Skrypty audytowe Active Directory

Poziom: zaawansowany **Dotyczy: Windows Server 2016+, moduł ActiveDirectory (RSAT)**

Czas: ~40 min

Sześć gotowych skryptów audytowych do skopiowania: od zagnieżdżonych grup między domenami, przez pełny zrzut DNS i GPO, po klucze BitLocker i wymuszenie synchronizacji z Entra ID.

Skróty użyte na tej stronie (14)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CSV	w Windowsie: Cluster Shared Volume, dysk widoczny naraz dla wszystkich węzłów klastra; w danych: Comma-Separated Values, plik z wartościami oddzielonymi przecinkami
DC	Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników
DN	Distinguished Name: pełny, jednoznaczny adres jednego obiektu w katalogu LDAP, na przykład konta albo grupy
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
GPMC	Group Policy Management Console: konsola do zarządzania politykami GPO w domenie
GPO	Group Policy Object: zestaw ustawień wypychanych z Active Directory na komputery
HTML	HyperText Markup Language: format tekstowy stron WWW, tu używany jako format raportu GPO do przeglądania w przeglądarce
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
OU	Organizational Unit: folder w Active Directory grupujący konta i komputery
RSAT	Remote Server Administration Tools: dodatkowe narzędzia administracyjne Windows do zarządzania AD, DNS i GPO ze stacji roboczej
SMTP	Simple Mail Transfer Protocol: protokół wysyłania poczty
XML	eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml

Fiszka

TYLKO ODCZYT

Audyt grupy z zagnieżdżoną grupą z innej domeny

```
# Pytaj użytkownika o nazwę grupy, kontroler domeny (DC) i poświadczenia
$Group = Read-Host -Prompt "Podaj nazwę grupy"
$DC = Read-Host -Prompt "Podaj kontroler domeny bądź jej nazwę (NazwaKontroleraDC / NazwaDomeny)"
$Credential = Get-Credential -Message "Podaj poświadczenia użytkownika do dostępu do domeny $DC"

# Pobierz członków grupy z określonego kontrolera domeny
$membership = Get-ADGroup $Group -Properties Member -Server $DC -Credential $Credential
$membership.Member | Group-Object { ($_ -split '(?=DC=)', 2)[1] } |
ForEach-Object {
    [adsi]$ldap = 'LDAP://{0}' -f $_.Name
    [string]$domain = $ldap.Name

    foreach ($member in $_.Group) {
        $obj = Get-ADObject $member -Server $domain -Credential $Credential
        [pscustomobject]@{
            MemberOf       = $membership.Name
            Domain         = $domain
            Name           = $obj.Name
            DSN            = $obj.DistinguishedName
            SamAccountName = $obj.SamAccountName
            ObjectClass    = $obj.ObjectClass
        }
    }
}
```

Komendę uruchamiamy jako Administrator w PowerShell na systemie z zainstalowanymi przystawkami RSAT. Rozwiązujemy tu problem, który zwykły `Get-ADGroupMember` nie obsługuje wprost: grupa ma wśród członków obiekty z innej domeny w tym samym lesie, a `Member` zwraca dla nich pełny `DistinguishedName` zamiast gotowego obiektu. Grupujemy członków po fragmencie DN odpowiadającym domenie (`(?=DC=)` jako wyrażenie regularne z pozytywnym wyprzedzeniem, dzielące napis tuż przed każdym „DC="), budujemy z tego fragmentu

ścieżkę LDAP, odczytujemy jej rzeczywistą nazwę domeny przez `[adsi]`, i dopiero z tak ustaloną domeną odpytujemy `Get-ADObject` dla każdego członka osobno.

Przykład:

```
MemberOf       Domain                  Name
DSN                                                    SamAccountName  ObjectClass
--------       ------                  ----           ---
                                                          --------------  -----------
GrupaProjektowa ad.zielona.example     Jan Kowalski  CN=Jan
Kowalski,OU=Users,DC=ad,DC=zielona,DC=example        jkowalski       user
GrupaProjektowa filia.ad.zielona.example Anna Nowak  CN=Anna
Nowak,OU=Users,DC=filia,DC=ad,DC=zielona,DC=example anowak          user
```

Zielona ma jedną domenę, więc w produkcji skrypt przydaje się głównie przy członkach z domen zaufanych; przykład pokazuje las laboratoryjny z domeną podrzędną `filia`.

