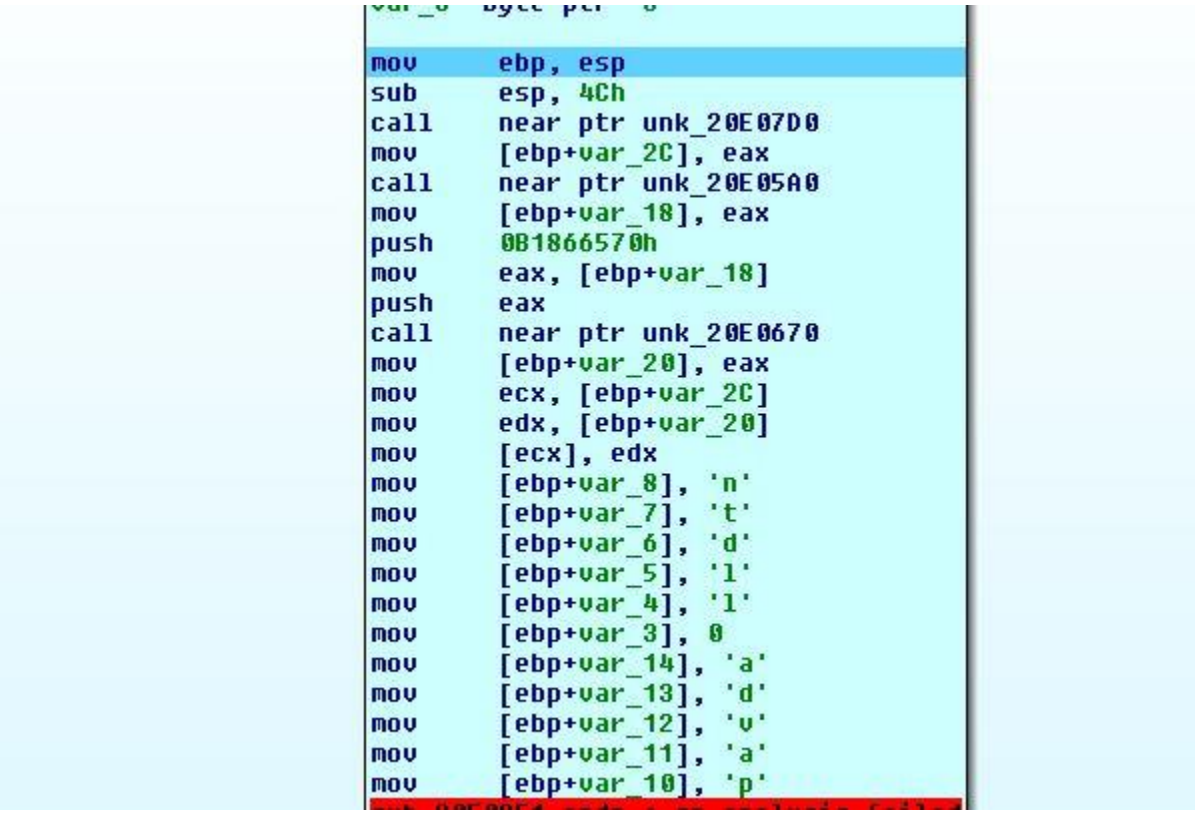
Summary of malicious and suspicious indicators taken from Hybrid-analysis.com and ThreatInfo (form December 15, 2016):

- Tries to locate where the browsers are installed
- One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.
- Performs some HTTP requests
- Allocates read-write-execute memory (usually to unpack itself)
- Creates executable files on the filesystem
- One or more of the buffers contains an embedded PE file
- **Creates an Alternate Data Stream (ADS)**
- Detects virtualization software with SCSI Disk Identifier trick(s)
- Executed a process and injected code into it, probably while unpacking
- Connects to IP addresses that are no longer responding to requests (legitimate services will remain up-and-running usually)

