Lab #14: **Understanding and Triggering Sentinel Alerts**

Purpose:

- We'll explore some of the custom SIEM rules that we set up in the last lab. We'll analyze the KQL and ensure the rules are appropriately configured.

Tasks:

1. **Trigger AAD Brute Force Success**
   - Generate some logs
   - Observe the generated logs (in Log Analytics workspace)
2. **Trigger MSSQL Brute Force Attempt**
   - Generate some logs
   - Observe the generated logs
3. **Trigger Malware Outbreak**
   - Generate some logs
   - Observe the generated logs
4. **Trigger Possible Privilege Escalation (in Key Vault)**
   - Generate some logs
   - Observe the generated logs
5. **Trigger Windows Host Firewall Tampering**
   - Generate some logs
   - Observe the generated logs
6. **Trigger Excessive Password Resets**
   - Generate some logs
   - Observe the generated logs

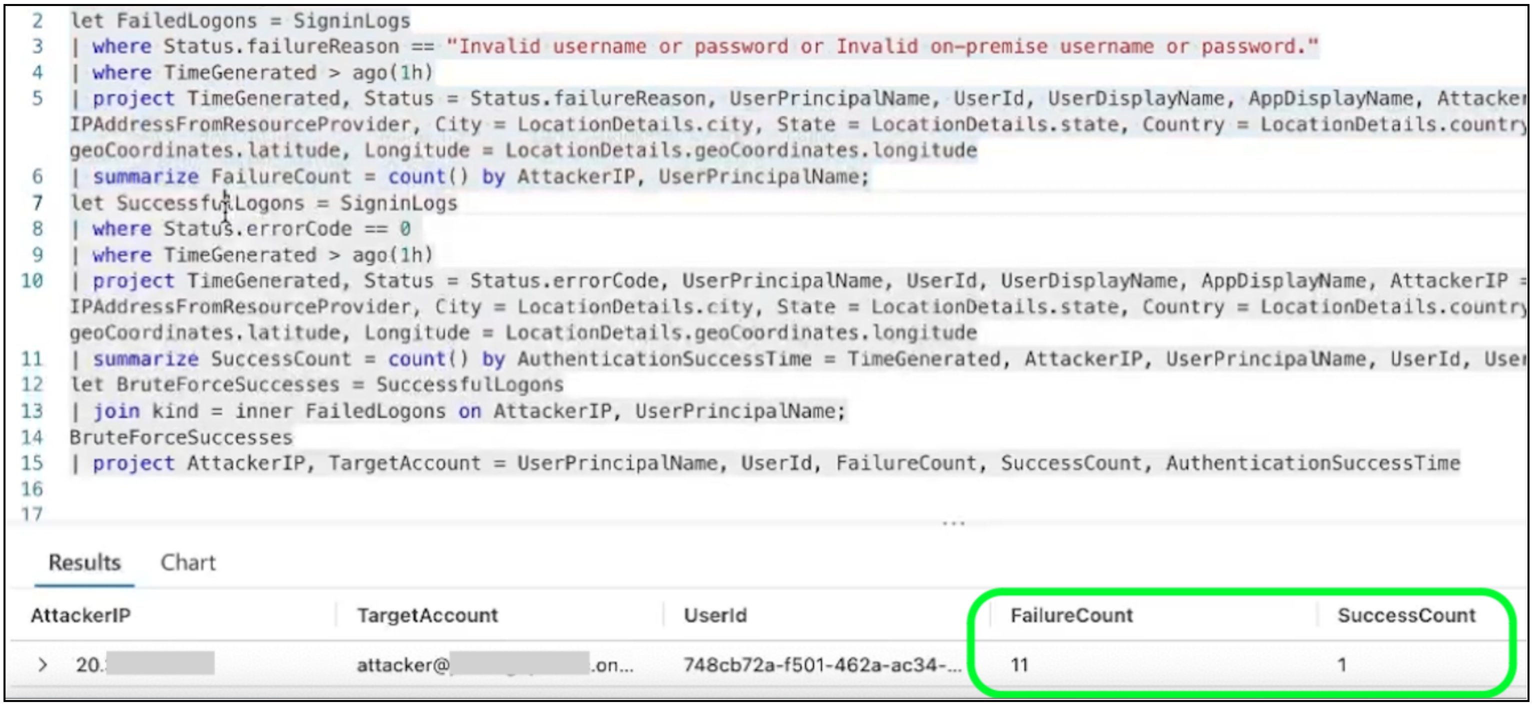
Task 1: **Trigger AAD Brute Force Success**

