MTP Advance Hunting Cheat Sheet v0.1 | https://github.com/MiladMSFT/AdvHuntingCheatSheet | @MiladMSFT

The purpose of this cheat sheet is to cover commonly used threat hunting queries that can be used with Microsoft Threat Protection. Microsoft Threat Protection has a threat hunting capability that is called Advance Hunting (AH). AH is based on Azure Kusto Query Language (KQL). If you are looking for an KQL cheat sheet click here. Special thanks to @PowershellPoet, @pawp81, @maarten_goet, @Bakk3rM and @MicrosoftMTP who contributed to this work.

Email (Office 365 ATP)

Pull SHA256 out of text file and look for Email attachments that matches the SHA256. Author: @pawp81

```
let abuse_sha256 =
(externaldata(sha256_hash: string )
[@"https://bazaar.abuse.ch/export/txt/sha2
56/recent/"]
with (format="txt"))
| where sha256_hash !startswith "#"
| project sha256_hash;
abuse_sha256
| join (EmailAttachmentInfo
| where Timestamp > ago(1d)
) on $left.sha256_hash == $right.SHA256
| project Timestamp,SenderFromAddress
,RecipientEmailAddress,FileName,FileType,S
HA256,
MalwareFilterVerdict,MalwareDetectionMetho
d
```

Lookup for emails coming into the organization from an external source that was targeted to more than 50 distinct corporate users. Author: @MiladMSFT

```
EmailEvents
| where SenderFromDomain !=
"corporatedomain.com"
| summarize dcount(RecipientEmailAddress)
by SenderFromAddress, NetworkMessageId,
AttachmentCount, SendTime = Timestamp
| where dcount_RecipientEmailAddress > 50
```

Lookup for all emails within last 7 days where the malware verdict was Malware. Author: @MiladMSFT

```
EmailEvents
| where Timestamp > ago(7d)
| where MalwareFilterVerdict == "Malware"
| project Timestamp,
SenderMailFromAddress,
RecipientEmailAddress,
MalwareDetectionMethod, DeliveryAction
```

Hybrid

Identity + Endpoint: Lookup processes that performed LDAP auth. with cleartext passwords. Author: @MicrosoftMTP

```
IdentityLogonEvents
| where Timestamp > ago(7d)
| where LogonType == "LDAP cleartext" and
isnotempty(AccountName)
| project LogonTime = Timestamp,
DeviceName, AccountName, Application,
LogonType
| join kind=inner (
DeviceNetworkEvents
| where Timestamp > ago(7d)
| where ActionType == "ConnectionSuccess"
| extend DeviceName =
toupper(trim(@"\..*$",DeviceName))
| where RemotePort == "389"
| project NetworkConnectionTime =
Timestamp, DeviceName, AccountName =
InitiatingProcessAccountName,
InitiatingProcessFileName,
InitiatingProcessCommandLine
) on DeviceName, AccountName
| where LogonTime - NetworkConnectionTime
between (-2m .. 2m)
| project Application, LogonType,
LogonTime, DeviceName, AccountName,
InitiatingProcessFileName,
InitiatingProcessCommandLine
```

Find processes that sent SAMR queries to Active Directory. Author: MTP engineering

```
IdentityQueryEvents
| where Timestamp > ago(7d)
| where ActionType == "SamrQuerySuccess"
and isnotempty(AccountName)
| project QueryTime = Timestamp,
DeviceName, AccountName, Query,
QueryTarget
| join kind=inner (
DeviceProcessEvents
| where Timestamp > ago(7d)
| extend DeviceName =
toupper(trim(@"\..*$",DeviceName))
| where InitiatingProcessCommandLine
contains "net.exe"
| project ProcessCreationTime = Timestamp,
DeviceName, AccountName,
InitiatingProcessFileName ,
InitiatingProcessCommandLine
) on DeviceName, AccountName
| where ProcessCreationTime - QueryTime
between (-2m .. 2m)
| project QueryTime, DeviceName,
AccountName, InitiatingProcessFileName,
InitiatingProcessCommandLine, Query,
QueryTarget
```

Cloud Apps (MCAS)

