Tracking Jupyter Malware AKA Solarmarker

By Luke Acha

Published: 2020-12-12 · Archived: 2026-04-06 00:46:47 UTC

*Updated March 10, 2022 (Detection rules for new variant observed March 2022.)

I have had the opportunity to track the .NET Backdoor, dubbed by Morphisec as Jupyter Infostealer A.K.A Solarmarker

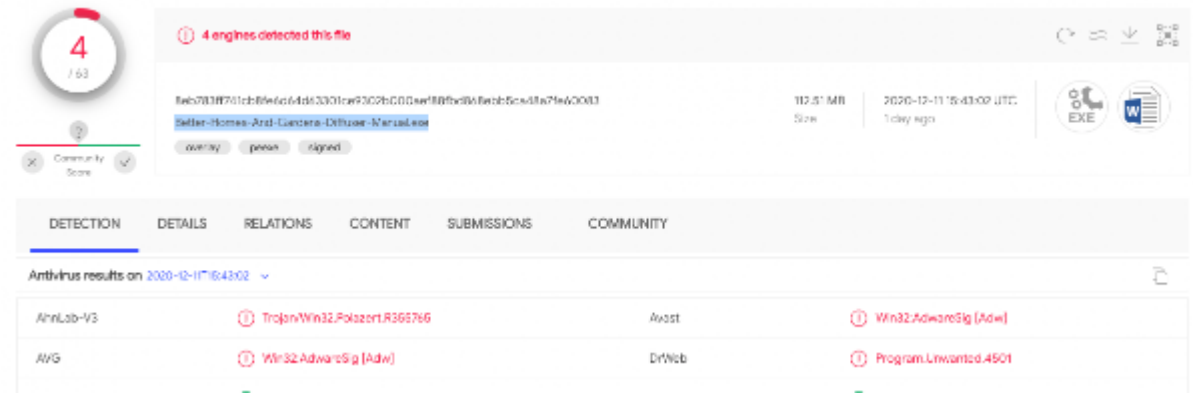
I was excited to see this writeup since this was a malware family that myself and other researchers on twitter were discussing for a couple weeks prior to the Morphisec article, before there was an attributed name to the malware. This was in October, and we were all sharing some bits of information we had on this, since that time I have also been using custom YARA signatures to perform live hunts and retro-hunts in VirusTotal to continue to keep up on this malware.

Recently I had seen Red Canary wrote up about this, dubbing it Yellow Cockatoo. Again, I was very excited to see some more attention being paid to this malware, I enjoyed both the writeups. Red Canary and Morphisec provided excellent information!

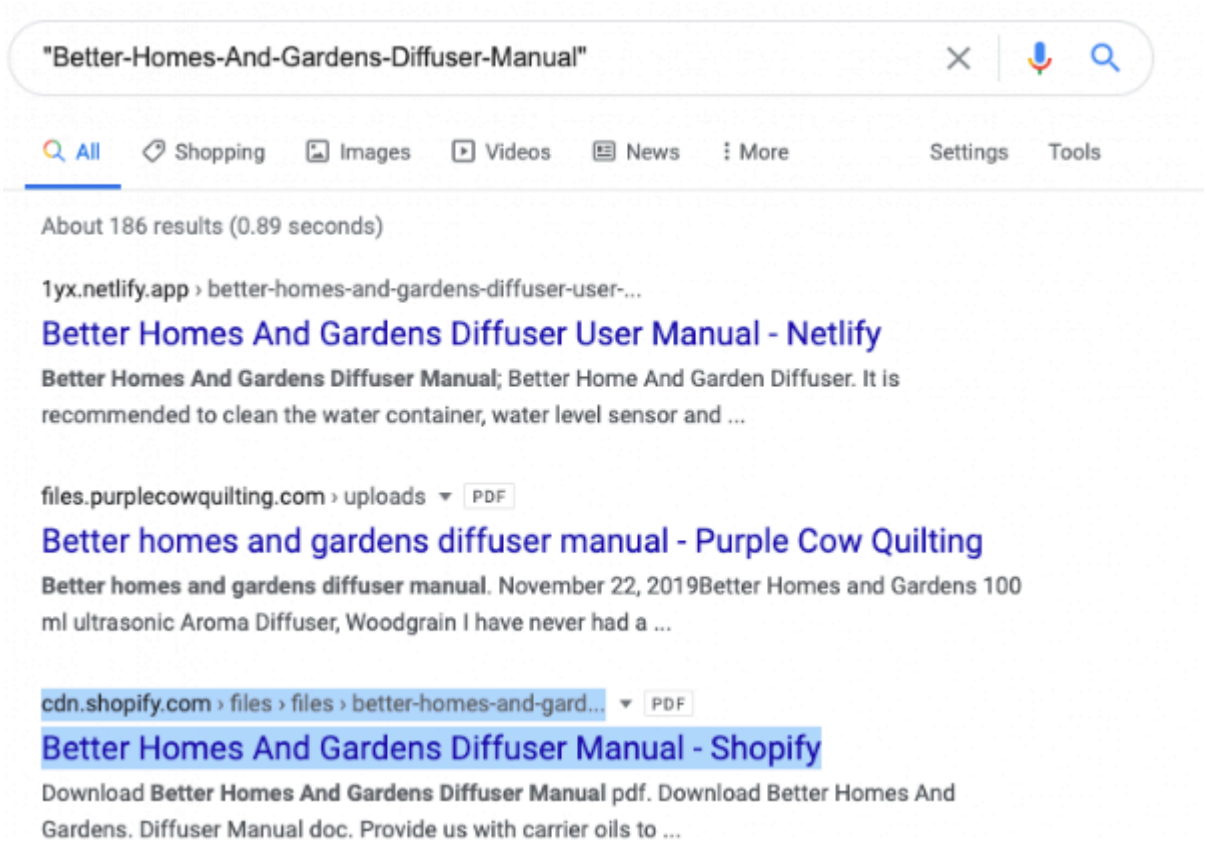
Since I've been tracking this for sometime, and commenting on all new samples I see uploaded to VirusTotal, I figured I would provide some perspective that I have on this malware as well.