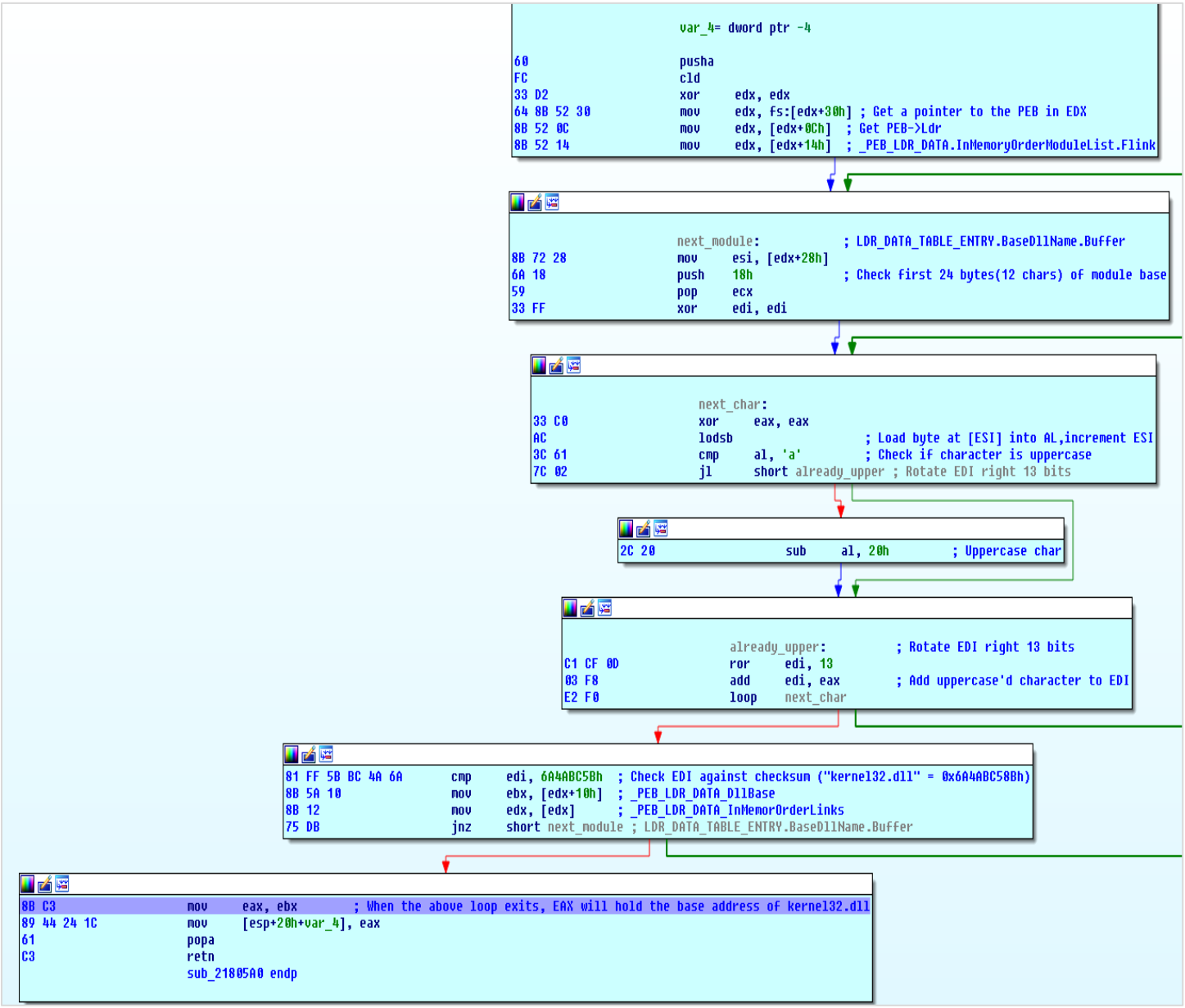
3. The Shellcode



Ndll and adavapi Libraries:



Main Sophisticated Shellcode Technique



This function uses the PEB (Process Environment Block) of the current process (stored at fs:[30h]) to locate a linked list of loaded modules. The PEB contains a member called Ldr that is a pointer to a PEB_LDR_DATA structure.

The InMemoryOrderModuleList linked list is used to enumerate the loaded modules by the order that they're loaded in memory.

The technique of enumerating the loaded modules using the PEB is commonly used by malware to locate the base address (which is also the module handle) of certain DLL files to avoid declaring and using GetModuleHandle/LoadLibrary API call implicitly.

There are a number of reasons for the attackers to apply such a bypass:

- Security solutions and generic unpackers hook calls to GetModuleHandle/LoadLibrary
- The address of GetModuleHandle/LoadLibrary is unknown unless declared implicitly - this usually happen if the code doesn't know what context it's running in, for example in **shellcode and injected threads**
- To make static analysis of the code more difficult for researcher.

CONCLUSION:

The macro-based evasion techniques used in this document have a very low detection ratio, demonstrating how most security solutions are always a few steps behind the attacker and how static based solutions cannot cope with today's rapid changes in attack scenarios. Social engineering methods are also constantly evolving, as we see in the new technique used to lure victims into activating the attack. This makes it difficult to educate users to identify phishing mail. More and more file-less or partially file-less attacks like this one are cropping up, which easily evade file-based solutions by persisting in memory.

HASHs:

Document "Statement of Liabilities.doc" –
d526f134d2fd0bc1bf29b8c0f684a3c518a71c0f3066a85bf54fe9c9457bf39d

Downloader "doc.exe"- 3128b0461d4d36a8763da1685d226fae5594364947c6bdff026eebcbde97a1cc

© Morphisec Ltd, 2016