EXCELLIUM

# O365 …
# Pitting the Theory Against the Practice

FIRST & AfricaCERT Symposium
Open-source Tools and CSIRT Success Stories

## Context

- **Business Email Compromise in O365**
- **Toolbox**: all open-source tools
- **To collect**
  - Azure Powershell modules and script
  - ANSSI framework
- **To parse and automate**
  - Jq (json)
  - Csvcut (… csv)

## Objective of the incident response …

- Root cause
  - **MFA bypassed**
  - Malicious application registered, for which user can give **consent**, with too much **permissions**
  - **Guest/partner** access abused
  - Anti-spoofing/anti-spam **policy bypassed** (simple spoofing, IA)
  - And all we don't know yet ☺

- Extent of the compromise
  - New **inbox rule** created, forwarding to attacker
  - Access by rogue **application** via tokens
  - Access to other services (Azure resources, SharePoint, Teams, Azure AD, …)

- Containment
  - Identify impacted **users**
  - Identify rogue **roles/accounts**
  - **Revoke** tokens/change password (not that easy)

## … Versus the complexity of the platform

- Number of **admin consoles** (more than **15** !)
- **Licensing** impacts logs retention and features (Identity Protection, policies, cmdlets)
- **Variety of logs** (sign-ins, audit logs, activity logs, risky users, risky sign-ins, …)
  - And **variety of results** (GUI versus Powershell): limitations, fields, latency, corrupted logs
- More than web services
  - Legacy/basic authentication protocols (**MFA bypass**)
  - OAuth2 applications (called “registered applications”): **new form of phishing**
  - **Guest/partner** access
  - **Add-in** (additional applications, no logs)
  - **Sharing** documents

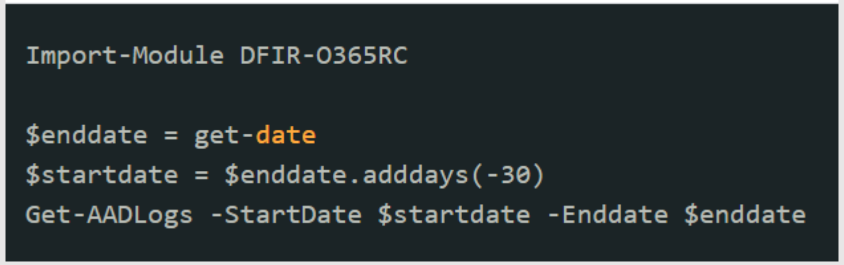
## Logs of interest

| Microsoft name | Description | Retention | How to collect |
|---|---|---|---|
| Azure AD sign-ins | All logins (Azure AD, O365 apps, admin) | 1 week/1 month | GUI/powershell |
| Risky users/risky sign-ins | Abnormal logins/unusual behavior reports | N/A | GUI |
| Azure AD audit logs | Tenant management (users, groups, applications registered, admin operations, …) | 1 week/1 month | GUI/powershell |
| O365 audit logs | Operations on apps (mailbox, office, teams, OneDrive, …) | 3 months | GUI/powershell |
| Message Trace Reports | Emails sent and received | 3 months | GUI |
| Registered applications | List and permissions | N/A | GUI/powershell |

- Powershell instead of GUI
  - **Scriptable**
  - **Homogeneity** of the results
  - Better **retention**
  - **Not corrupted**
  - No **latency** between user' actions and availability in logs

## Azure AD sign-ins

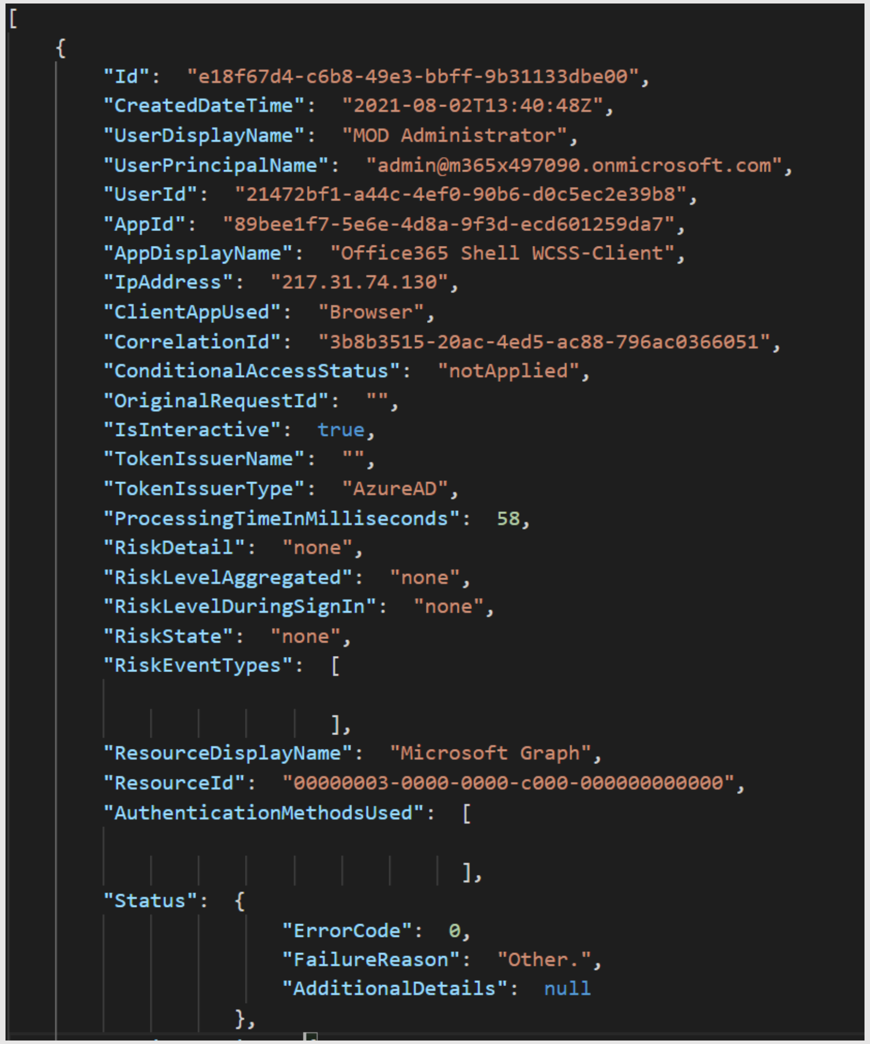
- Extract
  - https://github.com/ANSSI-FR/DFIR-O365RC
  - Json output



```
Import-Module DFIR-O365RC

$enddate = get-date
$startdate = $enddate.adddays(-30)
Get-AADLogs -StartDate $startdate -Enddate $enddate
```