Generate some logs:

1. Log into the **attack-vm** > log into portal.azure.com using a test "attacker" account (Entra ID).
2. Attempt to log in 10x (using valid username and incorrect password).
3. Attempt to log in 1x, but now use the correct password.

Observe the generated logs (in Log Analytics workspace):

1. In **Sentinel** > **Analytics**, locate the 'ADD Brute Force Success' rule and copy its query script.
2. Run the copied query (go to **Log Analytics workspace** > **Logs**):



```
let FailedLogons = SigninLogs
| where Status.failureReason == "Invalid username or password or Invalid on-premise username or password."
| where TimeGenerated > ago(1h)
| project TimeGenerated, Status = Status.failureReason, UserPrincipalName, UserId, UserDisplayName, AppDisplayName, Attacker
IPAddressFromResourceProvider, City = LocationDetails.city, State = LocationDetails.state, Country = LocationDetails.country
geoCoordinates.latitude, Longitude = LocationDetails.geoCoordinates.longitude
| summarize FailureCount = count() by AttackerIP, UserPrincipalName;
let SuccessfulLogons = SigninLogs
| where Status.errorCode == 0
| where TimeGenerated > ago(1h)
| project TimeGenerated, Status = Status.errorCode, UserPrincipalName, UserId, UserDisplayName, AppDisplayName, AttackerIP =
IPAddressFromResourceProvider, City = LocationDetails.city, State = LocationDetails.state, Country = LocationDetails.country
geoCoordinates.latitude, Longitude = LocationDetails.geoCoordinates.longitude
| summarize SuccessCount = count() by AuthenticationSuccessTime = TimeGenerated, AttackerIP, UserPrincipalName, UserId, User
let BruteForceSuccesses = SuccessfulLogons
| join kind = inner FailedLogons on AttackerIP, UserPrincipalName;
BruteForceSuccesses
| project AttackerIP, TargetAccount = UserPrincipalName, UserId, FailureCount, SuccessCount, AuthenticationSuccessTime
```

AttackerIP	TargetAccount	UserId	FailureCount	SuccessCount
20.[redacted]	attacker@[redacted].on...	748cb72a-f501-462a-ac34-...	11	1

