

# API Microsoft Graph M365 - Guide Pratique Cybersecurite

Catégorie : Microsoft 365 Lecture : 7 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

*API Microsoft Graph M365 : authentication, endpoints, scripts PowerShell. Guide complet pour audit et monitoring automatisé Microsoft 365.*

---

Cette analyse detaillee de API Microsoft Graph M365 - Guide Pratique Cybersecurite s'appuie sur les retours d'experience d'equipes de securite confrontees quotidiennement aux menaces actuelles. Les methodologies presentees couvrent l'ensemble du cycle de vie de la securite, de la detection initiale a la remediation complete, en passant par l'investigation forensique et le durcissement des configurations. Les recommandations sont directement applicables dans les environnements de production et tiennent compte des contraintes operationnelles rencontrees par les equipes techniques sur le terrain. Les outils et techniques presentes ont ete valides dans des contextes reels d'incidents et de tests d'intrusion. La mise en oeuvre d'une strategie de defense en profondeur reste essentielle face a l'evolution constante du paysage des menaces, en combinant prevention, detection et capacite de reponse rapide aux incidents de securite.

Cette analyse technique de API Microsoft Graph M365 - Guide Pratique Cybersecurite s'appuie sur les retours d'experience d'equipes confrontees quotidiennement aux defis operationnels du domaine. Les methodologies presentees couvrent l'ensemble du cycle de vie, de la conception initiale au deploiement en production, en passant par les phases de test et de validation. Les recommandations sont directement applicables dans les environnements professionnels.

## Introduction à l'API Microsoft Graph pour l'audit M365

---

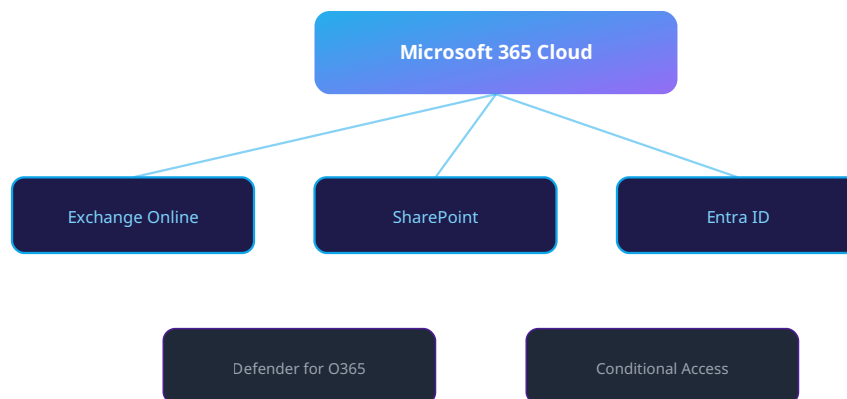
L'API Microsoft Graph représente la porte d'entrée unifiée vers l'écosystème Microsoft 365, Azure AD et Microsoft Cloud. Pour les experts en cybersécurité, elle constitue l'outil indispensable pour développer des solutions d'audit personnalisées, automatiser la surveillance des environnements et extraire des données critiques pour l'analyse de sécurité.

Avec plus de 1000 endpoints disponibles et une architecture RESTful moderne, Microsoft Graph permet d'accéder programmatiquement à l'ensemble des données de sécurité : journaux d'audit, authentifications, permissions, activités utilisateurs, et alertes de sécurité. Cette richesse fonctionnelle nécessite une expertise technique approfondie pour exploiter efficacement ses capacités.

### Avantages de l'API Graph pour l'audit :

- • **Accès unifié** : Une API pour tous les services M365
- • **Données temps réel** : Informations à jour et cohérentes
- • **Scalabilité** : Gestion de millions d'objets et d'événements

- • **Standardisation** : Format JSON, pagination, filtrage OData
- • **Sécurité intégrée** : OAuth 2.0, scopes granulaires



*Architecture Microsoft 365 - Services et sécurité*

Votre MFA est-il résistant aux attaques de type adversary-in-the-middle ?

## Authentification et gestion des permissions

L'authentification auprès de l'API Microsoft Graph repose sur OAuth 2.0 et Azure AD. Pour les scénarios d'audit, deux modes d'authentification sont disponibles : Application Permissions (sans utilisateur connecté) et Delegated Permissions (au nom d'un utilisateur). Le choix du mode détermine les données accessibles et les niveaux de sécurité.