RSAT = Remote Server Administration Tools: dodatkowe narzędzia administracyjne Windows do zarządzania AD, DNS i GPO ze stacji roboczej · **DN** = Distinguished Name: pełny, jednoznaczny adres jednego obiektu w katalogu LDAP, na przykład konta albo grupy · **DC** = Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników · **LDAP** = Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA) · **DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC

TYLKO ODCZYT

Audyt grupy z zagnieżdżoną grupą: wariant z zapisem do CSV

```
# Pytaj użytkownika o nazwę grupy, kontroler domeny (DC) i poświadczenia
$Group = Read-Host -Prompt "Podaj nazwę grupy"
$DC = Read-Host -Prompt "Podaj kontroler domeny bądź jej nazwę (NazwaKontroleraDC /
NazwaDomeny)"
$Credential = Get-Credential -Message "Podaj poświadczenia użytkownika do dostępu do
domeny $DC"

# Pobierz członków grupy z określonego kontrolera domeny
$membership = Get-ADGroup $Group -Properties Member -Server $DC -Credential $Credential

# Zbieramy obiekty członków grupy
$results = foreach ($domainGroup in $membership.Member | Group-Object { ($_ -split
'(?=DC=)', 2)[1] }) {
```

```
    [adsi]$ldap = "LDAP://$($domainGroup.Name)"
    [string]$domain = $ldap.Name

    foreach ($member in $domainGroup.Group) {
        $obj = Get-ADObject $member -Server $domain -Credential $Credential
        [pscustomobject]@{
            MemberOf       = $membership.Name
            Domain         = $domain
            Name           = $obj.Name
            DSN            = $obj.DistinguishedName
            SamAccountName = $obj.SamAccountName
            ObjectClass    = $obj.ObjectClass
        }
    }
}

# Pytanie do użytkownika czy zapisać wynik
$saveCsv = Read-Host -Prompt "Czy chcesz zapisać wynik do pliku CSV? (t/n)"

if ($saveCsv -eq 't') {
    $path = Read-Host -Prompt "Podaj pełną ścieżkę do pliku CSV"

    try {
        $results | Export-Csv -Path $path -NoTypeInformation -Encoding UTF8
        Write-Host "Wynik zapisano do pliku: $path" -ForegroundColor Green
    }
    catch {
        Write-Host "Wystąpił błąd podczas zapisywania pliku:" -ForegroundColor Red
        Write-Host $_.Exception.Message -ForegroundColor Red
    }
}
else {
    Write-Host "Wynik nie został zapisany do pliku." -ForegroundColor Yellow
}

# Wyświetlenie wyników na ekranie
$results
```

Ten sam mechanizm co wyżej, rozbudowany o opcjonalny zapis do CSV: wygodne, gdy audyt trzeba udokumentować albo porównać ze stanem sprzed miesiąca, zamiast tylko przejrzeć wynik na ekranie i zapomnieć.

CSV = w Windowsie: Cluster Shared Volume, dysk widoczny naraz dla wszystkich węzłów klastra; w danych: Comma-Separated Values, plik z wartościami oddzielonymi przecinkami ·**DC** = Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników ·**LDAP** = Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA) ·**DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC

TYLKO ODCZYT

Skrypt wykonujący audyt wszystkich wpisów w DNS

```
### Pobieranie argumentu adresu serwera DNS ###
if ($args.Count -eq 2) {
    $DnsServer = $args[0]
    $SavePath = $args[1]
} else {
    # Jeśli uruchomiono bez argumentów, prosimy o wpisanie ręcznie
    Write-Host "Uruchom skrypt poprzez ./NazwaSkryptu.ps1 AdresSerweraDNS
ŚcieżkaDoPliku.txt bądź podaj je teraz: "
    $DnsServer = Read-Host "Podaj adres serwera DNS"
    $SavePath = Read-Host "Podaj ścieżkę do zapisu"
}

# Sprawdzenie czy podano ścieżkę do zapisu
if (-not $SavePath) {
    Write-Host "Nie podano ścieżki do zapisu. Skrypt zostanie przerwany." -
ForegroundColor Red
    exit
}

# Pobranie stref DNS z serwera
$Zones = @(Get-DnsServerZone -ComputerName $DnsServer)

ForEach ($Zone in $Zones) {
    # Wypisywanie strefy do konsoli i pliku
    Write-Host "`n$($Zone.ZoneName)" -ForegroundColor "Green"
    "`n$($Zone.ZoneName)" | Out-File -Append -FilePath $SavePath
```

```
    # Pobranie rekordów DNS w strefie i zapis do pliku
    Get-DnsServerResourceRecord -ComputerName $DnsServer -ZoneName $Zone.ZoneName | Out-
File -Append -FilePath $SavePath
}

Write-Host "Zapis zakończony. Dane znajdują się w: $SavePath" -ForegroundColor Cyan
```

Skrypt uruchamiamy jako Administrator w PowerShell, z zainstalowanymi przystawkami RSAT DNS. Zapisuje pliki w aktualnie używanym folderze, jeśli podajemy ścieżkę względną: zamierzone, żeby ten sam skrypt dało się odpalić z różnych katalogów roboczych bez modyfikacji. Uruchomienie z argumentami (`./audyt-dns.ps1 dc01 raport.txt`) pomija oba pytania interaktywne, co czyni go gotowym do wywołania z harmonogramu zadań.