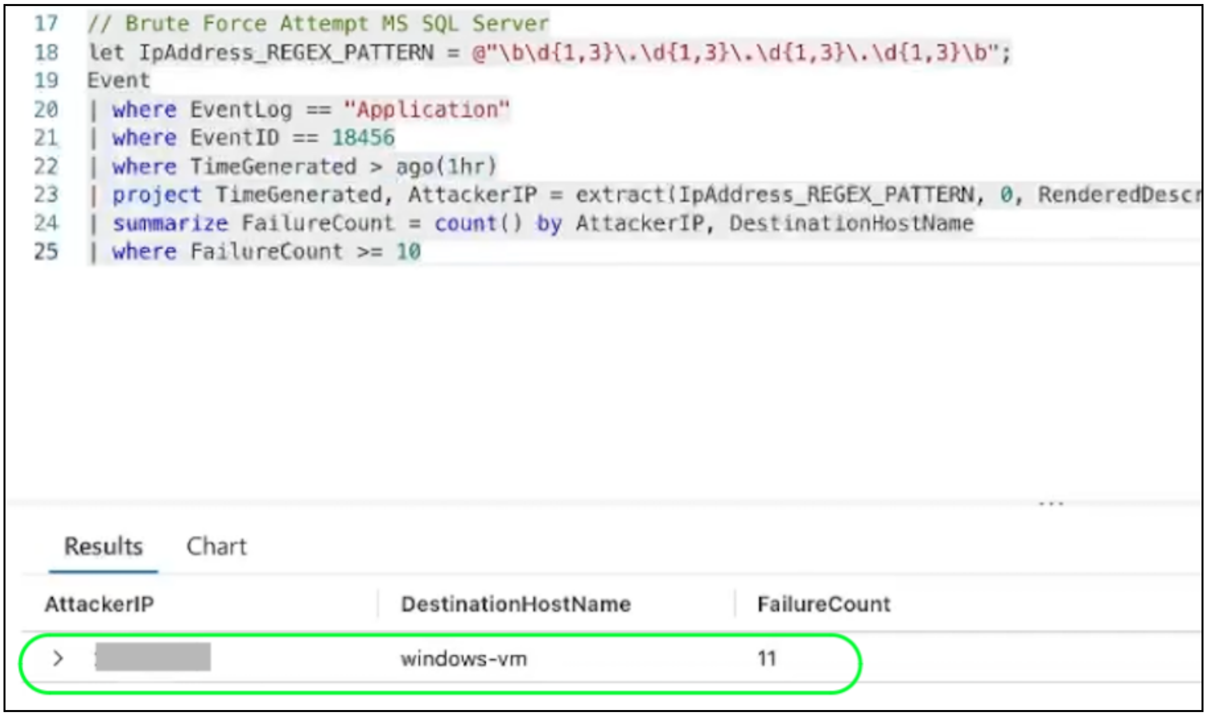
Task 2: Trigger MSSQL Brute Force Attempt

Generate some logs:

1. Log into the **attack-vm** > open SSMS.
2. In SSMS, attempt to log into SQL server 15x (using valid username, incorrect password)

Observe the generated logs (in Log Analytics workspace):

1. (in **Sentinel** > **Analytics**) Locate this rule and copy its query script.
2. Run the query:



```
// Brute Force Attempt MS SQL Server
let IpAddress_REGEX_PATTERN = @"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b";
Event
| where EventLog == "Application"
| where EventID == 18456
| where TimeGenerated > ago(1hr)
| project TimeGenerated, AttackerIP = extract(IpAddress_REGEX_PATTERN, 0, RenderedDescr
| summarize FailureCount = count() by AttackerIP, DestinationHostName
| where FailureCount >= 10
```