First, the initial access. Red Canary does correctly point out that there is redirecting of search engine queries, to dig a bit deeper, it appears that this is being done by abusing legit sites such as sites.google.com and cdn.shopify.com.

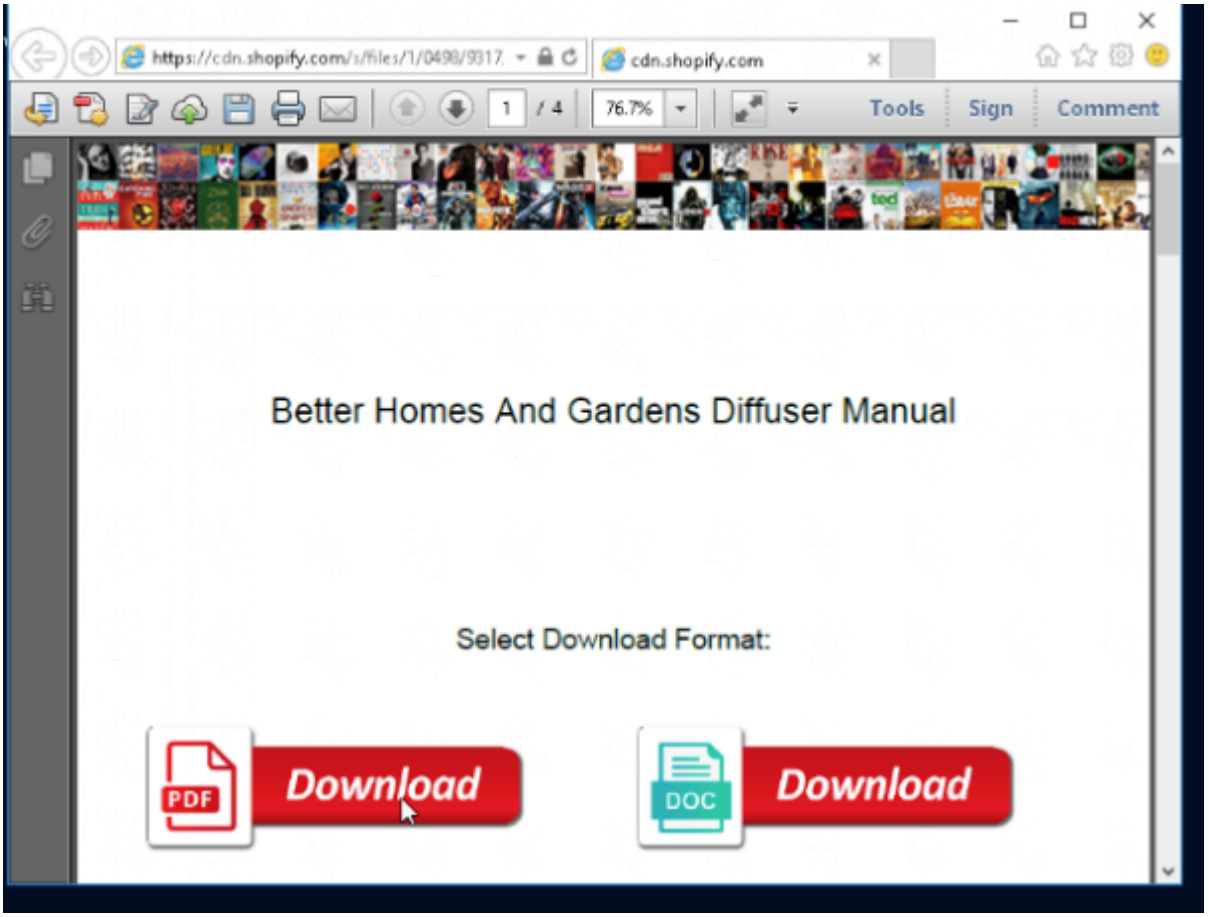
The following image is a recent (as of this writing) sample uploaded to VirusTotal. Take note of 3 things. The first being the file name, the second being the file size, and finally the 3rd item being the Icon which appears to mimic a Word Document.