Przykładowe uruchomienie:

```
PS C:\Audyty> .\audyt-dns.ps1 dc01 dns-raport.txt

ad.zielona.example
dc01         Host (A)     10.20.10.11
app01        Host (A)     10.20.10.41

Zapis zakończony. Dane znajdują się w: dns-raport.txt
```

□□ **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **RSAT** = Remote Server Administration Tools: dodatkowe narzędzia administracyjne Windows do zarządzania AD, DNS i GPO ze stacji roboczej · **exit** = zakończ bieżącą powłokę albo skrypt z podanym kodem wyjścia

□□ **ZMIENIA SYSTEM**

Wariant DNS z interaktywnym oknem dla każdej strefy

```
Add-Type -AssemblyName Microsoft.VisualBasic
Add-Type -TypeDefinition @"
using System;
using System.Runtime.InteropServices;

public class WinAPI {
    [DllImport("user32.dll")]
    public static extern bool SetForegroundWindow(IntPtr hWnd);
```

```
}
"@ -Language CSharp

# Wyświetlenie okna dialogowego do wpisania adresu serwera DNS z instrukcją
$DnsServer = [Microsoft.VisualBasic.Interaction]::InputBox(
    "Podaj adres serwera DNS w formacie: NazwaSerwera.NazwaDomeny`n`nUWAGA: Skrypt musi
być uruchomiony na komputerze z zainstalowaną przystawką RSAT: DNS!`n`nUWAGA: Skrypt
otwiera nowe interaktywne okno dla każdej zony!",
    "Wprowadź adres DNS",
    "dc01.ad.zielona.example"
)

# Ustawienie okna na pierwszym planie po zamknięciu InputBox
$process = Get-Process | Where-Object { $_.MainWindowTitle -match "Wprowadź adres DNS" }
if ($process) {
    [WinAPI]::SetForegroundWindow($process.MainWindowHandle)
}

# Jeśli użytkownik nic nie wpisał i kliknął "Anuluj", skrypt się zatrzyma
if (-not $DnsServer) {
    Write-Host "Nie podano adresu serwera DNS. Skrypt zostanie zakończony." -
ForegroundColor Red
    exit
}

# Pobranie stref DNS z serwera
$Zones = @(Get-DnsServerZone -ComputerName $DnsServer)

ForEach ($Zone in $Zones) {
    Write-Host "`n$($Zone.ZoneName)" -ForegroundColor "Green"

    # Pobranie rekordów DNS w strefie
    $records = Get-DnsServerResourceRecord -ComputerName $DnsServer -ZoneName
$Zone.ZoneName

    if ($records) {
        # Wymuszenie oczekiwania na zamknięcie Out-GridView przed otwarciem kolejnego
        $records | Out-GridView -Title "Rekordy DNS - $($Zone.ZoneName)" -PassThru |
Out-Null
```

```
    } else {
        Write-Host "Brak rekordów w strefie $($Zone.ZoneName)" -ForegroundColor Yellow
    }
}

Write-Host "Koniec działania skryptu." -ForegroundColor Cyan
```

Ten sam audyt, ale zamiast pliku tekstowego każda strefa otwiera się jako osobne, interaktywne okno `Out-GridView`: okno kolejnej strefy otwiera się dopiero po zamknięciu poprzedniego, więc przy dużej liczbie stref przeklikanie się przez wszystkie może zająć sporo czasu. Przydatne do jednorazowego, ręcznego przeglądania małej liczby stref, niepraktyczne jako zadanie cykliczne bez nadzoru: do tego celu służy wariant zapisujący do pliku wyżej.