### 1. Configuration de l'application Azure AD

#### Étapes de configuration :

1. Création d'une App Registration dans Azure AD
2. Configuration des API Permissions selon les besoins d'audit
3. Génération d'un secret client ou certificat
4. Consentement administrateur pour les permissions sensibles

```

# PowerShell - Configuration App Registration
$AppName = "M365-Audit-Tool"
$RequiredScopes = @(
    "AuditLog.Read.All",
    "Directory.Read.All",
    "SecurityEvents.Read.All",
    "Reports.Read.All",
    "User.Read.All"
)

# Création de l'application
$App = New-AzADApplication -DisplayName $AppName -ReplyUrls "http://localhost"

# Attribution des permissions
foreach ($Scope in $RequiredScopes) {
    $Permission = Get-AzADServicePrincipal -ApplicationId "00000003-0000-0000-
c000-000000000000" |
        Get-AzADServicePrincipalAppRole | Where-Object {$_.Value -eq $Scope}

    Add-AzADAppPermission -ObjectId $App.ObjectId -ApiId "00000003-0000-0000-
c000-000000000000" -PermissionId $Permission.Id -Type Role
}

```

## 2. Authentification par certificat (Recommandée)

### Avantages de l'authentification par certificat :

- • **Sécurité renforcée** : Pas de secrets en texte clair
- • **Rotation automatisée** : Gestion via Azure Key Vault
- • **Audit complet** : Traçabilité des accès
- • **Conformité** : Standards PKI d'entreprise

```

# PowerShell - Authentification par certificat
$TenantId = "your-tenant-id"
$ClientId = "your-client-id"
$CertThumbprint = "your-certificate-thumbprint"

# Connexion avec certificat
$Context =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Inst
ance.Profile.DefaultContext
$Token =
[Microsoft.Azure.Commands.Common.Authentication.AzureSession]::Instance.AuthenticationFact
ory.Authenticate($Context.Account, $Context.Environment, $TenantId, $null, "Never", $null,
"https://graph.microsoft.com/").AccessToken

$headers = @{
    'Authorization' = "Bearer $Token"
    'Content-Type' = 'application/json'
}

# Test de connexion
$Users = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/users?`$top=5" -Headers
$headers -Method Get

```

### 3. Gestion des permissions granulaires

#### Permissions d'audit essentielles :

##### Lecture des données :

- • AuditLog.Read.All
- • SecurityEvents.Read.All
- • Directory.Read.All
- • Reports.Read.All
- • Policy.Read.All

##### Actions administratives :

- • User.ReadWrite.All
- • RoleManagement.ReadWrite.Directory
- • Policy.ReadWrite.ConditionalAccess
- • SecurityActions.ReadWrite.All
- • ThreatIndicators.ReadWrite.OwnedBy

Element	Description	Priorite
<b>Prevention</b>	Mesures proactives de reduction de la surface d'attaque	Haute
<b>Detection</b>	Surveillance et alerting en temps reel	Haute
<b>Reponse</b>	Procedures d'incident response et remediation	Critique
<b>Recovery</b>	Plan de reprise et continuite d'activite	Moyenne

#### Notre avis d'expert

La prévention de fuite de données (DLP) dans Microsoft 365 est puissante sur le papier, mais son efficacité dépend entièrement de la qualité de la classification des données en amont. Nos missions montrent que moins de 20% des organisations ont une politique de classification opérationnelle.

### Endpoints d'audit essentiels pour la sécurité M365

L'API Microsoft Graph expose une multitude d'endpoints dédiés à l'audit et à la sécurité. La maîtrise de ces endpoints permet de construire des tableaux de bord personnalisés, d'automatiser la détection d'incidents et de générer des rapports de conformité détaillés.

## 1. Journaux d'audit et d'authentification

### Endpoints cruciaux :

```
# Audit des authentifications
GET https://graph.microsoft.com/v1.0/auditLogs/signIns
# Filtrage par risque élevé
GET https://graph.microsoft.com/v1.0/auditLogs/signIns?$filter=riskLevelDuringSignIn eq 'high'

# Activités d'administration
GET https://graph.microsoft.com/v1.0/auditLogs/directoryAudits
# Filtrage par type d'opération
GET https://graph.microsoft.com/v1.0/auditLogs/directoryAudits?$filter=activityDisplayName eq 'Add member to role'

# Audit des applications
GET https://graph.microsoft.com/v1.0/auditLogs/provisioning
# Échecs d'approvisionnement
GET https://graph.microsoft.com/v1.0/auditLogs/provisioning?$filter=provisioningStatusInfo/status eq 'failure'
```

#### Astuce avancée :

Utilisez les filtres OData pour optimiser les requêtes : `$filter`, `$select`, `$orderby`, `$top`. La pagination automatique permet de traiter de gros volumes de données efficacement.