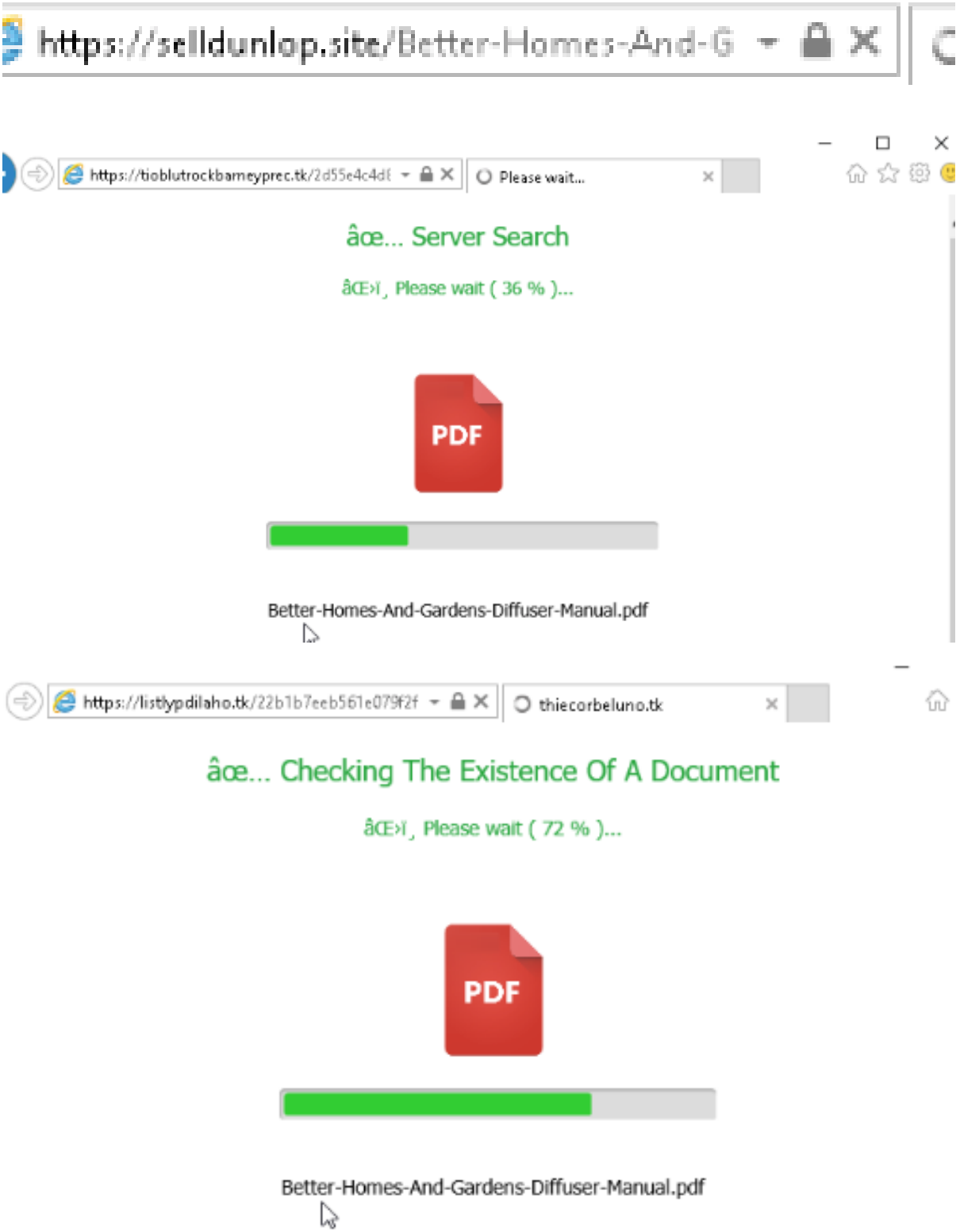
In the next screenshot we can see potentially where a users search criteria may lead to this malware.

