**Hunting with ELK (Lab 1)**

**LAB 16.1**

**Scenario**

The IT Security manager has asked your internal Penetration team to generate malicious PowerShell traffic in the environment and has now tasked you, the only Threat hunter, to create detection rules for potentially malicious usage of PowerShell. He has directly tasked you to ensure that your rules detect their commands, where with additional research, he expects you for all to take the detection rules a step further by ensuring that they expand the range of detection for other variations that would match the commands executed by the Penetration team (where possible).

Goals

The learning objective of this lab is the rule generation ability.

What you will learn

You will learn how to utilize available logs to detect PowerShell based activity on a windows host.

Recommended tools

ELK

Introduction To ELK

Elastic's ELK is an open source stack that consists of three applications (Elasticsearch, Logstash and Kibana) working in synergy to provide users with end-to-end search and visualization capabilities to analyze and investigate log file sources in real time.

ELK's architecture, at a high level, is the following.

beats 
Data 
Collection 
logstash 
Data 
Aggregation 
& Processing 
elasticsearch 
Indexing & 
storage 
kibana 
Analysis & 
visualization 

On demanding/data-heavy environments, ELK's architecture can be reinforced by Kafka, RabbitMQ and Redis for buffering and resilience and by ngnix for security.

redis 
kafka 
logstash 
beats 
Data 
Collection 
elasticsearch kibana 
NGMX 
bRabbitMO 
Buffering 
Data 
Aggregation 
& Processing 
Indexing & 
storage 
Analysis & 
visualization 

**Let's dive into all of ELK's components.**

**Elasticsearch** is a NoSQL database based on the Lucene search engine and built with RESTful APIs. It is essentially the index, store and query application of the ELK stack. It provides users with the capability to perform advanced queries and analytics operations against the log file records processed by Logstash.

**Logstash**is the tool responsible for the collection, transformation and transport of log file records. The great thing about Logstash is that it can unify data from disparate sources and also normalize them. Logstash has three areas of function.

Process input of the log file records from remote locations into a machine understandable format. Logstash can receive records through a variety of ways (https://www.elastic.co/guide/en/logstash/current/input-plugins.html) such as reading from a flat file, reading events from a TCP socket or directly reading syslog messages. When Logstash completes processing input it proceeds to the next function.

Transform and enrich log records. Logstash provides users with numerous methods to make changes to the format (and even content) of a log record. Specifically, filter plugins exist that can perform intermediary processing on an event (most of the times based on a predefined condition). Once a log record is transformed Logstash processes it.

Send log records to Elasticsearch by utilizing any of the output plugins.

**Kibana** is the tool used for visualizing the Elasticsearch documents. Through Kibana users can view the data stored in Elasticsearch and perform queries against them. It also facilitates the understanding of query results through tables, charts and custom dashboards.

Note: Beats is an additional download that should be installed in every remote location for its logs to be shipped to the Logstash component.

**ELK's Search:**

As threat hunters, chances are that we will spend the majority of our ELK-time inside Kibana. For this reason, we will focus on submitting searches through Kibana.

K kibana 
Index 
Search Bar 
Pattern 
17th 
New 
Save 
Shr. 
C Auto•refresh 
Auto 
Time 
Picker 
st 'O days 
Histogram 
Document 
Table 
Side 
Navigation 
• 27th 20" — 
12 
ntials the 
, • es my user to 
be a user' s s 
Syste in the Of trust bee 
n by the Certificate. Certificate) 
for TLS'SSL a user to b 
Of CM They t 
Of need to t'. of 
t s the to to the 

Kibana searches are usually formatted as

FieldName:SearchTerm

. Fields and search terms are case sensitive.

Boolean operators like AND, OR are supported (and are sometimes implied).

Wildcards and free text searches can be used, but use sparingly.

**Access the Kibana at Web (**[**http://demo.ine.local:5601**](http://demo.ine.local:5601/)**)**

**Tasks**

**Task 1. Perform a hunt for well-known PowerShell Offensive Frameworks and commands**

Search for technical articles/posts/threat intelligence feeds on PowerShell abuse and define an ELK query which detects known PowerShell-based attacking frameworks or scripts/commands. For this task focus only in the available PowerShell logs.

Hint: Pay attention to the ScriptBlockText field of event ID 4104.

**Slolutions**

The information we are interested in is contained in the **ScriptBlockText field of event ID 4104.** Therefore, to begin with, we set a filter in ELK for event ID 4104. We then proceed into constructing the following query after performing a search on popular PowerShell Frameworks/Commands:

|  |  |
| --- | --- |
| winlog.event\_data.ScriptBlockText:(PowerUp OR Mimikatz OR NinjaCopy OR Get-ModifiablePath OR AllChecks OR AmsiBypass OR PsUACme OR Invoke-DLLInjection OR Invoke-ReflectivePEInjection OR Invoke-Shellcode OR Get-GPPPassword OR Get-Keystrokes OR Get-TimedScreenshot OR PowerView) | PowerShell Frameworks/Commands |