## 2. Sécurité et détection des menaces

### Endpoints de sécurité :

```
# Alertes de sécurité
GET https://graph.microsoft.com/v1.0/security/alerts
# Alertes critiques non résolues
GET https://graph.microsoft.com/v1.0/security/alerts?$filter=severity eq 'high' and status ne 'resolved'

# Détection d'identités à risque
GET https://graph.microsoft.com/v1.0/identityProtection/riskyUsers
GET https://graph.microsoft.com/v1.0/identityProtection/userRiskHistory

# Événements de risque
GET https://graph.microsoft.com/v1.0/identityProtection/riskDetections
# Détections récentes
GET https://graph.microsoft.com/v1.0/identityProtection/riskDetections?$filter=detectedDateTime ge 2025-01-01T00:00:00Z

# Scores de sécurité
GET https://graph.microsoft.com/v1.0/security/secureScores
GET https://graph.microsoft.com/v1.0/security/securityActions
```

### 3. Gouvernance et compliance

#### Endpoints de conformité :

```
# Politiques de conditionnement d'accès
GET https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies
# Politiques désactivées (risque de sécurité)
GET https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies?$filter=state eq 'disabled'

# Rôles et permissions
GET https://graph.microsoft.com/v1.0/directoryRoles
GET https://graph.microsoft.com/v1.0/directoryRoleTemplates
# Membres des rôles administrateurs
GET https://graph.microsoft.com/v1.0/directoryRoles/{role-id}/members

# Applications et Service Principals
GET https://graph.microsoft.com/v1.0/applications
GET https://graph.microsoft.com/v1.0/servicePrincipals
# Applications avec permissions élevées
GET https://graph.microsoft.com/v1.0/servicePrincipals?$filter=appRoles/any(role:role/value eq 'Directory.ReadWrite.All')
```

#### Scripts de monitoring automatisé

---

L'automatisation du monitoring M365 via l'API Graph nécessite des scripts robustes capables de gérer la pagination, les erreurs et les limitations de débit. Voici des exemples de scripts production-ready pour les cas d'usage les plus courants.

## 1. Surveillance des connexions suspectes

```
# PowerShell - Détection de connexions anormales
function Get-SuspiciousSignIns {
    param(
        [int]$HoursAgo = 24,
        [string[]]$RiskLevels = @('high', 'medium')
    )

    $StartTime = (Get-Date).AddHours(-$HoursAgo).ToString('yyyy-MM-ddTHH:mm:ssZ')
    $Filter = "createdDateTime ge $StartTime and (" +
        ($RiskLevels | ForEach-Object { "riskLevelDuringSignIn eq '$_" }) -join '
or ' + ")"

    $Uri = "https://graph.microsoft.com/v1.0/auditLogs/signIns?
$filter=$Filter&$select=id,createdDateTime,userDisplayName,userPrincipalName,clientAppUs
ed,ipAddress,location,riskLevelDuringSignIn,riskDetails"

    $AllSignIns = @()

    do {
        try {
            $Response = Invoke-RestMethod -Uri $Uri -Headers $Headers -Method Get
            $AllSignIns += $Response.value
            $Uri = $Response.'@odata.nextLink'

            # Respect rate limiting
            Start-Sleep -Milliseconds 100
        }
        catch {
            Write-Warning "Erreur lors de la récupération: $($_.Exception.Message)"
            break
        }
    } while ($Uri)

    # Analyse et alertes
    $SuspiciousSignIns = $AllSignIns | Where-Object {
        $_.location.countryOrRegion -notin @('France', 'Belgium', 'Switzerland') -or
        $_.riskLevelDuringSignIn -eq 'high' -or
        ([datetime]$_.createdDateTime).Hour -lt 6 -or ([datetime]$_.createdDateTime).Hour
-gt 22
    }

    return $SuspiciousSignIns
}

# Exécution et export
$SuspiciousLogins = Get-SuspiciousSignIns -HoursAgo 24
$SuspiciousLogins | Export-Csv -Path "SuspiciousSignIns_$(Get-Date -Format
'yyyyMMdd_HHmms').csv" -NoTypeInfo
```