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **RSAT** = Remote Server Administration Tools: dodatkowe narzędzia administracyjne Windows do zarządzania AD, DNS i GPO ze stacji roboczej · **exit** = zakończ bieżącą powłokę albo skrypt z podanym kodem wyjścia

ZMIENIA SYSTEM

Skrypt dodający aliasy SMTP użytkownikom w parametrach AD

```
# Importowanie modułu Active Directory
Import-Module ActiveDirectory

# Definicja zmiennych dla domen
$oldDomain = "NazwaStarejDomeny"
$newDomain = "NazwaNowejDomeny"

# Określenie wyróżnionej nazwy (DN) docelowej jednostki organizacyjnej (OU)
$ou = "SciezkaDoOU"

# Pobranie wszystkich użytkowników (aktywnych i nieaktywnych) w określonym OU wraz z ich
UserPrincipalName i ProxyAddresses
$users = Get-ADUser -Filter * -SearchBase $ou -Properties UserPrincipalName,
ProxyAddresses, SamAccountName

# Iteracja przez każdego użytkownika (zarówno aktywnego, jak i nieaktywnego) i dodanie
aliasu
foreach ($user in $users) {
```

```
    # Upewnienie się, że użytkownik ma ustawiony UserPrincipalName kończący się na
starej domenie
    if ($user.UserPrincipalName -match "@$oldDomain$") {

        # Wyodrębnienie nazwy użytkownika poprzez usunięcie starej domeny z
UserPrincipalName
        $username = $user.UserPrincipalName -replace "@$oldDomain", ""

        # Utworzenie aliasu w formacie username@newDomain
        $alias = "smtp:$username@$newDomain"

        # Pobranie aktualnych ProxyAddresses (aliasów e-mail)
        $proxyAddresses = $user.ProxyAddresses

        # Upewnienie się, że ProxyAddresses jest zainicjalizowane
        if (-not $proxyAddresses) {
            $proxyAddresses = @()
        }

        # Sprawdzenie, czy alias już istnieje
        if ($proxyAddresses -notcontains $alias) {
            try {
                # Dodanie nowego aliasu
                Set-ADUser -Identity $user -Add @{ProxyAddresses = $alias}
                Write-Host "Dodano alias $alias dla użytkownika $username"
            } catch {
                Write-Host "Błąd podczas dodawania aliasu $alias dla użytkownika
$username: $_" -ForegroundColor Red
            }
        } else {
            Write-Host "Alias $alias już istnieje dla użytkownika $username"
        }
    } else {
        Write-Host "Pomijanie użytkownika $($user.SamAccountName), ponieważ nie ma
poprawnego UserPrincipalName kończącego się na @$oldDomain" -ForegroundColor Yellow
    }
}
```

Skrypt wykonujemy z poziomu PowerShell po zmianie zmiennych `$oldDomain`, `$newDomain` i `$ou` na wartościach właściwych dla naszego środowiska. Skrypt musi być uruchomiony na serwerze z zainstalowaną funkcją RSAT, z dostępem do domeny, w której chcemy go uruchomić. Typowy scenariusz użycia to migracja domeny pocztowej: organizacja zmienia domenę SMTP i wszyscy użytkownicy w danej jednostce organizacyjnej mają dostać nowy adres jako alias, bez usuwania starego z listy `ProxyAddresses`.

Przykład:

```
Dodano alias smtp:jkowalski@zielona.example dla użytkownika jkowalski
Alias smtp:anowak@zielona.example już istnieje dla użytkownika anowak
Pomijanie użytkownika switek, ponieważ nie ma poprawnego UserPrincipalName kończącego
się na @ad.zielona.example
```

SMTP = Simple Mail Transfer Protocol: protokół wysyłania poczty · **AD** = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie · **RSAT** = Remote Server Administration Tools: dodatkowe narzędzia administracyjne Windows do zarządzania AD, DNS i GPO ze stacji roboczej · **alias** = nadaj poleceniu własny, krótszy skrót na czas sesji powłoki