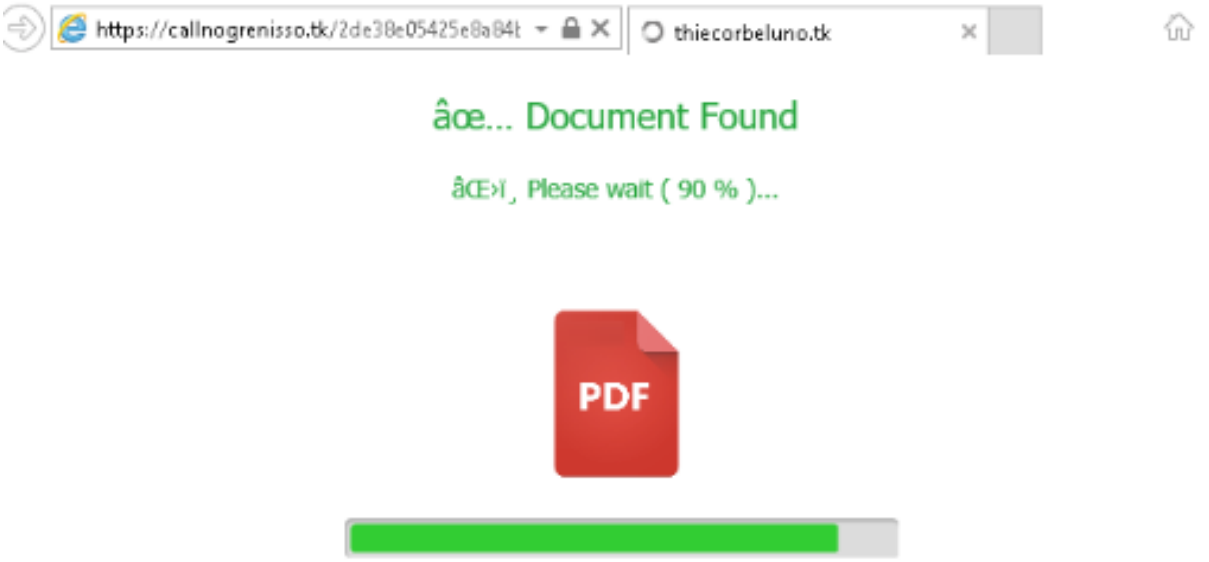


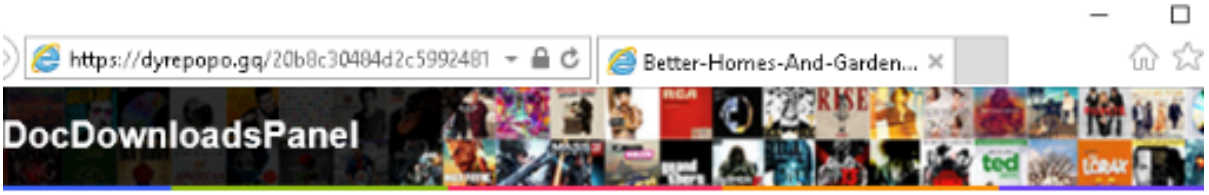
Note: I have seen this also on various sites.google.com pages as well with earlier samples. So, what happens when we go to this link which may potentially lead to the malware?



Now this is interesting right! OK, so when I click on the PDF download, I watched the Address Bar redirect several times until I was able to get the final malware. Look at the following screenshots!


https://selldunlop.site/Better-Homes-And-G
https://tioblutrockbarneyprec.tk/2d55e4c4d
Please wait...
âœ... Server Search
âŒ›¸ Please wait (36 %)...
PDF
Better-Homes-And-Gardens-Diffuser-Manual.pdf
https://listlypdilaho.tk/22b1b7eeb561e079f2f
thiecorbeluno.tk
âœ... Checking The Existence Of A Document
âŒ›¸ Please wait (72 %)...
PDF
Better-Homes-And-Gardens-Diffuser-Manual.pdf