## 2. Audit des permissions administratives

```

# PowerShell - Audit complet des rôles admin
function Get-AdminRoleAudit {
    $AdminRoles = @(
        "Global Administrator",
        "Security Administrator",
        "Exchange Administrator",
        "SharePoint Administrator",
        "User Administrator"
    )

    $AuditResults = @()

    foreach ($RoleName in $AdminRoles) {
        # Récupération du rôle
        $Role = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/directoryRoles?`$filter=displayName eq '$RoleName'" -Headers $Headers

        if ($Role.value) {
            $RoleId = $Role.value[0].id

            # Membres du rôle
            $Members = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/directoryRoles/$RoleId/members" -Headers $Headers

            foreach ($Member in $Members.value) {
                # Détails utilisateur
                $UserDetails = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/users/$($Member.id)?`$select=id,displayName,userPrincipalName,accountEnabled,lastSignInDateTime,createdDateTim e" -Headers $Headers

                # Dernières authentifications
                $RecentSignIns = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/auditLogs/signIns?`$filter=userId eq '$($Member.id)'" -Headers $Headers

                $AuditResults += [PSCustomObject]@{
                    RoleName = $RoleName
                    UserName = $UserDetails.displayName
                    UserPrincipalName = $UserDetails.userPrincipalName
                    AccountEnabled = $UserDetails.accountEnabled
                    LastSignIn = $UserDetails.lastSignInDateTime
                    AccountCreated = $UserDetails.createdDateTime
                    RecentSignInCount = $RecentSignIns.value.Count
                    LastSignInIP = $RecentSignIns.value[0].ipAddress
                    LastSignInLocation = $RecentSignIns.value[0].location.city
                }

                Start-Sleep -Milliseconds 200
            }
        }
    }

    return $AuditResults
}

# Génération du rapport
$AdminAudit = Get-AdminRoleAudit
$AdminAudit | Export-Csv -Path "AdminRoleAudit_$(Get-Date -Format 'yyyyMMdd').csv" -NoTypeInformation

# Alertes pour comptes suspects

```

```
$SuspiciousAdmins = $AdminAudit | Where-Object {
    $_.LastSignIn -eq $null -or
    ([datetime]$_.LastSignIn) -lt (Get-Date).AddDays(-90) -or
    $_.AccountEnabled -eq $false
}

if ($SuspiciousAdmins) {
    Write-Warning "Comptes administrateurs suspects détectés: $($SuspiciousAdmins.Count)"
}
```

### 3. Monitoring des applications OAuth

```

# PowerShell - Audit applications OAuth suspects
function Get-OAuthApplicationAudit {
    # Applications avec permissions dangereuses
    $DangerousPermissions = @(
        'Directory.ReadWrite.All',
        'Mail.ReadWrite',
        'Files.ReadWrite.All',
        'User.ReadWrite.All',
        'RoleManagement.ReadWrite.Directory'
    )

    $AllApps = @()
    $Uri = "https://graph.microsoft.com/v1.0/servicePrincipals?`
`$select=id,appId,displayName,publisherName,createdDateTime,appRoles,oauth2PermissionScope
s"

    do {
        $Response = Invoke-RestMethod -Uri $Uri -Headers $Headers
        $AllApps += $Response.value
        $Uri = $Response.'@odata.nextLink'
        Start-Sleep -Milliseconds 100
    } while ($Uri)

    $SuspiciousApps = @()

    foreach ($App in $AllApps) {
        $HasDangerousPermissions = $false
        $DangerousPermsFound = @()

        # Vérification des permissions d'application
        foreach ($AppRole in $App.appRoles) {
            if ($AppRole.value -in $DangerousPermissions) {
                $HasDangerousPermissions = $true
                $DangerousPermsFound += $AppRole.value
            }
        }

        # Vérification des permissions déléguées
        foreach ($Scope in $App.oauth2PermissionScopes) {
            if ($Scope.value -in $DangerousPermissions) {
                $HasDangerousPermissions = $true
                $DangerousPermsFound += $Scope.value
            }
        }

        # Critères de suspicion
        $IsSuspicious = $HasDangerousPermissions -or
            $App.publisherName -eq $null -or
            $App.publisherName -eq '' -or
            ([datetime]$App.createdDateTime) -gt (Get-Date).AddDays(-7)

        if ($IsSuspicious) {
            # Récupération des consentements
            $AppConsents = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/
oauth2PermissionGrants?`$filter=clientId eq '$($App.id)'" -Headers $Headers

            $SuspiciousApps += [PSCustomObject]@{
                DisplayName = $App.displayName
                AppId = $App.appId
                PublisherName = $App.publisherName
                CreatedDateTime = $App.createdDateTime
                DangerousPermissions = ($DangerousPermsFound -join ', ')
            }
        }
    }
}