**Screenshots**

lew Save Open Share Inspect 
Filters winlog.event_data.ScriptBlockText:(PowerUp OR Mimikatz OR NinjaCopy OR Get-ModifiablePath OR AllChecks OR AmsiBypassOR Invoke-DLLlnkection OR Invol 
04 
2012-01-01 
-01-01 
KQL 
Last 15 years 
Auto 
2018-01-01 
Show dates 
Update 
+ Add filter 
logstash-2020.02.02 
;elected fields 
_source 
Xvailable fields 
'opular 
winlog.event_data.Script810ckText 
winlog.event_id 
@timestamp 
@version 
_index 
_score 
_ type 
agent.ephemeral_id 
agent.hostname 
2010-01-01 
Jan 16, 2008 @ 19:2343.492 - Jan 16, 2023 @ 192343.492 — 
2014 
2016-01-01 
@timestamp per 30 days 
2020-01-01 
2022-01-01 
Time 
Feb 2, 
2020 
winlog. event_data. Script810ckText: Get-TimedScreenshot 
log. file.path: log. level: verbose winlog.opcode: On create calls 
winlog.activlty_id: {cB0b7b33-d9b0-eee3-88b6-ebcBb0dgd5B1} winlog.provider_name: Microsoft-Windows-PowerShe11 winlog.record_id: 36, 820 winlog.process.pld: 3, 596 
.process.thread.id: 4,168 winlog.user.identifier: s-1-5-21-2551594947-389063BB63-1692720143-1BB2 winlog.provider_guid: {aBc1853b-5c4a-4b15-8766-3cf1c58f985a} 
win log 
winlog.computer_name: "INIO winlog.event_data.Script810ckId: c8f81317-3ff8-466g-bd26-d4516B84363d winlog.event_data.MessageNumber: 1 winlog.event_data.MessageTota1: 
winloq.channel: Microsoft-Windows-PowerShe11/0perationa1 winloq.task: Execute a Remote Command winlog .api: wineventlog winloq.version: 1 winloq.event_id: 4, 1ß4 
View surrounding documents View single document 
Expanded document 
Table JSON 

Time - 
Feb 2, 
202B @ 
_source 
winlog.event_data.ScriptB10ckText: Get-TimedScreenshot message: Creating Scriptblock text (1 of 1): Get-TimedScreenshot Script810ck ID: c8f81317-3ff8-4669-bd26- 
d4516084363d Path: log. file.path: log. level: verbose winlog.opcode: On create calls winlog.activity_id: {ceeb7b33- 
d9b0-eee3-88b6-ebcBb0dgd5e1} winlog.provider_name: Microsoft-Windows-PowerShe11 winlog. record_id: 36, 820 winlog.process.pid: 3, 596 winlog.process.thread.id: 4, 168 
winlog.user .identlfier: S-1 -5-21-2551594947-3890630063-1692720143-1002 winlog.provider_guid: {aeci853b-5c4B-4b15 
-8766-3cf1 c58f985a} winlog. computer_name: WINIB 
winloq.event_data.ScriptB10ckId: c8f81317-3ff8-4669-bd26-d4516B84363d winloq.event_data.MessaqeNumber: 1 winloq.event_data.MessaqeTota1: 1 winloq.channel: Microsoft- 

**Task 2.**Perform a hunt for suspicious parent process spawning PowerShell

Research on suspicious parent processes that may spawn PowerShell and define a rule which detects the usage of a malicious command in the available PowerShell logs.

**Hints:**

Focus on Sysmon's Process creation events (event ID 1). A lot of "everyday" processes have the capability to spawn PowerShell, for example, the entire MS Office Suite etc.

* <https://www.varonis.com/blog/living-of-the-land-lol-with-microsoft-tools-part-i-intro-to-regsvr/>
* <https://www.carbonblack.com/2016/04/28/threat-advisory-squiblydoo-continues-trend-of-attackers-using-native-os-tools-to-live-off-the-land/>

**Solutions**

For this task, we'll look into Sysmon's Process creation events, **event id 1** so we start by filtering out for those. Then we proceed with our research for identifying suspicious parent processes. To construct the following query with the data researched, we should remember that it will only be valid if it spawns a process "powershell.exe" (let's omit the fact that a renamed PowerShell could be spawned for now):

|  |  |
| --- | --- |
| **winlog.event\_data.ParentImage:(\*mshta.exe OR \*rundll32.exe OR \*regsvr32.exe OR \*services.exe OR \*winword.exe OR \*wmiprvse.exe OR \*powerpnt.exe OR \*excel.exe OR \*msaccess.exe OR \*mpub.exe OR \*visio.exe OR \*outlook.exe OR \*chrome.exe OR \*iexplorer.exe OR \*sqlserver.exe) AND winlog.event\_data.Image : \*powershell.exe** | identifying suspicious parent processes. |

**Screenshot**

If you expand the first match and look at the parent process's command line argument, you'll notice the following which indeed looks suspicious:

Regsvr32 using a well-known command execution to spawn PowerShell

New Save Open Share Inspect 
Filters winlog.event_data.Parentlmage : ('mshta.exe,C rund1132.exe, OR •Regsvr32.exe, OR •services.exe, OR •winword.exe, OR • wmiprvse.exe, OR •services. 
@ winlog.event_data.parentlmage: x + Add filter 
04 
2012-01-01 
KQL 
Last 15 years 
Auto 
2018-01-01 
Show dates 
logstash-2020.02.02 
Selected fields 
? _source 
Available fields 
popular 
winlog event_data. ScriptBlockText 
winlog.event_id 
@timestamp 
@version 
_ index 
-score 
t _ type 
2010-01-01 
Jan 16, 2008 @ - Jan 16, 2023 @ — 
2014-01-01 
2016-01-01 
@timestamp per 30 days 
2020-01-01 
C Refresh 
2022-01-01 
Time 
Feb 2, 2020 .563 
winlog.event_data.Origina1FileName: powerShe11.EXE winlog.event_data.parentlmage: winlog.event_data.ComandLine: 
WindowspowerShe11\v1.mpowershe11.exe• winlog.event_data.lnage: .mpowershell.exe message: process Create: UtcTime: processGuid: 
2020-02-02 processld: Image: 5344 Fileversion: Description: 10.0.18362.1 (HinBui1d.1601Ø1 .egee) 
product: Windows PowerShe11 Cæpany: Microsoft• Windows• Operating System ComandLine: Microsoft Corporation CurrentDirectory: powerShe11.EXE User: 
LoqonGuid: Loqonld: exe Terminalsessionld: InteqrityLeve1: øx1105bf Hashes: 1 parentprocessGuid: High 