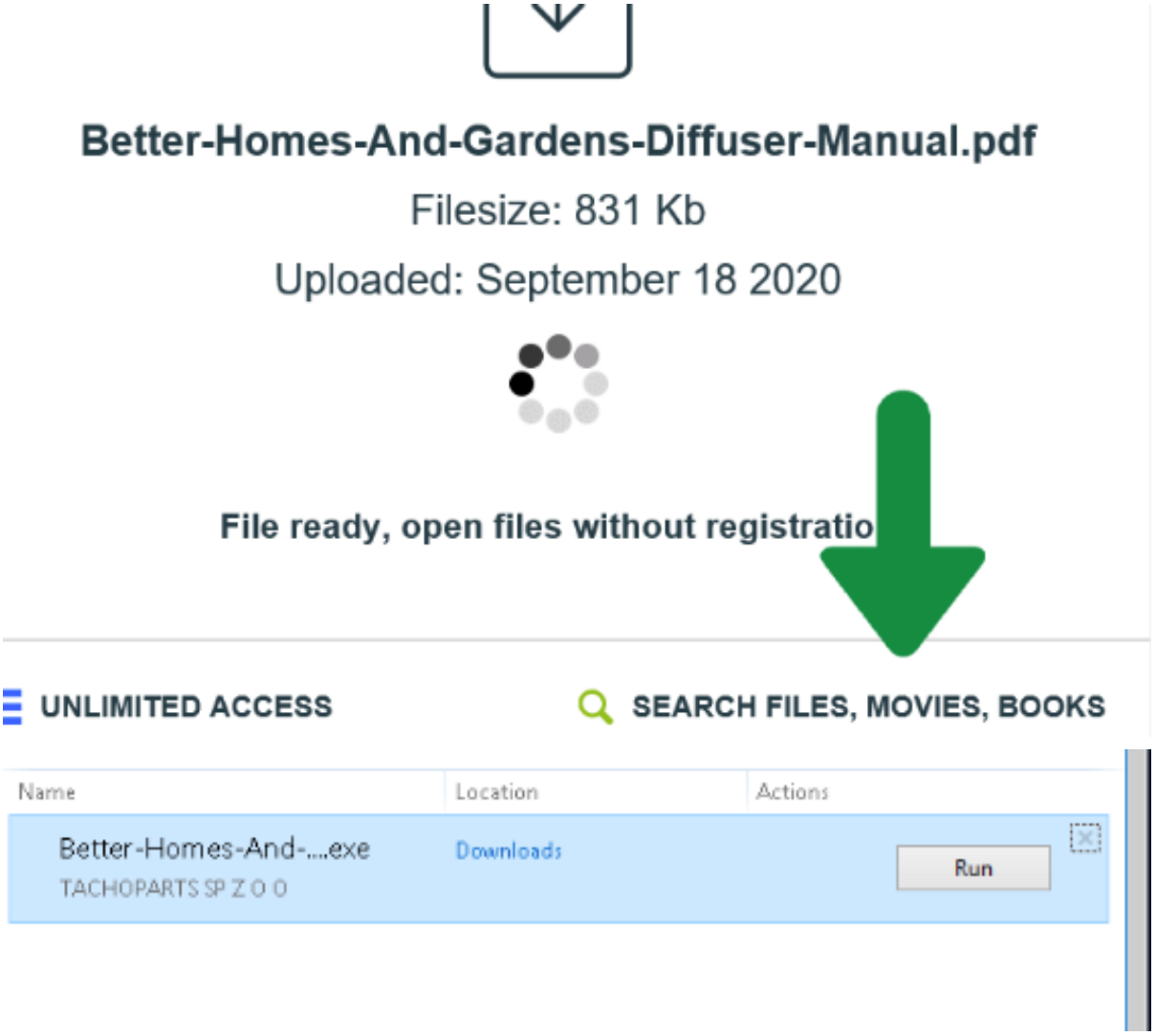


Better-Homes-And-Gardens-Diffuser-Manual.pdf

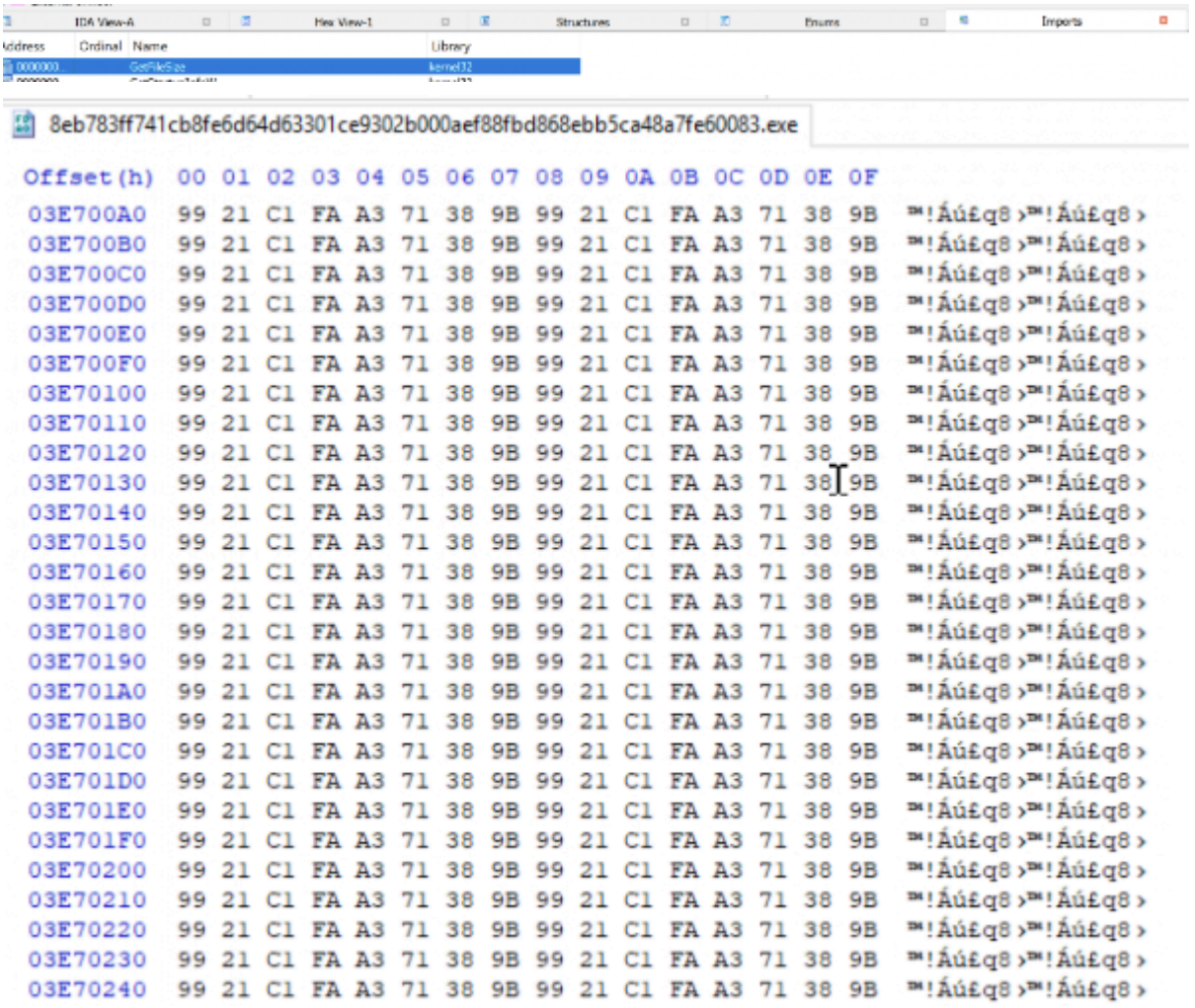
Filesize: 831 Kb

Uploaded: September 18 2020

OPEN DOCUMENT

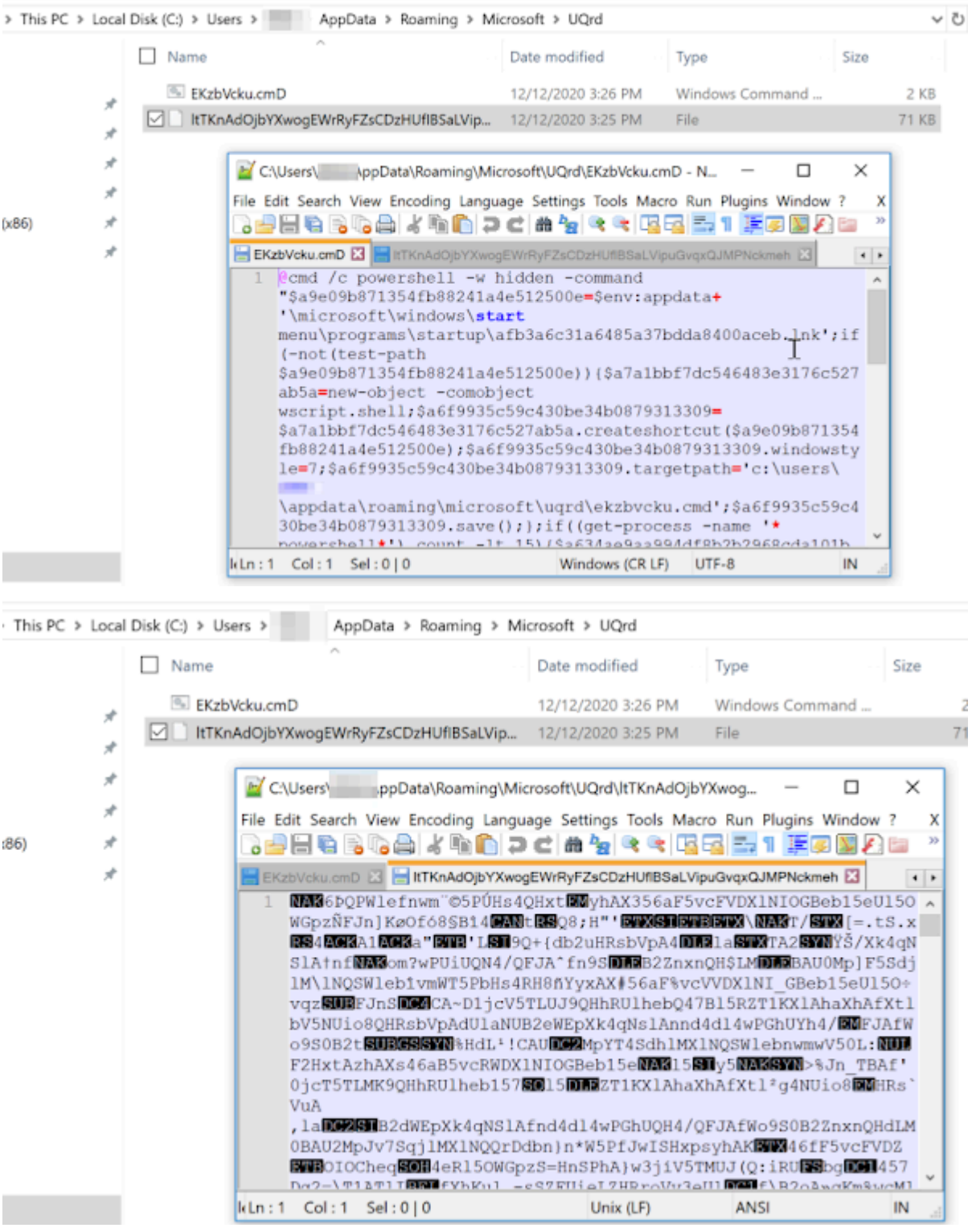


What is really interesting is how quickly this 100MB+ file actually downloads! Why is this you ask? It's because the file appears to be heavily padded (older samples were padded with NULL Bytes, this latest one is padded with repeating garbage bytes 99 21 C1 FA A3 71 38 9B). Even more interesting is that the malware seems to perform a filesize check, so that if an analyst attempts to alter the size the malware errors out. If I remove even 1 byte, it errors, if I add even 1 byte, it errors.... but... if I just flip a bunch of the NULL Bytes and the file size remains the same, it works fine. Below are acouple interesting screenshots of the padding and the import of GetFileSize which might be used to see if the file was altered (ie. padding removed). **NOTE: The April 2021 variant appears to have dropped much of the padding, file sizes are now 16-17MB**