- Parse
  - *jq -r '.[] | [.createdDateTime, .userPrincipalName, .ipAddress, .appDisplayName, **.appId**, **.resourceId**, .resourceDisplayName, .clientAppUsed, .status.errorCode, .status.failureReason, .deviceDetail.operatingSystem, .deviceDetail.browser, .location.city, .location.countryOrRegion, .appliedConditionalAccessPolicies[].displayName, .appliedConditionalAccessPolicies[].result] | @csv'*
  - *"appID" = **source** / "resourceId" = **destination***



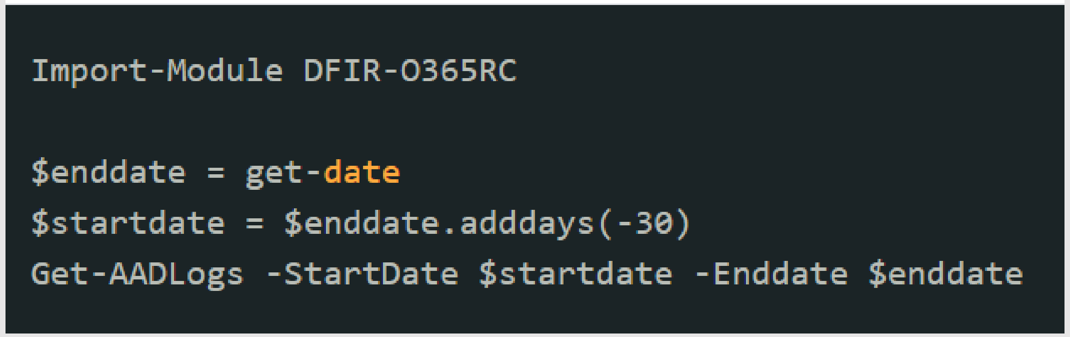
```
[
    {
        "Id":  "e18f67d4-c6b8-49e3-bbff-9b31133dbe00",
        "CreatedDateTime":  "2021-08-02T13:40:48Z",
        "UserDisplayName":  "MOD Administrator",
        "UserPrincipalName":  "admin@m365x497090.onmicrosoft.com",
        "UserId":  "21472bf1-a44c-4ef0-90b6-d0c5ec2e39b8",
        "AppId":  "89bee1f7-5e6e-4d8a-9f3d-ecd601259da7",
        "AppDisplayName":  "Office365 Shell WCSS-Client",
        "IpAddress":  "217.31.74.130",
        "ClientAppUsed":  "Browser",
        "CorrelationId":  "3b8b3515-20ac-4ed5-ac88-796ac0366051",
        "ConditionalAccessStatus":  "notApplied",
        "OriginalRequestId":  "",
        "IsInteractive":  true,
        "TokenIssuerName":  "",
        "TokenIssuerType":  "AzureAD",
        "ProcessingTimeInMilliseconds":  58,
        "RiskDetail":  "none",
        "RiskLevelAggregated":  "none",
        "RiskLevelDuringSignIn":  "none",
        "RiskState":  "none",
        "RiskEventTypes":  [

                           ],
        "ResourceDisplayName":  "Microsoft Graph",
        "ResourceId":  "00000003-0000-0000-c000-000000000000",
        "AuthenticationMethodsUsed":  [

                                      ],
        "Status":  {
                       "ErrorCode":  0,
                       "FailureReason":  "Other.",
                       "AdditionalDetails":  null
                   },
```