winLog. convuter_nane 
. event _data. ComandL ine 
t 
t 
t 
. event _data. Description 
t 
. event _data. FileVersion 
t 
t 
t 
t 
t 
t 
t 
t 
. event _ data. Parentlmage 
t 
t 
t 
winlog 
winlog 
winlog 
winlog 
winlog 
wi nlog 
wi nlog 
wi nlog 
wi nlog 
wi nlog 
nlog 
nlog 
nlog 
nlog 
nlog 
. event _data. Company 
. event _data. CurrentDirectory 
. event_data. Hashes 
. event_data. Image 
. event _data. IntegrityLevel 
. event _data. LogonGuid 
. event _data. Logonld 
. event _ data. Or iginalFileName 
. event _data. ParentComandLine 
. event _ data. parentProcessGuid 
. event _ data. Pa rentProcessId 
•C : mpowershell. exe • 
Microsoft Corporation 
Windows PowerSheIl 
10.0.18362.1 
hD5=CDA48FC75952AD12D99E526DOB6BF70A , SHA2S6=9e8B64B1971 A979C7E3E8CE4621945CBA84854CB98D76367B791 A6E22B 
81 
C : mpowe rshe11. exe 
High 
{e83105be-aøc7-5e36-øeøe-møbf0511 
øx1105bf 
PowerShe11.EXE 
regsvr32. exe Is 'u /i:shotter. sct scrobj . dll 
C : mndows regsvr32. exe 
{e83105bø-ø647-5e37-øøøe-001014f27døe} 
6996 

**Task 3. Perform a hunt for renamed PowerShell.exe**

Research on PowerShell abuse and define a rule which detects the usage of a malicious file disguised as PowerShell in the available logs.

Hint: Focus on Sysmon's Process creation events (event ID 1) and be aware that the PowerShell executable preserves "PowerShell" in its description, regardless of its name.

**Solution**

Our detection is based on the fact that the PowerShell executable preserves "PowerShell" in its description, regardless of the name. So, we look into Sysmon again for any created process (event id 1) with description containing "PowerShell" that is not powershell.exe or powershell\_ise.exe. We construct the following quer**y**

|  |  |
| --- | --- |
| **winlog.event\_data.Description:\*PowerShell AND NOT (winlog.event\_data.Image:\*powershell.exe OR winlog.event\_data.Image:\*powershell\_ise.exe)** |  |

The image above reveals that the program executed is "**C:\ProgramData\Windows.exe"**

Jew Save Open Share Inspect 
Filters winlog.event_data.Description:•PowerShell AND NOT (winlog.event_data.lmage:*powershell.exe OR winlog.event_data.lmage:•powershell_ise.exe) 
04 
2012-01-01 
-01-01 
KQL 
Last 15 years 
Auto 
2018-01-01 
Show dates 
+ Add filter 
logstash-2020.02.02 
;elected fields 
_source 
\vailable fields 
'opular 
winlog.event_data.Script810ckText 
winlog.event_id 
@timestamp 
@version 
_index 
_score 
_ type 
agent.ephemeral_id 
2010-01-01 
Jan 16, 2008 @ 2358:49.444 
2014 
- Jan 16, 2023 @ — 
2016-01-01 
2020-01-01 
C Refresh 
2022-01-01 
@timestamp per 30 days 
Time 
Feb 2, 
2020 P 
winlog.event_data.Description: Windows PowerShe11 log. file.path: log. level: information winlog.event_id: 1 
winlog.provlder_name: Microsoft-Windows-sysmon winlog. record_id: 1,948 winlog.user.domain: NT AUTHORITY winlog.user.type: Well Known Group winlog.user.identifier: 
{5770385f-c22a-43ee-bf4c-06f5698ffbd9} winlog.process.pld: 2, 916 winlog.process. thread.id: 3,236 
s- 
1-5-18 
winlog . user. name: SYSTEM winlog.provider_guid: 
Expanded document 
winlog.computer_name: "INIO winlog.event_data.LogonGuid: {e83105bB-a0c7-5e36-aeee-BB20f2e511B0} winlog.event_data.User: WIN1e\AdminELS 
winloq.event_data.Fi1eVersion: 10.0.18362.1 (WinBui1d.16B101 .080B) winlog.event_data.Integr1tyLeve1: Medium 
View surrounding documents 
View single document 

tags 
winlog . api 
winlog . channel 
winlog . computer_name 
winlog 
. event _ data. CommandLine 
winlog 
. event_data. Company 
winlog 
. event_data. CurrentDirec tory 
winlog 
. event_data. Desc ription 
winlog 
. event_data. FileVersion 
winlog 
. event_data. Hashes 
winlog 
. event_data. Image 
winlog 
. event_data. Integ rityLeve1 
winlog . event_data. LogonGuid 
winlog 
. event_data. Logonld 
winlog 
. event_data. OriginalFi1eName 
wineventlog 
Microsoft-Windows-Sysmon/Operationa1 
WIN1ß 
"C : exe " 
Microsoft Corporation 
Windows PowerShe11 
10.0.18362.1 (WinBu11d.16e1B1.ß8ee) 
MD5=CDA48FC75952AD12D99E526DOB68F70A , SHA256=9B886481971 A979C7E3E8CE4621945C8A84854C898D763678 
C : exe 
Medium 
{e831ß5be-aBc7-5e36 ee2Bf2ß5110B} 
exilß5f2 
Powershell . EXE 

Another approach would be to look for **EventID 400**, where the HostName is ConsoleHost but the HostApplication is not powershell.exe.