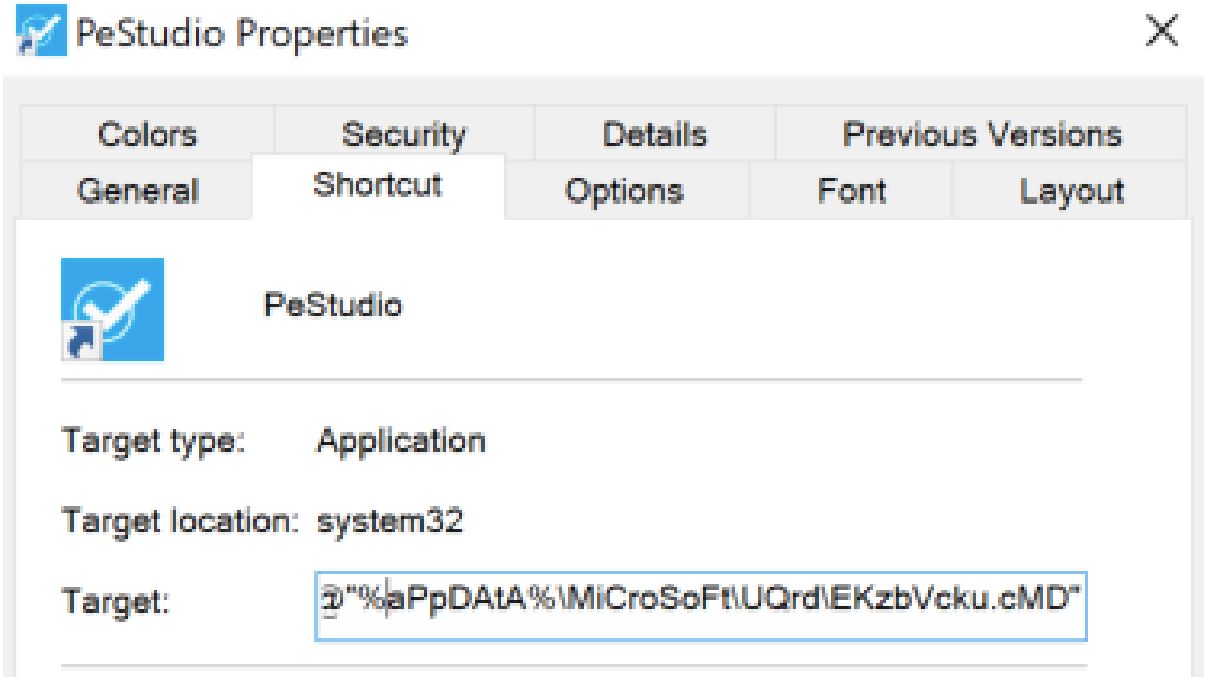
At this point, I feel we have a good sense of the initial vector for this. We know a lot based on the Red Canary and Morphisec write-ups, we know that the malware drops and launches a legit program as a red herring in this case Soda PDF, we also know that it drops 2 .txt files in appdata\local\temp, which are really powershell files. The first one decodes the second one, then they delete themselves.

This is because they create persistence in the form of a .cmd file (which launches powershell) it also drops a larger file which is a heavily encoded file that is decoded from the .cmd file.



Two other interesting things happen during all of this.... 1, the powershell process connects to the C2 per the loaded DLL and even more interesting is that is modifies existing desktop LNK files (shortcuts), keeping the original launch string and then adds an operator to also launch the .CMD file!

#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	C
12	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
13	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
14	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
15	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
16	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
17	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
18	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
19	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
20	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
21	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
22	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
23	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
24	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
25	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
26	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
27	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
28	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	
29	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:2892	
30	502	HTTP	91.241.19.110	/	512	no-cac...	text/html; c...	powershell:5332	