```

```

        ConsentCount = $AppConsents.value.Count
        LastConsentDate = ($AppConsents.value | Sort-Object createdDateTime
-Descending | Select-Object -First 1).createdDateTime
        SuspicionReasons = @(
            if ($HasDangerousPermissions) { "Permissions élevées" }
            if (!$App.publisherName) { "Éditeur inconnu" }
            if ([datetime]$App.createdDateTime -gt (Get-Date).AddDays(-7))
{ "Application récente" }
        ) -join ', '
    }
}

    Start-Sleep -Milliseconds 50
}

return $SuspiciousApps
}

# Exécution de l'audit
$OAuthAudit = Get-OAuthApplicationAudit
$OAuthAudit | Export-Csv -Path "OAuthApplicationAudit_$(Get-Date -Format 'yyyyMMdd').csv"
-NoTypeInfoInformation

# Rapport de synthèse
Write-Host "Applications OAuth auditées: $($AllApps.Count)" -ForegroundColor Green
Write-Host "Applications suspectes: $($OAuthAudit.Count)" -ForegroundColor $
(if($OAuthAudit.Count -gt 0) {'Red'} else {'Green'})

```

### Cas concret

Les campagnes de phishing via Microsoft Teams se sont multipliées en 2024, avec des attaquants créant des tenants externes pour envoyer des messages directement aux employés ciblés. L'exploitation de la fédération Teams par défaut a permis de contourner les protections email traditionnelles.

Avez-vous vérifié les permissions effectives de vos comptes de service Azure AD ?

## Analyse et corrélation des données d'audit

L'exploitation avancée de l'API Graph nécessite des techniques d'analyse et de corrélation poussées pour identifier les patterns d'attaque, détecter les anomalies comportementales et générer des insights de sécurité actionables.

### 1. Corrélation d'événements multi-sources

#### Techniques de corrélation :

- • **Timeline analysis** : Chronologie des événements par utilisateur/IP
- • **Behavioral clustering** : Groupement par patterns similaires
- • **Geolocation correlation** : Détection de voyages impossibles
- • **Risk score aggregation** : Calcul de scores de risque composites

```

# PowerShell - Corrélation d'événements
function Invoke-SecurityEventCorrelation {
    param(
        [datetime]$StartDate = (Get-Date).AddDays(-7),
        [datetime]$EndDate = (Get-Date)
    )

    # Collecte des données multi-sources
    $SignIns = Get-GraphData -Endpoint "auditLogs/signIns" -StartDate $StartDate -EndDate $EndDate
    $DirectoryAudits = Get-GraphData -Endpoint "auditLogs/directoryAudits" -StartDate $StartDate -EndDate $EndDate
    $SecurityAlerts = Get-GraphData -Endpoint "security/alerts" -StartDate $StartDate -EndDate $EndDate

    # Corrélation par utilisateur
    $UserTimelines = @{}

    foreach ($SignIn in $SignIns) {
        $UserId = $SignIn.userId
        if (-not $UserTimelines[$UserId]) {
            $UserTimelines[$UserId] = @{
                User = $SignIn.userPrincipalName
                SignIns = @()
                AdminActions = @()
                Alerts = @()
                RiskScore = 0
            }
        }

        $UserTimelines[$UserId].SignIns += $SignIn

        # Calcul du score de risque
        switch ($SignIn.riskLevelDuringSignIn) {
            'high' { $UserTimelines[$UserId].RiskScore += 50 }
            'medium' { $UserTimelines[$UserId].RiskScore += 20 }
            'low' { $UserTimelines[$UserId].RiskScore += 5 }
        }
    }

    # Corrélation avec les actions administratives
    foreach ($Audit in $DirectoryAudits) {
        $UserId = $Audit.initiatedBy.user.id
        if ($UserTimelines[$UserId]) {
            $UserTimelines[$UserId].AdminActions += $Audit

            # Privilège administratif = risque supplémentaire
            if ($Audit.category -eq 'RoleManagement') {
                $UserTimelines[$UserId].RiskScore += 30
            }
        }
    }

    # Corrélation avec les alertes de sécurité
    foreach ($Alert in $SecurityAlerts) {
        foreach ($Entity in $Alert.userStates) {
            $UserId = $Entity.userPrincipalName
            if ($UserTimelines[$UserId]) {
                $UserTimelines[$UserId].Alerts += $Alert
                $UserTimelines[$UserId].RiskScore += 40
            }
        }
    }
}