**Screenshot**

O 
event.co e 
event . created 
event . kind 
. provider 
event 
host. name 
log . file. path 
log . level 
message 
tags 
winlog . api 
winlog . channel 
winlog . computer _name 
400 
Feb 18, 
event 
2020 21 .313 
PowerShe11 
WIN1 
C: Desktop \WindowsPS. evtx 
information 
Engine state is changed from None to Available. 
Details : 
NewEng1neState=Avai1ab1e 
PreviousEngineState=None 
SequenceNumber= 13 
HostName=Cons01eHost 
ostVersion=51.183 
HostApp1ication=C : \Progr amData Wiindows. exe 
ng ne ers on- 
Runspace1d=fef5dbb7-de9a-44ee-8744-682eB6ec029b 
Pipel Ineld= 
CommandName= 
CommandType= 
ScriptName= 
CommandPath= 
CommandLine= 
beats _ input_codec _ plain _applied 
wineventlog 
Windows PowerShe11 
WIN1 B 
winlog . event_data. paraml 
Available 
winlog . event_data. param2 
None 
winlog . event_data. param3 
NewEng1neState=Avai1ab1e 
PreviousEngineState=None 
SequenceNumber= 13 
HostName=Cons01eHost 
Hostversion=5.1 .18362.145 

**Task 4. Perform a hunt for base64-encoded PowerShell commands**

Research on PowerShell abuse and define a rule which detects the usage malicious PowerShell code, disguised in Base64 format.

Hint: Focus on Sysmon's Process creation events (event ID 1) looking for command line arguments that would match those of an encoded command. To see how attackers are encoding their commands during PowerShell attacks, refer to the resource below.

<https://docs.broadcom.com/docs-and-downloads/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf>

**Solution**

For this part, we'll use Sysmon event id 1, looking for command line arguments that would match those of an encoded command. The parameter that we are looking for is "-encodedcommand" but the bare minimum that PowerShell needs provided as arguments to a threat it as an encoded command is simply "-e". So, let's filter out for event id 1, look for all powerShell processes (powershell.exe or PowerShell in description), and parameter that contains "-e" - final query is shown below:

|  |  |
| --- | --- |
| **(winlog.event\_data.Description : \*Powershell OR winlog.event\_data.Image : \*powershell.exe) AND winlog.event\_data.CommandLine : \*-e\*** | Detecting encoded commands |

**Screenshot**

Feb 2, 
2020 P 
Expanded document 
winlog.event_data.Description: Windows PowerShe11 winlog.event_data.CommandLine: .e\powershell.exe" -ec dw80AG8AYQ8tAGkA 
winlog.event_data.lmage: C: .ß\powershell.exe winlog.event_id: 1 winlog.provider_name: Microsoft-Windows-sysmon 
winlog. record_id: 1,951 winlog.user.domain: NT AUTHORITY winlog.user.type: Well Known Group winlog.user.name: SYSTEM winlog.user.identifler: S-1-5-18 
winlog.process.pid: 2,916 winlog .process .thread.id: 3, 236 winlog.provider_guid: {577B385f-c22a-43ee-bf4c-06f5698ffbd9} winlog.computer_name: WINIB 
winloq.event_data.LogonGuid: {e83105be-a0c7-5e36-ooee-eB20bf0511BB} winlog .event_data.User: WIN10\AdminELS winloq.event_data.F11eVers10n: 10.8.18362.1 
View surrounding documents View single document 

ec pLain_appL 
winlog . channel 
Microsoft-Windows-Sysmon/Operationa1 
. event_data. Desc ription 
. event_data. FileVersion 
. event_data. Parentlmage 
tags 
winlog . api 
winlog . computer_name 
winlog . event_data. ConnandLine 
winlog 
. event_data. Company 
winlog 
. event_data. CurrentDirec tory 
winlog 
winlog 
winlog 
. event_data. Hashes 
winlog 
. event_data. Image 
winlog 
. event_data. Integ rityLeve1 
winlog 
. event_data. LogonGuid 
winlog 
. event_data. Logonld 
winlog 
. event_data. OriginalFi1eName 
winlog 
. event_data. ParentCommandLine 
winlog 
winlog 
. event_data. ParentProcessGuid 
beats_input_c 
wineventlog 
WINIB 
- C : . e \powershell. exe " 
Microsoft Corporation 
Windows PowerShe11 
ne.e.18362.1 (WinBui1d.16e1B1 .ggee) 
-ec dwBoAG8AYQBtAGkA 
MD5=CDA48FC75952AD12DggE526DOB6BF79A , SHA256=9B8B64B1971 A979C7E3E8CE4621945CBA84854CB98D76367B791 Al 
C : '\Windows \System32 . e \powershell. exe 
High 
{e831B5be-aec7-5e36 fB511 ee} 
exilB5bf 
Powershell . EXE 
- C : \Windows \System32 . e \powershell. exe " 
C : \Windows \System32 . e \powershell. exe 
{e831B5be-fa31-5e36 -egge-ee IBdeb24bee} 

If we decode the passed command, we'll find that it is the command "**whoami"**.

**Task 5. Perform a hunt for PowerShell attacks utilizing GZIP compression**

Research on PowerShell abuse and define a rule which detects the usage of GZIP compression in the available PowerShell logs.

Hint: Pay attention to event ID 4104 (PowerShell logs) for this task. GZIP archives have a magic number that can be easily identified in the logs.