AttackerIP	DestinationHostName	FailureCount
[redacted]	windows-vm	11

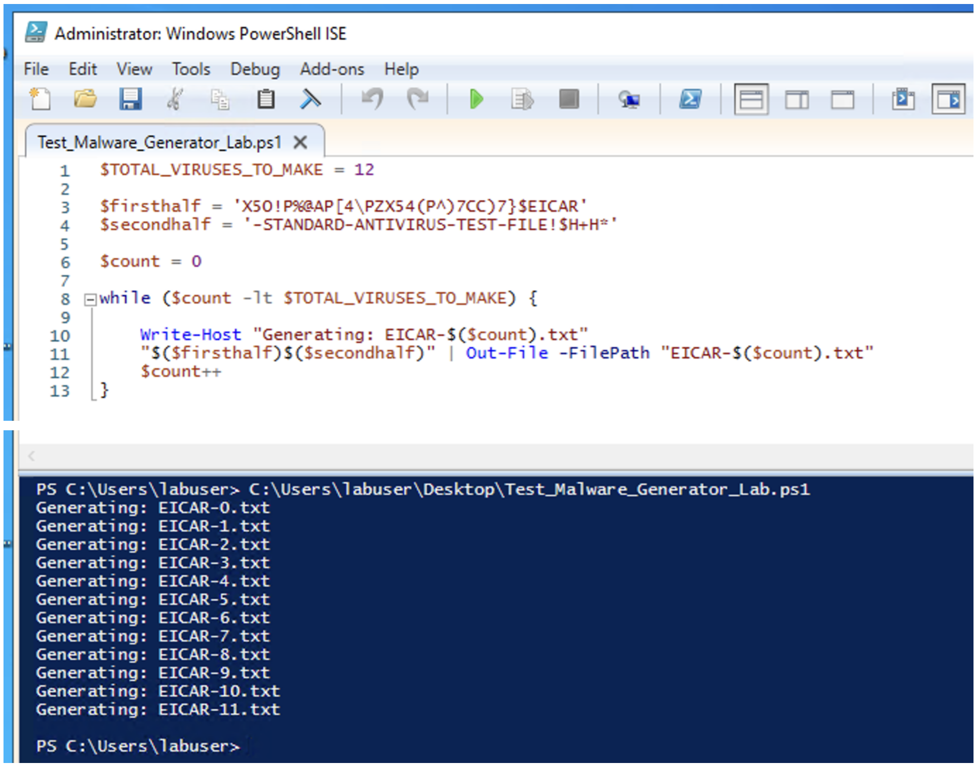
Task 3: Trigger Malware Outbreak

Generate some logs:

1. Log into the **windows-vm** > open the Microsoft **Edge** browser.
2. In **Edge**, go to this GitHub link and select **Copy raw file** (malware PS generator script): github.com/erichmair/Azure-SOC-Honeynet-Project/blob/main/Attack-Scripts/Malware-Generator-EICAR.ps1

   ***Note**: The test script includes strings that automatically get flagged as malware. It'll trigger a malware alert without actually installing malware.*