# Azure AD audit logs

- Extract
  - https://github.com/ANSSI-FR/DFIR-O365RC
  - Json output
  - Same command as sign-ins: it collects all in one



```
Import-Module DFIR-O365RC

$enddate = get-date
$startdate = $enddate.adddays(-30)
Get-AADLogs -StartDate $startdate -Enddate $enddate
```

- Parse
  - **Overview** of who did what
    *jq -r '.[] | [.initiatedBy.user.userPrincipalName, .activityDisplayName]| @csv' | sort -u*
  - **Timeline** with more details (typically who gave consent to an application)
    *jq –r ‘select(.activityDisplayName == “Consent to application”) | [.activityDateTime, .initiatedBy.user.ipAddress, .initiatedBy.user.userPrincipalName, .targetResources[].displayName] | @csv’*
  - **Generic form** to extract a timeline related to the activity “X” (fieldN of interest for this activity)
    *jq –r ‘select(.activityDisplayName == “X”) | [.activityDateTime, .<field1>, .<field2>] | @csv’*

## Applications registered: list and permissions

- List **applications**
  - Module AzureADPreview, cmdlet Get-AzureADServicePrincipal
  - Json output
- List **permissions**
  - Get-AzureADPSPermissions.ps1: https://gist.github.com/psignoret/41793f8c6211d2df5051d77ca3728c09
  - From https://docs.microsoft.com/en-us/microsoft-365/security/office-365-security/detect-and-remediate-illicit-consent-grants?view=o365-worldwide
  - Csv output

```
Import-Module AzureADPreview
Connect-AzureAD

Get-AzureADServicePrincipal -All:$true | ConvertTo-Json | Out-File -Encoding utf8 -FilePath AllApplications.json
.\Get-AzureADPSPermissions.ps1 -ShowProgress | Export-csv -Path "Permissions.csv" -NoTypeInformation
```

- Audit logs limited to **activity** involving applications
  - DFIR-O365RC

```
Import-Module DFIR-O365RC

$enddate = get-date
$startdate = $enddate.adddays(-30)
Get-AADApps -StartDate $startdate -Enddate $enddate
```

# O365 audit logs

- Extract
  - https://github.com/ANSSI-FR/DFIR-O365RC
  - Json output

```
Import-Module DFIR-O365RC

$enddate = get-date
$startdate = $enddate.adddays(-90)
Get-O365Full -StartDate $startdate -Enddate $enddate -RecordSet "All"
```

```
Import-Module DFIR-O365RC

$enddate = get-date
$startdate = $enddate.adddays(-90)
Search-O365 -StartDate $startdate -Enddate $enddate -UserIds "email1@domain.tld", "email2@domain.tld"
```

- Parse
  - **Overview** of who did what
    *jq -r '. | [.Operation, .Workload, .UserId ]| @csv' |sort –u*
  - Examples of **Operations**
    - Add-MailboxPermission, MailboxLogin, MailItemsAccessed, Create, Sent, New-InboxRule, Set-InboxRule, UpdateInboxRules
    - AnonymousLinkCreated, FileAccessed, FileCopied, FileDeleted, FileModified, FileMoved, FileDownloaded, FileUploaded
  - **Timeline** for the operation "X"
    *jq –r 'select(.Operation == "X") | [.eventDateTime, .field1, .field2, .field3, …, .fieldN] | @csv'*

## Message Trace Reports

- Extract
  - GUI
  - **Asynchronous** process
  - Choose “**Extended Report**”

- Caution
  - Up to **24 hours** to see last emails
  - **Not exhaustive**: official answer from Microsoft **“blocked emails not included until explicitly requested in the filter”**
  - Body **spoofing** difficult to detect (sender = body, not envelop)

- Parse
  - **Timeline**
    csvcut -c date_time_utc,original_client_ip,client_hostname,server_ip,server_hostname,message_id,reference,
    directionality,sender_address,return_path,recipient_address,message_subject,total_bytes

  - Spoofed email: “client_hostname”, “server_hostname”, “server_ip”, “directionality”, “sender” will point **inconsistency**
  - Email **headers** (SPF/DKIM/DMARC/Spam checks): fields “message_info”, “custom_data”

## In a “nutshell”

- Data acquisition
  - PowerShell to collect **logs,** https://github.com/ANSSI-FR/DFIR-O365RC
    - It **works** and handles token refresh, API throttling, limited number of results per query
    - Ensure **content stability** (structure, fields, field names) to then automate parsing
  - PowerShell to collect **configuration** (application IDs, application permissions)
  - Collect Message Trace Reports from **GUI** (beware of latency)
  - For other configurations: … **screenshots** (user consent, policies)

- Logs analysis
  - **Fields vary** with Operations/Activity, but **automation possible** thanks to stable logs collection
  - **jq**: https://stedolan.github.io/jq/
  - **Csvcut**: https://csvkit.readthedocs.io/en/latest/scripts/csvcut.html

- Biggest known caveats
  - Web logins: **hard to identify** the primary connection of the attacker (1 login = ~30 lines of logs)
  - Application ID **puzzle** (~30% of IDs neither in Tenant list, nor documented by Microsoft)
  - Message Trace Report latency and still, not **exhaustive**

# Thank you

*contact: cert@excellium-services.com*