**Solution**

For this task, we'll be looking at ScriptBlockText of event id 4104. After some research we find that GZIP archives have the magic number "H4sI", which appears quite unique. That will be our detection -- let's filter for event id 4104 and final query below:

|  |  |
| --- | --- |
| **winlog.event\_data.ScriptBlockText:\*H4sI\*** | Final query to find out the GZIP |

1 hit 
New Save Open 
@ + Add filter 
04 
2012-01-01 
2014-01-01 
Share Inspect 
Filters message : •scriptblock* AND winlog.event_id . 
• 4104 AND *base6. 
2010-01-01 
KQL 
Jan 17, 2008 @ 0384:47.281 - Jan 17, 2023 @ 0384:47.281 — 
2016-01-01 
@timestamp per 30 days 
Last 15 years 
Auto 
2018-01-01 
Show dates 
logstash-2020.02.02 
Selected fields 
? _source 
Available fields 
Popular 
2020-01-01 
C Refresh 
2022-01-01 
O 
winlog.event_data.Script810ckText 
winlog.event_id 
@timestamp 
@version 
_id 
_index 
_score 
_type 
Time 
10. St reamReader( New-Obj ect 
10.Compress10n.GzipStream(Sa, [10.Compression.Compress10nModel : :Decompress))) .ReadToEnd() message: Creating Scriptblock text (1 of 1 
) : Sa=New- 
Object 10. Memorystream( , [Convert] : 'H4sICIbCN14AA2EAK8/IT8zN5AIA1aßZ3AcAAAA;' )) 10 .StreamReader(New-Object 10.Compress10n .GzipStream(Sa, 
110.Compress10n.CompressionMode] : Script810ck ID: 94174ffc-f96b-4b0a-a522-fbacfe4f28ad Path: log. file.path: 
\PSOperationa1.evtx loq.level: warning winloq.opcode: On create calls winloq.activlty_id: {cBBb7b33-d9b0-eee3-94bb-ebcBb0dgd5e1} winlog.provider_name: Microsoft-Windows- 

winlog . channel 
_data. MessageNumber 
. event 
_ data. MessageTota1 
. event 
winlog . event_data. Script810ckId 
winlog . event_data. Script810ckText 
winlog . event_ld 
winlog . process . pld 
winlog . process 
. thread . id 
event . kind 
event . provider 
host. name 
log . file. path 
log . level 
message 
tags 
winlog . activity_id 
winlog . api 
winlog . computer_name 
winlog 
winlog 
winlog . opcode 
event 
Microsoft-Windows-PowerShe11 
WIN1 
C: VJsers\1user \ Desktop PSOperat10na1. evtx 
warning 
Creating Scriptblock text (1 of 1): 
Sa=New-Object 10. Memorystream( , [Convert) : H4sICIbCN14AA2EAK8/IT8zN5AIA1aBZ3AcAAAA=' )) 10. StreamReader (New-object 
Compression. GzipStream( Sa, [10.Compress10n. CompressionMode] : : Decomp ress)) ) . ReadToEnd( ) 
Script810ck ID: 94174ffc-f96b-4bßa-a522-fbacfe4f28ad 
Path : 
beats _ input_codec _ plain_applied 
{ceeb7b33-d9bB-ßß03-94bb-ebcebBdgd501 } 
wineventlog 
Microsoft-Windows-PowerShe11/Operationa1 
WIN1 
94174ffc- f96b -4bßa- a522- fbacfe4f28ad 
Sa=New-Object 10. Memorystream( , [Convert) : H4sICIbCN14AA2EAK8/IT8zN5AIA1aBZ3AcAAAA=' )) 10. StreamReader (New-object 
Compression. GzipStream( Sa, [10.Compress10n. CompressionMode] : : Decomp ress)) ) . ReadToEnd( ) 
4, 104 
On create calls 
3, 596 
4, 168 

**Task 6. Perform a hunt for obfuscated PowerShell code using XOR**

Research on PowerShell abuse and define a rule which detects the usage of a XOR command in the available PowerShell logs (usually done for code obfuscation purposes).

Hint: Pay attention to event ID 4104 (PowerShell logs) for this task. XOR usage in PowerShell involves the operators "char", "bxor" and "join".

**Solution**

For this task, we'll be looking at ScriptBlockText of event id 4104. After an extensive research, we find that XOR usage involves the operators "char", "bxor" and "join". After filtering for event id 4104e construct a query searching only for the operators "bxor" and "join" as a more broad perspective to the search:

|  |  |
| --- | --- |
| **winlog.event\_data.ScriptBlockText:(\*bxor\* AND \*join\*)** | **Searching for**XOR executed Scripts |

New Save Open Share Inspect 
Filters winlog.event_data.ScriptBlockText : ('bxor AND •join* AND •char') 
Show dates 
@ + Add filter 
logstash-2020.02.02 
Selected fields 
? _source 
Available fields 
popular 
winlog ScriptBlockText 
winlog.event_id 
@timestamp 
@version 
_ index 
-score 
04 
2010-01-01 
2012-01-01 
Time 
KQL 
Jan 17, 2008 @ - Jan 17, 2023 @ — 
2014-01-01 
2016-01-01 
@timestamp per 30 days 
Last 15 years 
Auto 
2018-01-01 
2020-01-01 
C Refresh 
2022-01-01 
Feb 2, 2020 
winlog.event_data.ScriptB10ckText: lex rnjdhl• 
I MS _ -bxor ex5} I -join • ) winlog.event_id: 4, 104 winlog.activity_id: (ceeb7b33-d9be-e0Ø3- 
f8bb-ebcøbød9d5e1} winlog.provider_name: Microsoft-Windows-powershell winlog.record_id: 45,891 winlog.user.identifier: S-1-5-21-2551594947-389e63ßß63-1692720143-1ßß2 
winlog.provider_guid: (aec1853b-5c4ø-4b15-8766-3cf1c58f985a} winlog.process.pid: 3,596 winlog.process.thread.id: 4, 168 winlog.computer_name: WINIO 
winlog.event_data.ScriptB10ckId: 73cb6688-75a3-4d18-8ca6-øf6488a2bø33 winlog.event_data.XessageNumber: 1 winlog.event_data.MessageTota1: 1 winlog .channel: Microsoft- 
Windows-powerShe11/Operational winloq.task: Execute a Remote winloq.api: wineventloq winloq.version: 1 winloq.opcode: On create calls loq.file.path: C: \Users 