3. Open **PS ISE** > select **New File** > paste the script > select **Run**.

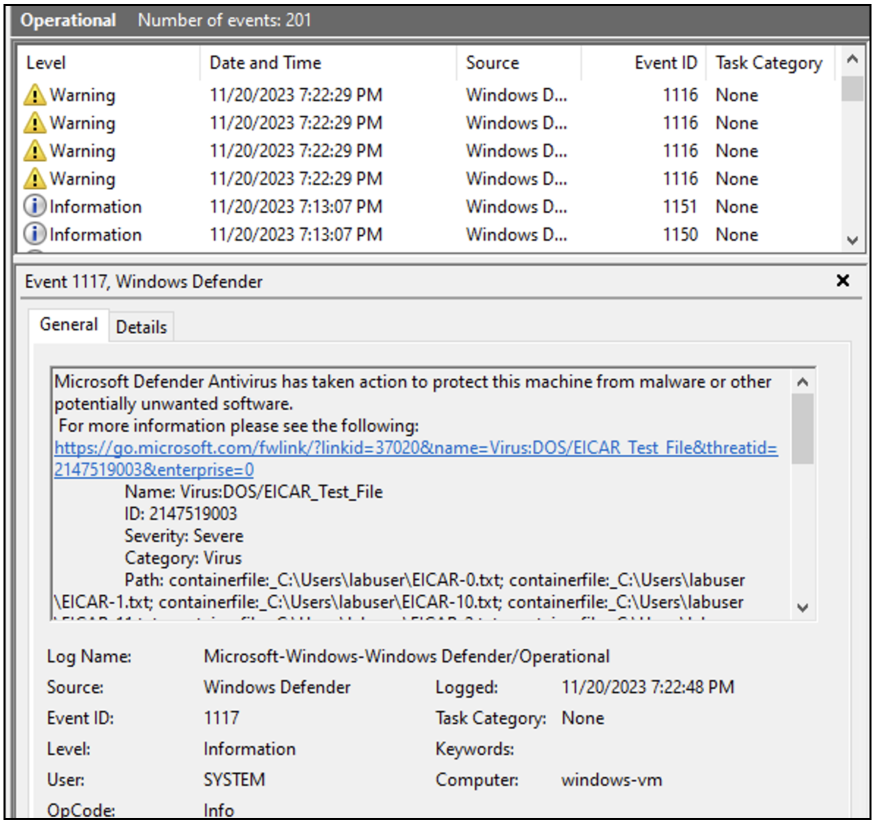


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Test_Malware_Generator_Lab.ps1

```
$TOTAL_VIRUSES_TO_MAKE = 12

$firsthalf = 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR'
$secondhalf = '-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*'

$count = 0

while ($count -lt $TOTAL_VIRUSES_TO_MAKE) {

    Write-Host "Generating: EICAR-$($count).txt"
    "$($firsthalf)$($secondhalf)" | Out-File -FilePath "EICAR-$($count).txt"
    $count++
}
```

```
PS C:\Users\labuser> C:\Users\labuser\Desktop\Test_Malware_Generator_Lab.ps1
Generating: EICAR-0.txt
Generating: EICAR-1.txt
Generating: EICAR-2.txt
Generating: EICAR-3.txt
Generating: EICAR-4.txt
Generating: EICAR-5.txt
Generating: EICAR-6.txt
Generating: EICAR-7.txt
Generating: EICAR-8.txt
Generating: EICAR-9.txt
Generating: EICAR-10.txt
Generating: EICAR-11.txt

PS C:\Users\labuser>
```

4. In **Event Viewer**, we can see the newly generated alerts:



Operational Number of events: 201

Level	Date and Time	Source	Event ID	Task Category
⚠ Warning	11/20/2023 7:22:29 PM	Windows D...	1116	None
⚠ Warning	11/20/2023 7:22:29 PM	Windows D...	1116	None
⚠ Warning	11/20/2023 7:22:29 PM	Windows D...	1116	None
⚠ Warning	11/20/2023 7:22:29 PM	Windows D...	1116	None
ⓘ Information	11/20/2023 7:13:07 PM	Windows D...	1151	None
ⓘ Information	11/20/2023 7:13:07 PM	Windows D...	1150	None

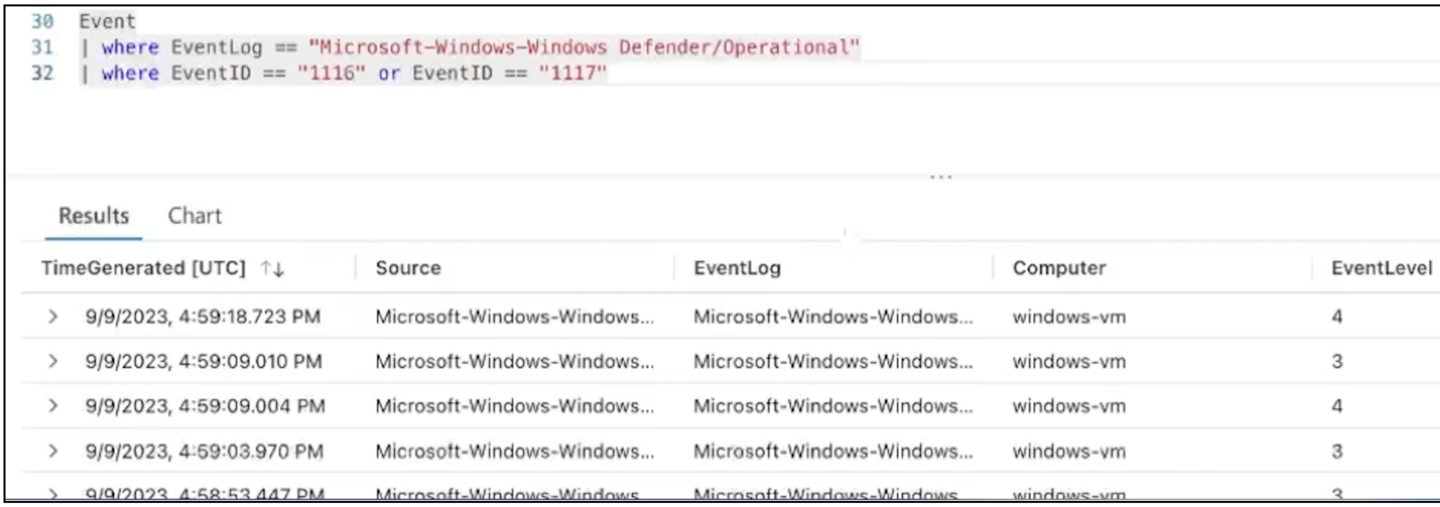
Event 1117, Windows Defender

Microsoft Defender Antivirus has taken action to protect this machine from malware or other potentially unwanted software.
For more information please see the following:
https://go.microsoft.com/fwlink/?linkid=37020&name=Virus:DOS/EICAR_Test_File&threatid=2147519003&enterprise=0
Name: Virus:DOS/EICAR_Test_File
ID: 2147519003
Severity: Severe
Category: Virus
Path: containerfile:_C:\Users\labuser\EICAR-0.txt; containerfile:_C:\Users\labuser\EICAR-1.txt; containerfile:_C:\Users\labuser\EICAR-10.txt; containerfile:_C:\Users\labuser