Identify which files within the last 24 hours had more then 10 data access, download or deletion activities on MCAS-protected applications. Author: @MiladMSFT

```
AppFileEvents
 | where Timestamp > ago(1d)
 | summarize count() by FolderPath,
FileName, ActionType,
AccountDisplayName
 | where count_ > 10
```

Identity (Azure ATP)

Find Active Directory user accounts that have been inactive for more than 14 days. Author: @MiladMSFT

```
IdentityLogonEvents
| project Timestamp, AccountName,
DeviceName, LogonType
| summarize LastLogon = max(Timestamp)
by AccountName, LogonType, DeviceName
| where LastLogon < ago(14d)
```

Endpoint (Microsoft Defender ATP)

Find endpoints communicating to a specific domain Author: @maarten_goet

```
let Domain = "http://domain.com";
DeviceNetworkEvents
| where Timestamp > ago(7d) and RemoteUrl contains Domain
| project Timestamp, DeviceName, RemotePort, RemoteUrl
| top 100 by Timestamp desc
```

Finds PowerShell execution events that could involve a download Author: @MicrosoftMTP

```
union DeviceProcessEvents, DeviceNetworkEvents
| where Timestamp > ago(7d)
| where FileName in~ ("powershell.exe", "powershell_ise.exe")
| where ProcessCommandLine has_any("WebClient",
 "DownloadFile",
 "DownloadData",
 "DownloadString",
"WebRequest",
"Shellcode",
"http",
"https")
| project Timestamp, DeviceName, InitiatingProcessFileName,
InitiatingProcessCommandLine,
FileName, ProcessCommandLine, RemoteIP, RemoteUrl, RemotePort, RemoteIPType
| top 100 by Timestamp
```

Find created scheduled tasks Author: @maarten_goet

```
DeviceProcessEvents
| where FolderPath endswith "\\schtasks.exe" and ProcessCommandLine has "
/create " and AccountName != "system"
| where Timestamp > ago(7d)
```

Find possible clear text passwords in Windows registry. Author: @MicrosoftMTP

```
DeviceRegistryEvents
| where ActionType == "RegistryValueSet"
| where RegistryValueName == "DefaultPassword"
| where RegistryKey has @"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
| project Timestamp, DeviceName, RegistryKey
| top 100 by Timestamp
```

Lookup process executed from binary hidden in Base64 encoded file. Author: @MicrosoftMTP

```
DeviceProcessEvents
| where Timestamp > ago(14d)
| where ProcessCommandLine contains ".decode('base64')"
        or ProcessCommandLine contains "base64 --decode"
        or ProcessCommandLine contains ".decode64("
| project Timestamp , DeviceName , FileName , FolderPath , ProcessCommandLine ,
InitiatingProcessCommandLine
| top 100 by Timestamp
```

identify strings in process command lines which match Base64 encoding format, extract the string to a column called Base64,a nd decode it in a column called DecodedString. Author: @PowershellPoet

```
DeviceProcessEvents
| extend SplitLaunchString = split(ProcessCommandLine, " ")
| mvexpand SplitLaunchString
| where SplitLaunchString matches regex "^[A-Za-z0-9+/]{50,}[=]{0,2}$"
| extend Base64 = tostring(SplitLaunchString)
| extend DecodedString = base64_decodestring(Base64)
| where isnotempty(DecodedString)
```

identifies applications which leverage a command line pattern which matches the 7zip and WinRAR command line executables to create or update an archive when a password is specified. Author: @PowershellPoet

```
DeviceProcessEvents
| where ProcessCommandLine matches regex @"\s[aukfAUKF]\s.*\s-p"  // Basic
filter to look for launch string
| extend SplitLaunchString = split(ProcessCommandLine, ' ') // Split on the
space
| where array_length(SplitLaunchString) >= 5 and SplitLaunchString[1] in~
('a','u','k','f') // look for calls to archive or update an archive specifically
as the first argument
| mv-expand SplitLaunchString // cross apply the array
| where SplitLaunchString startswith "-p"  // -p is the password switch and is
immediately followed by a password without a space
| extend ArchivePassword = substring(SplitLaunchString, 2,
strlen(SplitLaunchString))
| project-reorder ProcessCommandLine, ArchivePassword // Promote these fields to
the left
```