OSTROŻNIE · NIEODWRACALNE

Wyszukiwanie frazy we wszystkich GPO

```
# Definiujemy wzorzec wyszukiwania na samej górze
$searchPattern = "SzukanaFraza"

Write-Host "`nTworzę folder na raporty GPO w bieżącym katalogu..."

# Próba pobrania ścieżki skryptu, jeśli brak to używamy bieżącego katalogu
if ($MyInvocation.MyCommand.Path) {
    $scriptFolder = Split-Path -Parent $MyInvocation.MyCommand.Path
} else {
    Write-Host "Nie wykryto ścieżki skryptu, używam bieżącego katalogu."
    $scriptFolder = (Get-Location).Path
}

$folder = Join-Path $scriptFolder "GPOReports"
New-Item -Path $folder -ItemType Directory -Force | Out-Null

Write-Host "Pobieram wszystkie obiekty GPO z domeny..."
$allgpos = Get-GPO -All
```

```
# Plik do zapisania raportu z listą i wynikiem w folderze skryptu
$reportFile = Join-Path $scriptFolder "GPO_export_and_scan_report.txt"

# Czyszczenie pliku jeśli istnieje
if (Test-Path $reportFile) {
    Remove-Item $reportFile
}

Write-Host "Eksportuję wszystkie GPO do osobnych plików XML oraz zapisuję listę do pliku raportu..."
Add-Content -Path $reportFile "Lista wyeksportowanych GPO:`n"
$allgpos | ForEach-Object {
    $path = Join-Path $folder ("{0}_{1}.xml" -f $_.DisplayName, $_.Id)
    Write-Host ("Eksportuję GPO: {0}" -f $_.DisplayName)
    Get-GPOReport -Guid $_.Id -ReportType Xml -Path $path
    Add-Content -Path $reportFile $_.DisplayName
}

# Dodaj pustą linię do rozdzielenia sekcji
Add-Content -Path $reportFile "`nWyniki wyszukiwania wzorca '$searchPattern':`n"

Write-Host "Szukam wzorca '$searchPattern' we wszystkich wyeksportowanych raportach..."
$patternMatches = Select-String -Path (Join-Path $folder "*.xml") -Pattern $searchPattern
if ($patternMatches -and $patternMatches.Count -gt 0) {
    $patternMatches | ForEach-Object { Add-Content -Path $reportFile $_.ToString() }
} else {
    Add-Content -Path $reportFile "Brak wzorca '$searchPattern' w żadnym z raportów GPO."
}

Write-Host "`nGotowe! Pełny raport w pliku: $reportFile"
```

Skrypt tworzy katalog `GPOReports` w folderze, z którego jest uruchamiany, eksportuje do niego pełny raport XML każdego GPO w domenie, po czym przeszukuje wszystkie wyeksportowane pliki w poszukiwaniu ustawionej na samej górze frazy. Uruchamiamy w PowerShell z uprawnieniami administratora, na kontrolerze domeny albo na maszynie z modułem `GroupPolicy` i dostępem do domeny. Typowe zastosowanie: sprawdzenie, w którym GPO (spośród często kilkudziesięciu) skonfigurowano konkretne ustawienie rejestru, nazwę

usługi albo ścieżkę skryptu logowania, bez ręcznego otwierania każdego GPO w konsoli zarządzania.

Przykład:

```
Pobieram wszystkie obiekty GPO z domeny...
Eksportuję GPO: Polityka haseł
Eksportuję GPO: Mapowanie dysków sieciowych
Eksportuję GPO: Konfiguracja proxy

Szukam wzorca 'SzukanaFraza' we wszystkich wyeksportowanych raportach...

Gotowe! Pełny raport w pliku: GPO_export_and_scan_report.txt
```

GPO = Group Policy Object: zestaw ustawień wypychanych z Active Directory na komputery · **XML** = eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml

TYLKO ODCZYT

Pobieranie klucza BitLocker z AD

```
$objComputer = Get-ADComputer NazwaKomputera
$Bitlocker_Objects = Get-ADObject -Filter {objectclass -eq 'msFVE-RecoveryInformation'}
-SearchBase $objComputer.DistinguishedName -Properties 'msFVE-RecoveryPassword'
$Bitlocker_Objects
```

Uruchamiamy jako user z uprawnieniami do odczytu atrybutu `msFVE-RecoveryPassword`: w wielu środowiskach domyślnie ograniczonego do administratorów domeny albo do wydzielonej grupy service desk. **Pojęcie: msFVE-RecoveryInformation.** Obiekt potomny konta komputera w Active Directory, tworzony automatycznie przy włączeniu BitLockera z kopią zapasową klucza do AD: jeden komputer może mieć kilka takich obiektów, po jednym na każde szyfrowanie/odszyfrowanie wolumenu w historii.

Przykład:

```
DistinguishedName : CN={a1b2c3d4-
...},CN=NazwaKomputera,OU=Komputery,DC=ad,DC=zielona,DC=example
msFVE-RecoveryPassword : 123456-654321-123456-654321-123456-654321-123456-654321
```