Log Name:	Microsoft-Windows-Windows Defender/Operational		
Source:	Windows Defender	Logged:	11/20/2023 7:22:48 PM
Event ID:	1117	Task Category:	None
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	windows-vm
OpCode:	Info		

Observe the generated logs (in Log Analytics workspace):

1. (in **Sentinel** > **Analytics**) Locate this rule and copy its query script.
2. Run the query:



```
Event
| where EventLog == "Microsoft-Windows-Windows Defender/Operational"
| where EventID == "1116" or EventID == "1117"
```

TimeGenerated [UTC] ↑↓	Source	EventLog	Computer	EventLevel
9/9/2023, 4:59:18.723 PM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	4
9/9/2023, 4:59:09.010 PM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	3
9/9/2023, 4:59:09.004 PM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	4
9/9/2023, 4:59:03.970 PM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	3
9/9/2023, 4:58:53.447 PM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	3

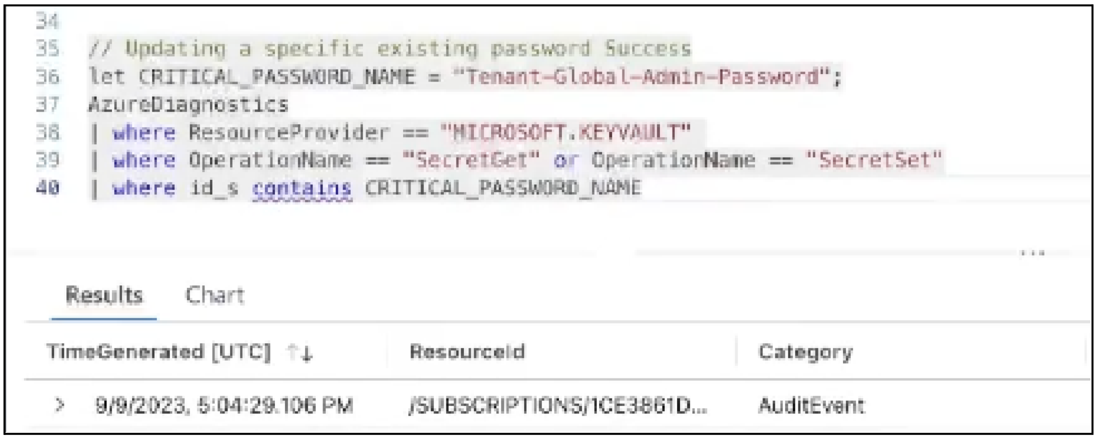
Task 4: Trigger Possible Privilege Escalation (in Key Vault)

Generate some logs:

1. **Azure** portal > **Key Vault** > (your key vault) > **Secrets** >
2. Open the **Tenant-Global-Admin-Password** secret.

Observe the generated logs (in Log Analytics workspace):

1. (in **Sentinel** > **Analytics**) Locate this rule and copy its query script.
2. Run the query:



```

// Updating a specific existing password Success
let CRITICAL_PASSWORD_NAME = "Tenant-Global-Admin-Password";
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.KEYVAULT"
| where OperationName == "SecretGet" or OperationName == "SecretSet"
| where id_s contains CRITICAL_PASSWORD_NAME
```