```

```

}

# Analyse des patterns suspects
$SuspiciousUsers = $UserTimelines.Values | Where-Object {
    $_.RiskScore -gt 75 -or
    ($_.SignIns | Group-Object ipAddress).Count -gt 10 -or
    ($_.AdminActions.Count -gt 0 -and $_.RiskScore -gt 20)
}

return $SuspiciousUsers
}

# Analyse de corrélation
$CorrelationResults = Invoke-SecurityEventCorrelation
$CorrelationResults | Sort-Object RiskScore -Descending |
    Select-Object User, RiskScore, @{n='SignInCount';e={$_.SignIns.Count}},
    @{n='AdminActionCount';e={$_.AdminActions.Count}}, @{n='AlertCount';e={$_.Alerts.Count}} |
    Export-Csv -Path "SecurityCorrelation_$(Get-Date -Format 'yyyyMMdd').csv"
-NoTypeInfoation

```

## 2. Détection d'anomalies comportementales

### Algorithmes de détection :

- • **Statistical outliers** : Détection d'écarts statistiques
- • **Time-series analysis** : Analyse des patterns temporels
- • **Machine learning clustering** : Groupement non supervisé
- • **Baseline deviation** : Écart par rapport au comportement normal

## Automatisation avancée et intégration SIEM

L'intégration de l'API Microsoft Graph dans des workflows automatisés permet de créer des solutions de sécurité proactives. L'automatisation couvre la collecte de données, l'analyse en temps réel, la génération d'alertes et la réponse aux incidents.

### 1. Pipeline de données pour SIEM

#### Architecture de collecte :

- • **Collecte incrémentale** : Delta queries pour optimiser les performances
- • **Format CEF/LEEF** : Standardisation pour ingestion SIEM
- • **Compression et chiffrement** : Transport sécurisé des données
- • **Retry logic** : Gestion robuste des erreurs

```

# PowerShell - Pipeline SIEM automatisé
class GraphToSIEMPipeline {
    [string]$TenantId
    [string]$ClientId
    [string]$ClientSecret
    [string]$SIEMEndpoint
    [hashtable]$DeltaTokens

    GraphToSIEMPipeline([string]$TenantId, [string]$ClientId, [string]$ClientSecret,
[string]$SIEMEndpoint) {
        $this.TenantId = $TenantId
        $this.ClientId = $ClientId
        $this.ClientSecret = $ClientSecret
        $this.SIEMEndpoint = $SIEMEndpoint
        $this.DeltaTokens = @{}
    }

    [object] GetAccessToken() {
        $Body = @{
            client_id = $this.ClientId
            client_secret = $this.ClientSecret
            scope = "https://graph.microsoft.com/.default"
            grant_type = "client_credentials"
        }

        $Response = Invoke-RestMethod -Uri "https://login.microsoftonline.com/$
($this.TenantId)/oauth2/v2.0/token" -Method Post -Body $Body
        return @{ Authorization = "Bearer $($Response.access_token)" }
    }

    [object[]] GetDeltaData([string]$Endpoint) {
        $Headers = $this.GetAccessToken()
        $AllData = @()

        # Utilisation du delta token s'il existe
        if ($this.DeltaTokens[$Endpoint]) {
            $Uri = $this.DeltaTokens[$Endpoint]
        } else {
            $Uri = "https://graph.microsoft.com/v1.0/$Endpoint/delta"
        }

        do {
            try {
                $Response = Invoke-RestMethod -Uri $Uri -Headers $Headers -Method Get
                $AllData += $Response.value

                # Mise à jour du delta token
                if ($Response.'@odata.deltaLink') {
                    $this.DeltaTokens[$Endpoint] = $Response.'@odata.deltaLink'
                    break
                }

                $Uri = $Response.'@odata.nextLink'
                Start-Sleep -Milliseconds 100
            }
            catch {
                Write-Error "Erreur lors de la récupération delta: $
($_.Exception.Message)"
                break
            }
        } while ($Uri)
    }
}