wi nlog. event _data. MessageNumber 
t 
wi nlog. event _data. MessageTotal 
t 
nlog. event _data. ScriptBlockId 
t 
winlog. event _data. ScriptBlockText 
t 
winlog. event_id 
nlog. opcode 
t 
winlog. process. pid 
winlog. process. thread. id 
73±6688-75a3-4d18-8ca6-0f6488a2b033 
lex I "{S _ -bxor ex5} I -join 
4, 104 
On create calls 
3,596 
4, 168 

**Task 7. Perform a hunt for execution of an assembly from file by PowerShell**

Research on PowerShell abuse and define a rule which detects the loading and executing an assembly from a file by PowerShell.

Hint: Pay attention to event ID 4104 (PowerShell logs) for this task. To execute an assembly from file, a function named "Load" together with either "ReadAllBytes" or "LoadFile" is utilized.

**Solutions**

For this task, we'll be looking at ScriptBlockText of event id 4104. After performing a research, we find that to execute an assembly from file, a function "Load" together with either "ReadAllBytes" or "LoadFile" is utilized. After filtering for event id 4104, we construct the final query as:

|  |  |
| --- | --- |
| **winlog.event\_data.ScriptBlockText:((\*Load\*) AND (\*ReadAllBytes\* OR \*LoadFile\*))** | Assmbly files are used these  two functions to load files into powershell |

Discover 
Save Open Share Inspect 
New 
Filters winlog.event_data.ScriptBlockText : •loadfile• OR •readallbytes• 
@ + Add filter 
logstash-2020.02.02 
Selected fields 
? _source 
Available fields 
04 
popular 
2012-01-01 
2010-01-01 
KQL 
Jan 17, 2008 @ - Jan 17, 2023 @ — 
2014-01-01 
2016-01-01 
@timestamp per 30 days 
Last 15 years 
Auto 
2018-01-01 
Show dates 
2020-01-01 
C Refresh 
2022-01-01 
winlog ScriptBlockText 
winlog.event_id 
@timestamp 
@version 
_ index 
_score 
t _ type 
agent.ephemeral_id 
Time 
Feb 2, 2020 
winlog.event_data.ScriptB10ckText: log. file.path: 
log. level: verbose winlog.event_id: 4.104 winlog.activity_id: {cøeb7b33-d9be-øøe2-e7b1-ebcebed9d5e1) winlog.provider_name: Microsoft-Hindows-powershell 
winlog.record_id: 49.612 winlog.provider_guid: {aec1853b-5c4ø-4b15-8766-3cf1c58f985a} winlog.user.identifier: s-1-5-21-2551594947-3890630063-1692720143-1002 
winlog.process.pid: 3.596 winlog.process.thread.id: 4.168 winlog.co.uter_name: WIN1e winlog.event_data.ScriptBlockId: d14417b7-f6a8-43ee-871f-69ed812c494f 
winloq.event_data.MessaqeNwber: 1 winloq.event_data.kssaqeTotal: 1 winloq.channel: Microsoft-Kindows-powerShell/Operationa1 winloq.task: Execute a Remote Command 

**Task 8. Perform a hunt for PowerShell commands downloading content**

Research on PowerShell abuse and define a rule which detects PowerShell downloading content.

Hint: Pay attention to event ID 4104 (PowerShell logs) for this task. There are multiple ways using which PowerShell can download content from remote sources. Some of them are WebClient, DownloadData, DownloadFile, DownloadString etc.

**Solution**

For this task, we'll be looking at ScriptBlockText of event id 4104. After doing our research, we identify a large number of possibilities for download content. The final query with all of them included is shown below:

|  |  |
| --- | --- |
| **winlog.event\_data.ScriptBlockText:(\*WebClient\* OR \*DownloadData\* OR \*DownloadFile\* OR \*DownloadString\* OR \*OpenRead\* OR \*WebRequest\* OR \*curl\* OR \*wget\* OR \*RestMethod\* OR \*WinHTTP\* OR \*InternetExplorer.Application\* OR \*Excel.Application\* OR \*Word.Application\* OR \*Msxml2.XMLHTTP\* OR \*MsXML2.ServerXML\* OR \*System.XML.XMLDocument\* OR \*BitsTransfer\*)** | To download content or files by powershell |

**Screenshot**

A screenshot of a computer

Description automatically generated

The search detected download through the commands:

* **Start-BitsTransfer**
* **Curl**
* **Invoke-RestMethod**

At this point, you should also be aware that COM objects can be used to perform file download, therefore, the following query should also be incorporate in the search. In our research, we identified the following CLSIDs to be interesting:

|  |  |
| --- | --- |
| **0002DF01-0000-0000-C000-000000000046** | **InternetExplorer.Application** |
| **F6D90F16-9C73-11D3-B32E-00C04F990BB4** | **Msxml2.XMLHTTP** |
| **F5078F35-C551-11D3-89B9-0000F81FE221** | **Msxml2.XMLHTTP.3.0** |
| **88D9D96A-F192-11D4-A65F-0040963251E5** | **Msxml2.XMLHTTP.6.0** |
| **AFBA6B42-5692-48EA-8141-DC517DCF0EF1** | **Msxml2.ServerXmlHttp** |
| **AFB40FFD-B609-40A3-9828-F88BBE11E4E3** | **Msxml2.ServerXmlHttp.3.0** |
| **88D96A0B-F192-11D4-A65F-0040963251E5** | **Msxml2.ServerXmlHttp.6.0** |
| **2087C2F4-2CEF-4953-A8AB-66779B670495** | **WinHttp.WinHttpRequest.5.1** |
| **000209FF-0000-0000-C000-000000000046** | **Word.Application** |
| **00023500-0000-0000-C000-000000000046** | **Excel.Application** |

**Task 9. Perform a hunt for obfuscated PowerShell commands**

Research on PowerShell abuse and define a rule which detects obfuscated PowerShell commands.