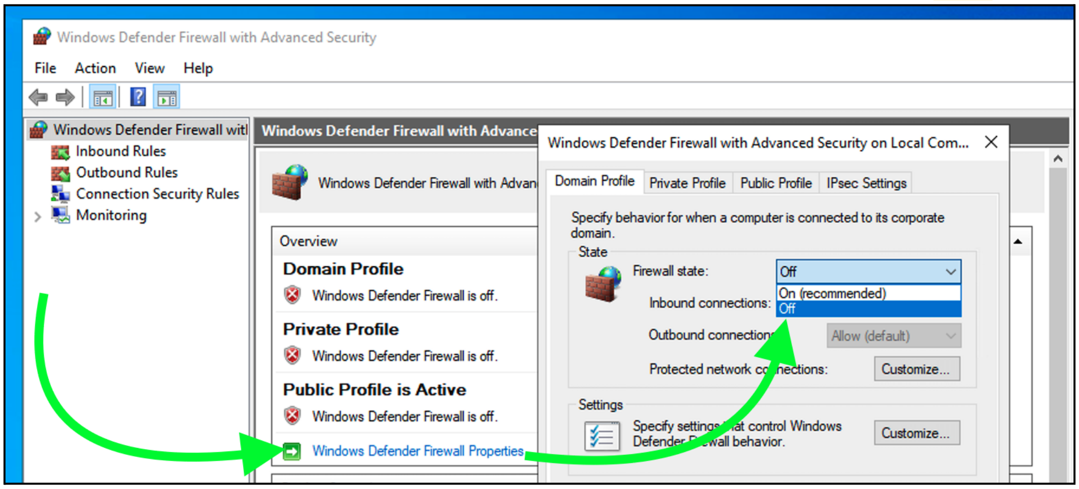
TimeGenerated [UTC] ↑↓	ResourceId	Category
9/9/2023, 5:04:29.106 PM	/SUBSCRIPTIONS/1CE3861D...	AuditEvent

Task 5: Trigger Windows Host Firewall Tampering

Generate some logs:

1. Log into the **windows-vm** > open **Windows Defender Firewall**

2. Select **Windows Defender Firewall Properties** > set **Firewall State** to **OFF**.


Windows Defender Firewall with Advanced Security
File
Action
View
Help
Windows Defender Firewall witl
Inbound Rules
Outbound Rules
Connection Security Rules
Monitoring
Windows Defender Firewall with Advance
Windows Defender Firewall with Advan
Overview
Windows Defender Firewall with Advanced Security on Local Com...
Domain Profile
Private Profile
Public Profile
IPsec Settings
Customize...
Customize...

Domain Profile
Windows Defender Firewall is off.

Private Profile
Windows Defender Firewall is off.

Public Profile is Active
Windows Defender Firewall is off.

Windows Defender Firewall Properties

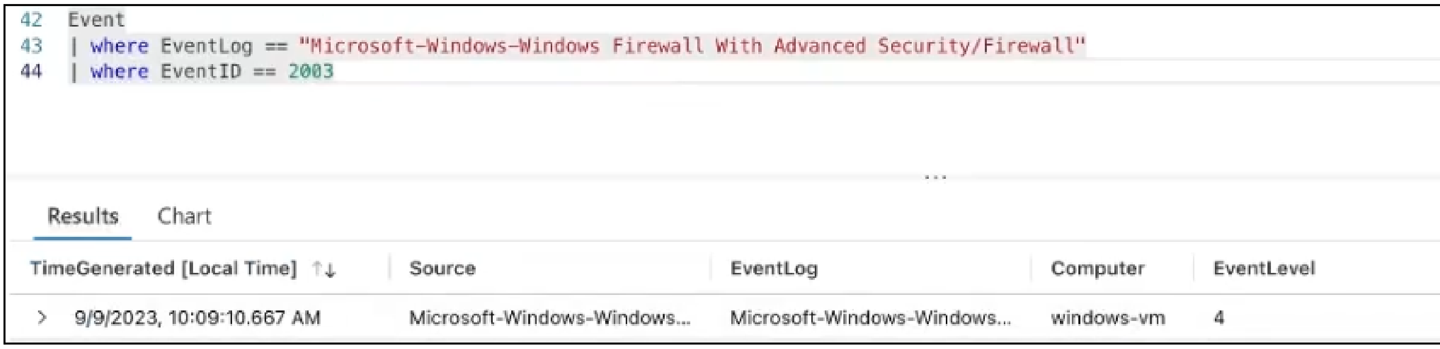
Specify behavior for when a computer is connected to its corporate domain.