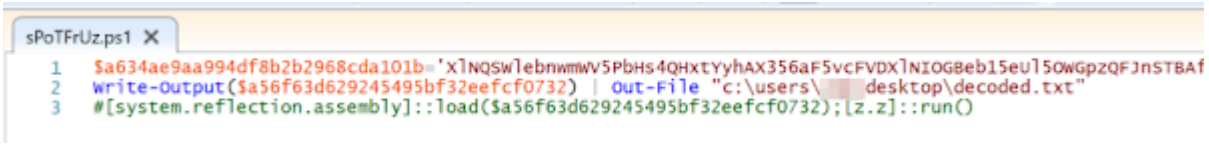


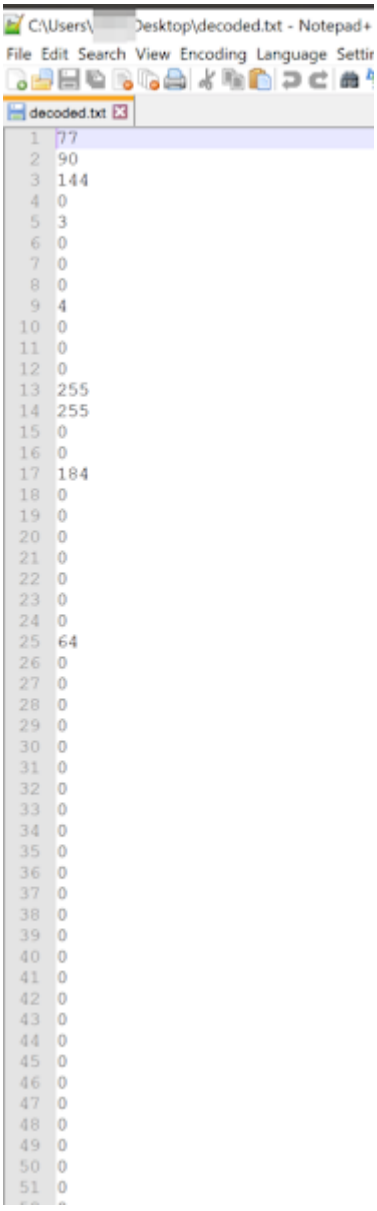
That's Right! My PEStudio still launches.... along with the malware!

OK, so for a little bit of fun, I'll quickly go over how you can easily decode this malware and extract the malicious DLL.

First, we modify the .CMD file a bit, comment out or remove the [system.reflection.....] line, and remove everything before (and including) the bracket {. Save the file as .ps1 for ease.

Next, we put in a line "Write-Output(Variable being loaded from system.reflection) | Out-File "c:\....."





At this point, this is simple encoding. I've been using CyberChef "From CharCode", Delimiter "Line Feed" "Base 10" to quickly get the DLL at this point.

Hopefully this helps provide some additional details on this malware, below are the IOCs for this specific example. Happy Hunting!

IOCs From initial writing:
Initial Executable: da2eb36e763ecf1a47532e9f8efeacb7
Malicious DLL: 147666fdb5f64f46a0a0add2cc428ec8
C2: 91.241.19[.]110
Observed Redirect Domains:
dyrepopo[.]gq
feedsterbomiditsign[.]tk
listlypdilaho[.]tk
callnogrenisso[.]tk
selldunlop[.]site
spherdoorgfinversbrookin[.]tk
tioblutrockbarneyprec[.]tk
thiecorbeluno[.]tk

VT Enterprise Hunting Tactics

Icon Hash searching:

PDF page that holds embeeded links to series of redirects: **main_icon_dhash:94148c3333001100**
PDF page that holds embeeded links to series of redirects: **main_icon_dhash:94228c3333001100**
PDF page that holds embeeded links to series of redirects: **main_icon_dhash:0f0f0307332f3f19**
Fake Word Document Icon Hash for dropper file: **main_icon_dhash:64dcd4d2c4c4d0d4**
Fake PDF Document Icon Hash for dropper file: **main_icon_dhash:b2b29696969ef66a size:100MB+**
Fake PDF Document Icon Hash for dropper April 2021: **main_icon_dhash:b2b29696969ef66a**
Fake PDF Document Icon Hash for dropper September 2021: **main_icon_dhash:64e4d4d4e8f4dcd4**
Fake PDF Document Icon Hash for dropper May 2023:**main_icon_dhash:74e4d4d4ecf4d4d4**
March 2022 VirusTotal search for Dropper:**entity:file tag:signed type:peexe size:250MB+ size:270MB- packer:".NET executable"**

YARA Rules:

SolarMarker March 2022 Malicious DLL Detection

SolarMarker 2021 DLL Detection

Suspicious_Powershell_Strings

OpenIOC Rules:

Solarmarker.dat File Creation (OpenIOC)

Suspicious_Porcesses_Writing_to_Startup (OpenIOC)

SIGMA Rules:

Solarmarker.dat File Creation (SIGMA)

Suspicious_Porcesses_Writing_to_Startup (SIGMA)

Updates:

Pulled one of the initial files that gets deleted after running, along with the powershell script that decodes and runs it:

https://app.any.run/tasks/fcd6eeb7-91bb-4e1d-b02d-983bae3786ec#

March 2022 App.Any.Run sandbox run/

Example Observed Lures from google searches:

site:byzcath[.]org "free template"

http://byzcath[.]org/nfl-playoff-bracket-excel-spreadsheet

site:www.braveheartmarine[.]com "free template"
https://www.braveheartmarine[.]com/free-invoice-template-for-handyman

site:prismic-io.s3.amazonaws[.]com "free template"
https://prismic-io.s3.amazonaws[.]com/whatsimdb/0fe19bd3-88a8-4ab5-b451-d78f1be51ef2_free-bbq-tickets-template-word.pdf

site:cdn.shopify[.]com "free template"

site:healingwithclarity.com "free template"
https://healingwithclarity[.]com/platte-county-warrant-list.pdf

site:strikinglycdn.com "free template"
https://uploads.strikinglycdn[.]com/files/18aa0685-0e17-4ea4-b308-1a717e293267/free-template-for-waiver-of-liability.pdf

---