AD = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie

TYLKO ODCZYT

Pobieranie klucza BitLocker: wariant z pytaniem o komputer i poświadczenia

```
# Pobieranie nazwy komputera
$ComputerName = Read-Host "Nazwa komputera [$env:COMPUTERNAME]"
if ([string]::IsNullOrWhiteSpace($ComputerName)) {
    $ComputerName = $env:COMPUTERNAME
}

# Pobieranie poświadczeń domenowych
$Credentials = Get-Credential -Message "Poświadczenia do domeny AD"

# Wyszukanie odpowiedniego komputera
$objComputer = Get-ADComputer $ComputerName -Credential $Credentials

# Pobranie kluczy BitLocker z dodatkowymi polami
$Bitlocker_Object = Get-ADObject `
    -Filter {objectclass -eq 'msFVE-RecoveryInformation'} `
    -SearchBase $objComputer.DistinguishedName `
    -Properties 'msFVE-RecoveryPassword', 'msFVE-RecoveryGuid', 'msFVE-VolumeGuid',
'whenCreated' `
    -Credential $Credentials |
    Sort-Object whenCreated -Descending |
    Select-Object `
        @{Name = 'CN'; Expression = { $ComputerName }},
        @{Name = 'Bitlocker Password'; Expression = { $_.'msFVE-RecoveryPassword' }},
        @{Name = 'Volume GUID'; Expression = { $_.'msFVE-VolumeGuid' }},
        @{Name = 'Recovery GUID'; Expression = { $_.'msFVE-RecoveryGuid' }},
        @{Name = 'Created'; Expression = { $_.whenCreated }}

# Wypisanie danych
$Bitlocker_Object | Format-Table -AutoSize
```

Rozszerzamy poprzedni wariant o pytanie interaktywne o nazwę komputera (z domyślną wartością bieżącego hosta) i o osobne poświadczenia domenowe, oraz sortujemy wynik po dacie utworzenia malejąco: jeśli komputer miał kilka cykli szyfrowania, najnowszy, aktualnie obowiązujący klucz znajdzie się na górze listy.

Przykład:

```
CN            Bitlocker Password                                          Volume
GUID                          Recovery GUID                          Created
--            ------------------                                          -----------
                          -------------                          -------
NazwaKomputera 123456-654321-123456-654321-123456-654321-123456-654321    {b2c3d4e5-
...}                       {c3d4e5f6-...}                        24.09.2026 09:12:41
```

AD = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie · **CN** = Common Name: główna nazwa w certyfikacie, np. adres serwera · **GUID** = Globally Unique Identifier: unikalny w skali globalnej identyfikator obiektu, tu konkretnego protektora klucza BitLockera

ZMIENIA SYSTEM

Manualne wymuszenie synchronizacji AD on-premises z Entra ID

```
Start-ADSyncSyncCycle -PolicyType Delta
```

Komendę uruchamiamy jako Administrator w PowerShell na serwerze z zainstalowanym i skonfigurowanym Azure AD Connect (obecnie Microsoft Entra Connect, w Zielonej `sync01`), ze skonfigurowanymi odpowiednimi połączeniami do Entra ID. Cykl `Delta` synchronizuje wyłącznie zmiany od ostatniej synchronizacji: dużo szybszy niż pełny cykl i wystarczający w zdecydowanej większości przypadków, gdy trzeba natychmiast przepchnąć pojedynczą zmianę (na przykład świeżo dodany alias SMTP ze skryptu wyżej) bez czekania na najbliższy zaplanowany cykl synchronizacji.

Przykład:

```
Result
------
Success
```

Pełny cykl (`-PolicyType Initial`) przelicza cały zestaw obiektów od nowa: używamy go rzadko, głównie po zmianie reguł synchronizacji, nie jako rutynowe polecenie.

Uważaj

- `Get-ADObject -Filter {objectclass -eq 'msFVE-RecoveryInformation'}` zwraca puste wyniki, jeśli konto komputera nigdy nie zapisało klucza do AD: dotyczy to maszyn z BitLockerem skonfigurowanym bez zasady grupy wymuszającej kopię zapasową klucza do katalogu.
- Skrypt dodający aliasy SMTP modyfikuje atrybuty kont w produkcyjnym katalogu: pierwsze uruchomienie warto wykonać na pojedynczym koncie testowym albo z dopisanym na sztywno filtrem ograniczającym `$users` do jednego obiektu, zanim puścimy go na całe OU.
- `Start-ADSyncSyncCycle -PolicyType Initial` na dużym katalogu potrafi trwać wyraźnie dłużej niż `Delta` i obciążać kontroler domeny odczytem: nie uruchamiamy go odruchowo zamiast `Delta`.

AD = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie · **SMTP** = Simple Mail Transfer Protocol: protokół wysyłania poczty · **OU** = Organizational Unit: folder w Active Directory grupujący konta i komputery

Z życia wzięte

Z ŻYCIA WZIĘTE

Pan Wojtek pyta w listopadzie, przed corocznym audytem, na których stacjach roboczych BitLocker w ogóle zapisał klucz odzyskiwania do katalogu. Odpalamy skrypt pobierający klucz z AD dla kilkunastu komputerów po kolei i dla trzech z nich `Get-ADObject` zwraca pustą listę. Okazuje się, że te trzy laptopy dostały szyfrowanie jeszcze przed włączeniem zasady grupy wymuszającej kopię zapasową klucza, więc klucz istnieje tylko lokalnie, nigdy nie trafił do Active Directory.