```

```

    return $AllData
}

[void] SendToSIEM([object[]]$Data, [string]$EventType) {
    foreach ($Event in $Data) {
        # Conversion au format CEF
        $CEFEvent = $this.ConvertToCEF($Event, $EventType)

        # Envoi vers SIEM
        try {
            Invoke-RestMethod -Uri $this.SIEMEndpoint -Method Post -Body $CEFEvent
            -ContentType "application/json"
        }
        catch {
            Write-Warning "Échec envoi SIEM pour événement $($Event.id): $
            ($_.Exception.Message)"
        }
    }
}

[string] ConvertToCEF([object]$Event, [string]$EventType) {
    $CEF = @{
        timestamp = $Event.createdDateTime
        event_type = $EventType
        source_system = "Microsoft 365"
        raw_data = ($Event | ConvertTo-Json -Depth 5 -Compress)
    }

    # Enrichissement selon le type d'événement
    switch ($EventType) {
        "SignIn" {
            $CEF.user_name = $Event.userPrincipalName
            $CEF.source_ip = $Event.ipAddress
            $CEF.risk_level = $Event.riskLevelDuringSignIn
            $CEF.location = "$($Event.location.city), $
            ($Event.location.countryOrRegion)"
        }
        "DirectoryAudit" {
            $CEF.activity = $Event.activityDisplayName
            $CEF.initiated_by = $Event.initiatedBy.user.userPrincipalName
            $CEF.target_resources = ($Event.targetResources | ForEach-Object
            { $_.displayName }) -join ', '
        }
    }

    return ($CEF | ConvertTo-Json -Compress)
}

[void] RunPipeline() {
    Write-Host "Démarrage du pipeline Graph vers SIEM..." -ForegroundColor Green

    # Collecte des différents types de données
    $Endpoints = @(
        @{Name = "auditLogs/signIns"; Type = "SignIn"},
        @{Name = "auditLogs/directoryAudits"; Type = "DirectoryAudit"},
        @{Name = "security/alerts"; Type = "SecurityAlert"}
    )

    foreach ($Endpoint in $Endpoints) {
        Write-Host "Traitement de $($Endpoint.Name)..." -ForegroundColor Yellow

        $Data = $this.GetDeltaData($Endpoint.Name)
    }
}

```

```

        if ($Data.Count -gt 0) {
            $this.SendToSIEM($Data, $Endpoint.Type)
            Write-Host "Envoyé $($Data.Count) événements de type $($Endpoint.Type)"
            -ForegroundColor Green
        }
    }

    Write-Host "Pipeline terminé avec succès." -ForegroundColor Green
}

# Utilisation du pipeline
$Pipeline = [GraphToSIEMPipeline]::new($TenantId, $ClientId, $ClientSecret, "https://your-siem.internal/api/events")
$Pipeline.RunPipeline()

```

## 2. Alerting intelligent et réponse automatisée

### Mécanismes d'alerte :

- • **Webhooks temps réel** : Notifications instantanées via Graph subscriptions
- • **Severity-based routing** : Escalade selon la criticité
- • **Automated remediation** : Actions de réponse programmées
- • **Context enrichment** : Ajout d'informations contextuelles

## Sécurité et bonnes pratiques

---

L'utilisation de l'API Microsoft Graph en production nécessite l'implémentation de mesures de sécurité rigoureuses pour protéger les accès, sécuriser les données et respecter les principes de moindre privilège.

### 1. Sécurisation des accès API

#### Checklist de sécurité :

- • **Certificate-based auth** : Éviter les secrets clients en texte clair
- • **Key Vault integration** : Stockage sécurisé des certificats et secrets
- • **Conditional Access** : Restriction des accès par IP/device
- • **Audit logging** : Traçabilité complète des appels API
- • **Rate limiting** : Gestion des quotas et throttling
- • **Error handling** : Gestion sécurisée des erreurs sans exposition de données

### 2. Protection des données sensibles

#### Mesures de protection :

- • **Data minimization** : Collecte uniquement des données nécessaires
- • **Encryption at rest** : Chiffrement des données stockées
- • **Encryption in transit** : TLS 1.3 obligatoire
- • **Data retention** : Politique de rétention conforme GDPR
- • **Access controls** : Contrôles d'accès granulaires aux données

## Articles connexes

---

Approfondissez vos connaissances en sécurité Microsoft 365 avec ces guides experts :

### **Automatisation Audit PowerShell**

Automatisez l'audit de sécurité Microsoft 365 avec PowerShell et l'API Graph pour des workflows d'audit avancés.