State
Firewall state: Off
On (recommended)
Off
Inbound connections:
Outbound connections: Allow (default)
Protected network connections:

Settings
Specify settings that control Windows Defender Firewall behavior.

Observe the generated logs (in Log Analytics workspace):

1. (in **Sentinel** > **Analytics**) Locate this rule and copy its query script.
2. Run the query:



```
Event
| where EventLog == "Microsoft-Windows-Windows Firewall With Advanced Security/Firewall"
| where EventID == 2003
```

TimeGenerated [Local Time]	Source	EventLog	Computer	EventLevel
9/9/2023, 10:09:10.667 AM	Microsoft-Windows-Windows...	Microsoft-Windows-Windows...	windows-vm	4

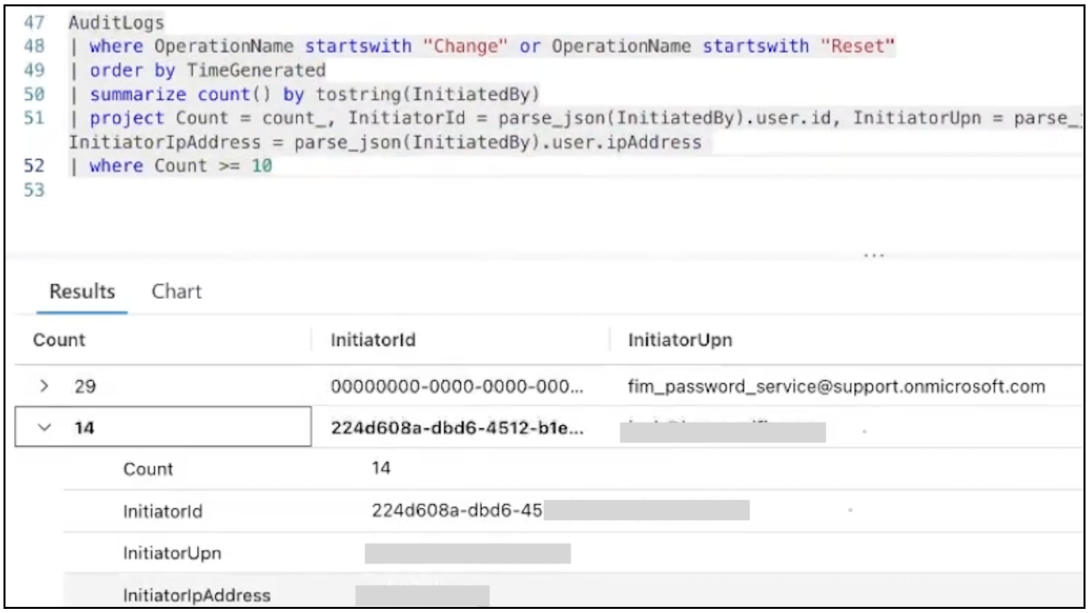
Task 6: Trigger Windows Host Firewall Tampering

Generate some logs:

1. **Azure** portal > **Entra ID** > create a new dummy user account and reset its password more than 10x.

Observe the generated logs (in Log Analytics workspace):

1. (in **Sentinel** > **Analytics**) Locate this rule and copy its query script.
2. Run the query:



```
AuditLogs
| where OperationName startswith "Change" or OperationName startswith "Reset"
| order by TimeGenerated
| summarize count() by tostring(InitiatedBy)
| project Count = count_, InitiatorId = parse_json(InitiatedBy).user.id, InitiatorUpn = parse_
InitiatorIpAddress = parse_json(InitiatedBy).user.ipAddress
| where Count >= 10
```

Count	InitiatorId	InitiatorUpn
29	00000000-0000-0000-000...	fim_password_service@support.onmicrosoft.com
14	224d608a-dbd6-4512-b1e...	[redacted]

Count	14
InitiatorId	224d608a-dbd6-45[redacted]
InitiatorUpn	[redacted]
InitiatorIpAddress	[redacted]

End:

- We tested several of the custom SIEM rules by triggering them.
- In future labs, we’ll investigate SIEM incidents and perform incident response steps.