Zanim zaczniemy

Wszystkie skrypty na tej stronie wymagają modułu `ActiveDirectory`, dostarczanego przez funkcję RSAT (Remote Server Administration Tools): na stacji roboczej instalujemy go poleceniem `Add-WindowsFeature RSAT-AD-PowerShell` (na serwerze) albo przez `Add-WindowsCapability -Online -Name Rsat.ActiveDirectory...` (na kliencie Windows 10/11), opisane szerzej na stronie Instalacja i usuwanie oprogramowania. Skrypt audytujący DNS wymaga dodatkowo przystawki RSAT DNS, a skrypt przeszukujący GPO wymaga modułu `GroupPolicy` z RSAT GPMC. Wszystkie skrypty uruchamiamy w PowerShell z uprawnieniami odpowiednimi do zakresu odczytu, jakiego wymaga dana operacja: audyt grup i DNS wystarczy zwykłe konto z prawem odczytu katalogu, pobranie klucza BitLocker wymaga uprawnień do odczytu atrybutu `msFVE-RecoveryPassword`, a synchronizacja

z Entra ID wymaga uprawnień administratora na serwerze z rolą Azure AD Connect.

Po co to istnieje

Audyt Active Directory ma jedną wspólną cechę we wszystkich sześciu skryptach powyżej: dane, których szukamy, technicznie istnieją w katalogu, ale żadna standardowa konsola graficzna (Active Directory Users and Computers, konsola zarządzania GPO) nie pokazuje ich w formie gotowej do przeszukania masowo: trzeba by klikać obiekt po obiekcie, GPO po GPO, strefę po strefie. PowerShell z modułem `ActiveDirectory` zamienia to z zadania ręcznego, powtarzalnego i podatnego na pominięcie czegoś, w jednorazowy skrypt, który przechodzi przez wszystko naraz i zwraca gotowy raport.

Jak to działa

Skrypt audytu grupy zagnieżdżonej rozwiązuje konkretny problem techniczny `Get-ADGroupMember`: to polecenie domyślnie odpytuje tylko bieżącą domenę i nie podąża automatycznie za członkami z innych domen tego samego lasu: zwraca dla nich błąd albo pomija ich całkowicie, w zależności od wersji. Rozbicie `DistinguishedName` członka po wzorcu `DC=` i zbudowanie z tego osobnej ścieżki LDAP pozwala ustalić, do jakiej domeny naprawdę należy dany obiekt, i odpytać właściwy kontroler domeny zamiast zakładać, że wszystko leży w tej samej domenie co grupa nadrzędna.

Wyszukiwanie frazy w GPO nie odpytuje samych zasad grupy bezpośrednio: `Get-GPOReport -ReportType Xml` generuje pełny, płaski zrzut XML każdego GPO (ustawienia, filtry zabezpieczeń, powiązania), a dopiero `Select-String` przeszukuje tekstowo te wyeksportowane pliki. To dwuetapowe podejście istnieje, bo GPO nie ma wbudowanego mechanizmu pełnotekstowego wyszukiwania po swojej zawartości: jedynym pewnym sposobem sprawdzenia, czy dana fraza (nazwa usługi, ścieżka rejestru, adres serwera) występuje gdziekolwiek w polityce, jest zrzucenie wszystkiego do formatu tekstowego i przeszukanie go standardowymi narzędziami.



Kazde GPO trafia do osobnego pliku XML, dopiero potem Select-String szuka w nich frazy (kliknięcie otwiera pełny rozmiar)

Częste błędy

Brak przystawki RSAT DNS przy próbie audytu DNS. `Get-DnsServerZone` zwraca błąd `The term 'Get-DnsServerZone' is not recognized`: moduł `DnsServer` (część RSAT DNS) nie instaluje się razem z modułem `ActiveDirectory`, trzeba go dodać osobno.

Uruchomienie skryptu SMTP na całym katalogu bez wcześniejszego testu na jednym koncie. Skrypt zmienia atrybuty realnych kont produkcyjnych: literówka w `$newDomain` albo błędnie zdefiniowany `$ou` obejmujący więcej użytkowników niż zamierzono, oznacza masową, trudną do szybkiego cofnięcia zmianę na wielu kontach naraz.