Hint: Pay attention to event ID 4104 (PowerShell logs) for this task. There are multiple strings that are associated with PowerShell code obfuscation. Some of them are "char", "join", "ToInt", "ToInt16", "ToDecimal", "ToString", "Xor" etc. Also be aware that sometimes attackers spell words/commands in the opposite order.

**Solution**

For this task, we'll be looking at ScriptBlockText of event id 4104. After doing a research on obfuscated commands, we identify a large number of interesting characters that may be of interest, which compose the final query as:

|  |  |
| --- | --- |
| **winlog.event\_data.ScriptBlockText:((\*char\* AND \*join\*) OR ((\*ToInt\* OR \*ToInt16\* OR \*ToDecimal\* OR \*ToByte\* OR \*ToUnit\* OR \*ToSingle\*) AND (\*ToChar\* OR \*ToString\* OR \*String\*)) OR (\*ForEach\* AND \*Xor\*))** | Obfusacted commands  **Screenshot 1** |
| **winlog.event\_data.ScriptBlockText:(\*hctac\* OR \*kearb\* OR \*dnammoc\* OR \*ekovn\* OR \*elifd\* OR \*rahc\* OR \*etirw\* OR \*eddih\* OR \*tpircs\* OR \*ssecorp\* OR \*llehsrewop\* OR \*esnopser\* OR \*daolnwod\* OR \*tneilcbew\* OR \*tneilc\* OR \*ptth\* OR \*elifotevas\* OR \*46esab\* OR \*tcejbo\* OR \*maerts\* OR \*hcaerof\* OR \*retupmoc\*)** | Reverse obfuscated command    **Screenshot 2** |

**Screenshot 1**

Time 
Feb 2, 
Feb 2, 
Feb 2, 
Feb 2, 
Feb 2, 
2020 
2020 0 .839 
2020 P 
2020 P 
2020 P 
winlog.event_data.Scr1pt810ckText: " S(sv OFS' 
'77q68q6fy61H6d-69' -SpliT'U' -SPLit'y' -SPLIt 
-SPLIT'}' -SPLIT' 
-SPLIt 
OBJEct{ ([CHaR] ( [conVERT]: ))) +" S(sET-iTem 'VaRiabLE:ofs' 
log. file 
-sPL1t'q' -spllt'z' ForEAcH 
. path: C: 
\PSOperationa1.evtx log. level: verbose winlog.event_id: 4, 104 winlog.activlty_id: {cB0b7b33-d9b0-eee3-35c3-ebcBb0dgd5B1} winlog.provider_name: Microsoft-Windows- 
PowerShe11 winlog. record_id: 67 , 841 winlog.provlder_guid: 
winloq.process.pid: 3,596 winlog .process .thread.ld: 4, 168 
winlog.event_data.Scr1pt810ckText: iex rmjdhl' 
f8bb-dbcebed9d5d1} winlog. provider_name: Microsoft-Windows 
{aeci853b-5c4e-4b15-8766-3cf1c58f985a} winlog.user.ldentifler: S-1-5-21-2551 594947-3890630063-1692720143-1002 
winloq. computer_name: WINIB winlog.event_data.ScriptB10ckId: 41494f34-db93-4fd4-9991-eb95a3a6d5B7 
-bxor Bx5} I 
' ' ) winlog.event_ld: 4, 104 winlog. activity_id: {ceeb7b33-dgbe-BB03- 
-join 
-Powershell winlog. record_id: 45,891 .user.identifier: S-1-5-21-2551 594947-3890630063-1692720143-1002 
winlog.provlder_guid: {a0c1853b-5c4e-4b15-8766-3cf1c58f985a} winlog.process.pid: 3, 596 winlog .process .thread.id: 4, 168 winlog.computer_name: WINIB 
winlog.event_data.Scr1pt810ckId: 73cb6688-75a3-4d18-8ca6-0f6488a2b033 winlog.event_data.MessageNumber: 1 winlog.event_data.MessageTota1: 1 winlog .channel: Microsoft- 
Windows-PowerShe11/Operationa1 winloq.task: Execute a Remote Command winloq.api: wineventlog winloq.version: 1 winloq.opcode: On create calls log. file.path: C:\Users 
winlog.event_data.Scr1pt810ckText: { ([CHaR] ( [conVERTl : [strInG]S_),16 ))) } log.file.path: log. level: verbose 
winlog.event_id: 4, 104 winlog.activity_ld: {ceeb7b33-d9bB-0003-37c3-0bcebBdgd501} winlog.provider_name: Microsoft-Windows-PowerShe11 winlog. record_id: 67,843 
.user.ldentifier: s-1-5-21-2551594947-3890630063-1692720143-1002 winlog.provlder_guld: {aec1853b-5c4e-4b15-8766-3cf1c58f985a} winlog.process.pid: 3, 596 
win log 
winlog.process.thread.id: 4,168 winlog.computer_name: WINIO winlog.event_data.Script810ckId: 8fb1152e-72fg-4e77-g46e-B35a752eabBa winlog .event_data.MessageNumber: 
winloq.event_data.MessaqeTota1: 1 winlog .channel: Microsoft-Windows-PowerShe11/0perationa1 winloq.task: Execute a Remote Command winlog.api: wineventloq 
winlog.event_data.Scr1pt810ckText: { ([CHaR] ( [conVERTl : [strInG]S_),16 ))) } winlog.opcode: On create calls winlog.activity_id: {ceBb7b33- 
d9bB-0003-13c6-0bcebBdgd501} winlog.provider_name: Microsoft-Windows-PowerShe11 winlog. record_id: 78,749 winlog.user .ldentifier: S- 
1-5-21-2551594947-3890630063-1692720143-1002 winlog.process.pid: 3, winlog.process.thread .id: 4, 168 winlog.provider_guid: {aBc1853b-5c4a-4b15-8766-3cf1c58f985a} 
winlog.computer_name: "INIO winlog.event_data.ScriptB10ckId: 40c7eB42-bda4-435e-94f1-56258f6791bf winlog.event_data.MessageNumber: 1 winlog.event_data.MessageTota1: 1 
winloq.channel: Microsoft-Windows-PowerShe11/Operationa1 winloq.task: Execute a Remote Command winlog .api: wineventlog winloq.version: 1 winloq.event_id: 4, 104 
winlog.event_data.Scr1pt810ckText: " S(sv OFS' 
-SPLIT'}' -SPLIT' 
'77q68q6fy61H6d-69' -SpliT'U' -SPLit'y' -SPLIt 
-SPLIt'H' 
-sPL1t'q' -spllt'z' ForEAcH 
OBJEct{ ([CHaR] ( [conVERT]: ))) 
1): 
S(sv 'OFS 
'77q68q6fy61H6d-69' -spli 
IconVERT] ))) +" S(sET-iTem 
loq.file.path: 
+" S(sET-iTem 'VaRiabLE:ofs' 
. ( message: Creating Scriptblock text (1 of 
U' -SPLit'y' -SPLIt 
VaRiabLE:ofs' 
log. level: verbose 
-SPLIT' } ' 
'HI -sPL1t'q' -split 'z' I ForEAcH-OBJEct{ ([CHaR] ( 
-SPLIT' ; 
-SPLIt 
Script810ck ID: 
5e381dad-3295-4385-8541-fef71 587b7ad Path: 
tags: host. name: 
WINIO agent . type: winloqbeat 