### **Corrélation des Journaux M365**

Techniques avancées de corrélation et d'analyse des logs pour détecter les menaces dans M365.

### **Détection Compromission Identités**

Déterminez et prévenez les attaques de compromission d'identités dans Azure AD et Microsoft 365.

### **Threat Hunting M365**

Techniques de threat hunting avec Microsoft Defender et Sentinel pour traquer proactivement les menaces.

## Cas d'usage avancés et perspectives

---

L'API Microsoft Graph ouvre des perspectives illimitées pour l'innovation en matière de sécurité M365. Les cas d'usage évoluent constamment avec l'enrichissement de l'API et l'émergence de nouvelles menaces.

### Cas d'usage émergents 2025 :

#### **IA et Machine Learning**

- Détection d'anomalies comportementales avancées
- Prédiction de risques de sécurité
- Classification automatique des incidents
- Recommandations de sécurisation personnalisées

#### **Threat Hunting**

- Corrélation multi-tenant pour MSP
- Hunt queries automatisées
- IOC enrichment en temps réel
- Threat intelligence integration

#### **Real-time Response**

- Isolation automatique de comptes compromis
- Révocation instantanée de sessions
- Containment orchestré des incidents

- • Communication de crise automatisée

### **Advanced Analytics**

- • Dashboards temps réel interactifs
- • Métriques de sécurité personnalisées
- • Reporting exécutif automatisé
- • Benchmarking sectoriel

### **Recommandations d'implémentation :**

- **Commencer petit** : Preuves de concept sur cas d'usage critiques
- **Itérer rapidement** : Développement agile avec feedback continu
- **Sécuriser d'abord** : Security by design dès la conception
- **Monitorer l'usage** : Surveillance des performances et quotas
- **Former les équipes** : Montée en compétences techniques continues

L'API Microsoft Graph est l'outil incontournable pour les experts en sécurité M365. Sa maîtrise technique ouvre la voie à des solutions d'audit innovantes et à une posture de sécurité proactive dans l'écosystème Microsoft Cloud.

#### **Ressources open source associées :**

- KQLHunter — Générateur de requêtes KQL avec IA (Python)
- m365-security-fr — Dataset sécurité M365 (HuggingFace)
- oauth-api-security-fr — Dataset sécurité OAuth/API (HuggingFace)

## Questions fréquentes

---

### **Comment ce sujet impacte-t-il la sécurité des organisations ?**

Ce sujet a un impact significatif sur la sécurité des organisations car il touche aux fondamentaux de la protection des systèmes d'information. Les entreprises doivent évaluer leur exposition, mettre en place des mesures préventives adaptées et former leurs équipes pour faire face aux risques associés à cette problématique.

### **Quelles sont les bonnes pratiques recommandées par les experts ?**

Les experts recommandent une approche basée sur les risques, incluant l'évaluation régulière de la posture de sécurité, la mise en place de contrôles techniques et organisationnels, la formation continue des équipes et l'adoption des référentiels de sécurité reconnus comme ceux du NIST, de l'ANSSI et de l'OWASP.

## Pourquoi est-il important de se former sur ce sujet en 2026 ?

En 2026, la maîtrise de ce sujet est devenue incontournable face à l'évolution constante des menaces et des exigences réglementaires. Les professionnels de la cybersécurité doivent maintenir leurs compétences à jour pour protéger efficacement les actifs numériques de leur organisation et répondre aux obligations de conformité.

**Sources et références :** [Microsoft Security Docs](#) · [CERT-FR](#)

## Conclusion

---

Cet article a couvert les aspects essentiels de l'introduction à l'API Microsoft Graph pour l'audit M365, l'authentification et la gestion des permissions, les endpoints d'audit essentiels pour la sécurité M365. La mise en pratique de ces recommandations permet de renforcer significativement la posture de sécurité de votre organisation.

---

**Ayi NEDJIMI Consultants** — Expert cybersécurité offensive & intelligence artificielle

[ayinedjimi-consultants.fr](https://ayinedjimi-consultants.fr) · [ayi@ayinedjimi-consultants.fr](mailto:ayi@ayinedjimi-consultants.fr)

© 2025 — Reproduction interdite sans autorisation.