Oczekiwanie klucza BitLocker dla komputera, który nigdy nie zapisał go do AD. Pusty wynik `Get-ADObject` nie zawsze oznacza błąd skryptu: równie często oznacza, że zasada grupy wymuszająca kopię zapasową klucza do katalogu nie była włączona w momencie szyfrowania tego konkretnego dysku.

□□ Ćwiczenia

> Ćwiczenia poniżej wymagają dostępu do testowej domeny Active Directory z zainstalowanym modułem RSAT: nie wykonujemy ich na domenie produkcyjnej.

Ćwiczenie 1: rozbicie DistinguishedName po domenie

```
$przykladowyDN = "CN=Jan Kowalski,OU=Users,DC=filia,DC=ad,DC=zielona,DC=example"
$przykladowyDN -split '(?=DC=)', 2
```

Sprawdzamy, jak dokładnie wyrażenie regularne z fiszki audytu grup rozdziela DistinguishedName na część obiektu i część domeny.

Odpowiedź

Wynikiem są dwa elementy: `CN=Jan Kowalski,OU=Users,` oraz `DC=filia,DC=ad,DC=zielona,DC=example`. Pozytywne wyprzedzenie `(?=DC=)` dzieli napis dokładnie przed pierwszym wystąpieniem `DC=`, nie konsumując samego dopasowanego fragmentu: dzięki temu drugi element zawiera kompletną, gotową do użycia w `[adsi]` ścieżkę domeny, bez potrzeby ręcznego doklejania z powrotem obciętego prefiksu.

Ćwiczenie 2: bezpieczne przetestowanie skryptu SMTP

```
$testowyUser = Get-ADUser -Filter * -SearchBase "SciezkaDoOU" -Properties UserPrincipalName |
Select-Object -First 1
$testowyUser.UserPrincipalName
```

Zanim uruchomimy pełny skrypt dodawania aliasów SMTP na całym OU, wyciągamy z niego pojedynczego, przykładowego użytkownika i sprawdzamy ręcznie, czy filtr `-SearchBase` faktycznie obejmuje tylko zamierzony zakres.

Odpowiedź

Sprawdzenie pojedynczego wyniku przed uruchomieniem pętli na całym zbiorze to praktyczna metoda uniknięcia masowej, niezamierzonej zmiany: jeśli `$testowyUser` pochodzi z niewłaściwej jednostki organizacyjnej albo ma nieoczekiwany `UserPrincipalName`, oznacza to błąd w wartości `$ou` jeszcze przed uruchomieniem właściwego skryptu na wszystkich kontach.

Pytania kontrolne

1. Dlaczego skrypt audytu GPO eksportuje każdą politykę do osobnego pliku XML zamiast przeszukiwać GPO bezpośrednio przez moduł GroupPolicy?

Odpowiedź

Moduł `GroupPolicy` nie udostępnia polecenia do pełnotekstowego przeszukania zawartości ustawień GPO: `Get-GPOReport` generuje kompletny, płaski zrzut ustawień w formacie XML (albo HTML), który dopiero wtedy da się przeszukać ogólnym narzędziem tekstowym, takim jak `Select-String`. To pośredni krok wymuszony brakiem bezpośredniego API do wyszukiwania w treści polityk.

2. Skrypt pobierający klucz BitLocker zwraca pusty wynik dla komputera, o którym wiadomo, że ma włączony BitLocker. Jaka jest najbardziej prawdopodobna przyczyna, niezwiązana z samym skryptem?

Odpowiedź

BitLocker mógł zostać włączony bez skonfigurowanej wcześniej zasady grupy wymuszającej zapis klucza odzyskiwania do Active Directory: w takim przypadku klucz nigdy nie trafił do obiektu `msFVE-RecoveryInformation`, bo taki obiekt w ogóle nie powstał. Sam skrypt działa poprawnie, po prostu przeszukuje katalog, w którym nie ma czego znaleźć.

Powiązane: Zadania zaplanowane, usługi i procesy · Poświadczenia i bezpieczne hasła w skryptach · Instalacja i usuwanie oprogramowania · Konta usługowe MSA i gMSA · Hartowanie Active Directory: model warstw, konta uprzywilejowane, protokoły i audyt · Entra Connect i Cloud Sync: synchronizacja ad.zielona.example z Entra ID

Źródła: `Get-Help Get-ADObject` · `Get-Help Get-GPOReport` · dokumentacja Microsoft Learn, „Active Directory module for Windows PowerShell" i „Microsoft Entra Connect Sync"

Revision #43
Created 2025-03-10 13:51:07 UTC by Tomasz Konieczny
Updated 2026-10-26 14:26:34 UTC by Tomasz Konieczny