Indeed, some ofcuscated commands were detected. 4 of the 5 identified logs appear to be associated with the same command, while he 5th one is a false positive from a previous exercise. Therefore, we have identified only 1 command so far, we should investigate for more obfuscation techniques to find others.

Additional research leads us to reverse obfuscated, or certain words spelled backwards. We go through common cmdlets, and terms and construct a query with them in reverse order as follows:

To detect the second obfuscated command, we utilize the following query, which identifies PowerShell commands spelled backwards:

**Screenshot 2**

**winlog.event\_data.ScriptBlockText:(\*hctac\* OR \*kearb\* OR \*dnammoc\* OR \*ekovn\* OR \*elifd\* OR \*rahc\* OR \*etirw\* OR \*eddih\* OR \*tpircs\* OR \*ssecorp\* OR \*llehsrewop\* OR \*esnopser\* OR \*daolnwod\* OR \*tneilcbew\* OR \*tneilc\* OR \*ptth\* OR \*elifotevas\* OR \*46esab\* OR \*tcejbo\* OR \*maerts\* OR \*hcaerof\* OR \*retupmoc\*)**

1 hit 
New Save Open Share Inspect 
Filters winlog.event_data.ScriptBlockText:(*hctac* OR •kearb• OR •dnammoc* OR •ekovn* OR *elifd* OR *rahc* OR *etirw* OR OR *tpircs* OR OR •l 
04 
2012-01-01 
2014-01-01 
2020 P 
KQL 
Last 15 years 
Auto 
2018-01-01 
Show dates 
@ + Add filter 
logstash-2020.02.02 
Selected fields 
? _source 
Available fields 
Popular 
winlog.event_data.Script810ckText 
winlog.event_id 
@timestamp 
@version 
_id 
_index 
_score 
_type 
agent.ephemeral_id 
2010-01-01 
Jan 17, 2008 @ 0508:54.919 - Jan 17, 2023 @ 0508:54.919 — 
2016-01-01 
@timestamp per 30 days 
2020-01-01 
C Refresh 
2022-01-01 
Time 
Feb 2, 
winlog. event_data. Script810ckText: SA06Jug= 
EcALPEr- 631raHC[, 'AzY' ECaLPErc- 93]raHcI, '0m2' ECaLPErc- ECaLPErc 
'+'B3[E'+'MOHSpAzY+l'+'41EMohsPA'+'zY (.Dxw y'+'D8 om2 0'+'m2 '+' om2sfo:ELbaiRavom2 me'+'T'+'i-TEs('+'AzY yD'+'8+ j)) 
'+'( IR'+'aH'+'C[ '+'( {t'+'cEJBO-H'+'cA'+'E'+' r'+'oF Dxw 0'+'m2'+'zom2t'+'11ps-om2qom2'+'t1'+'LPs- om2Hom2t'+'1cps-'+' 
cps- om2}'+' om2TILps-om2-om2 t' *'ILps-om2y0'+'m2tiLPS- om2U'+'0m2Ti'+' Ips-0'+'m296- 70m2 ('+1 lg'+'N'+' SI+YDB'+') om20m2'+' 
om2SF00m2 vs(AzY ( & " ; 
(ql :aB6JuG' - 

**KEY TAKEWAYS**

PowerShell has a number of techniques that can be utilized to circumvent detection and analysts.