Delegowanie i ACL – instrukcja krok po kroku w PowerShell ISE (wersja ISE). 14.1c2

Cel ogólny lekcji: Nauczyć poprawnego delegowania uprawnień i konfigurowania zabezpieczeń (ACL) dla obiektów Active Directory w domenie rol00.edu.pl.

Cele szczegółowe:

1. Przygotować środowisko laboratoryjne (uruchomienie DC, weryfikacja sieci/DNS, załadowanie modułu AD; uruchomienie skryptu przygotowawczego).
2. Delegować kontrolę dla użytkowników puz1 i puz2 na jednoznacznie wskazanym obiekcie (domena rol00.edu.pl lub OU=pAbc) – spójnie z ilustracjami.
3. Skonfigurować zabezpieczenia (ACL) dla OU=pAbc: nadać użytkownikowi puz2 uprawnienia Odczyt, Zapis, Tworzenie wszystkich obiektów podrzędnych, Usuwanie wszystkich obiektów podrzędnych.
4. Wyjaśnić działanie Advanced Security i dziedziczenia (Inheritance) w ACL.
5. Zweryfikować wykonane czynności w ADUC (Security > Advanced) oraz w PowerShell (odczyt wpisów ACL) i udokumentować wyniki w zeszycie.

A. Przygotowanie środowiska. Przed przystąpieniem do ćwiczenia sprawdź i ustaw:

1. Przygotowanie maszyn w Hyper-V

a. Sprawdzenie punktu kontrolnego z DC

W Menedżer funkcji Hyper-V wybierz nazwa maszynę wirtualna twojej grupy_dc2019

Upewnij się, że punkt kontrolny, zawiera serwer z zainstalowanym kontrolerem domeny.

2. Start systemu i weryfikacja

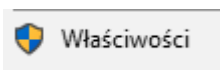
a. Uruchomienie serwera: Uruchom maszynę serwera.

b. Logowanie Administrator: Ctrl+Delete > Administrator > zaq1@WSX

c. Sprawdzenie konfiguracji sieci:



> 192.167.0.1 > 

> 

>

DHCP włączone	Nie
Adres IPv4	192.167.0.1
Maska podsieci IPv4	255.255.255.0
Brama domyślna IPv4	
Serwer DNS IPv4	192.167.0.1

3. Konfiguracja serwera - część automatyczna (PowerShell ISE):

a. Na kontrolerze domeny (dc2019) uruchom Windows PowerShell ISE jako administrator.

b. Otwórz plik cw14.1c2.txt.

c. Skopiuj całą jego zawartość do okna Script Pane (górne okno w ISE).

d. Uruchom skrypt przyciskiem Run Script (F5).

e. Poczekaj na komunikaty potwierdzające: utworzenie kont puz1 i puz2, utworzenie OU pMan, utworzenie OU pAbc pod domeną.

Skrypt **nie wykonuje delegacji ani ACL** – przygotowuje tylko środowisko wymagane w części właściwej.

W zeszycie opisz procedury konfiguracji.

W zeszycie wykonaj pełny opis procedury konfiguracji. Każde polecenie lub każdy krok, który wykonujesz na serwerze lub kliencie, musi być przepisany do zeszytu i opatrzony krótkim opisem, co robi. Przykładowy wpis, który uczeń powinien umieć zrobić:

W PowerShell ISE uruchomiłem komendę pobierającą obiekt domeny za pomocą DirectoryEntry:

New-Object System.DirectoryServices.DirectoryEntry("LDAP://DC=rol00,DC=edu,DC=pl")

Ta czynność pozwala uzyskać bezpośredni dostęp do listy zabezpieczeń (ACL) obiektu domeny i przygotowuje środowisko do ręcznej modyfikacji uprawnień użytkowników.

Instrukcja prowadzi ucznia przez ręczne wykonanie zadań z pliku cw14.1c2odp.txt w środowisku PowerShell ISE. Uczeń wkleja i uruchamia małe porcje poleceń, obserwuje efekt i weryfikuje wynik. Plik cw14.1c2odp.txt NIE jest uruchamiany w całości.

B. Przygotowanie PowerShell ISE (na kontrolerze domeny)

Uruchom PowerShell ISE jako Administrator. W dolnym oknie (Console Pane) wykonaj:

```
Import-Module ActiveDirectory
```

Ładuje moduł RSAT ActiveDirectory, który udostępnia cmdlety takie jak Get-ADUser, Get-ADDomain, New-ADOrganizationalUnit itp.

Wymagania/uwagi:

Musisz uruchamiać na kontrolerze domeny lub na hoście z zainstalowanym RSAT i połączeniem do DC.

Jeśli moduł nie ładuje się, sprawdź rolę/RSAT i uprawnienia konta.

```
(Get-ADDomain).DNSRoot

(Get-ADRootDSE).DefaultNamingContext
```

Get-ADDomain daje obiekt bieżącej domeny (m.in. DNSRoot, NetBIOSName).

Get-ADRootDSE ujawnia metadane katalogu; DefaultNamingContext to DN bazy katalogowej domeny (np. DC=rol00,DC=edu,DC=pl).

Zastosowanie: Te dwa wywołania potwierdzają, że sesja widzi poprawny kontekst domeny i umożliwiają budowanie ścieżek LDAP (DistinguishedName) w dalszych krokach.

C. Zmienne sterujące scenariuszem

```
$DelegateOnDomain = $true  # true => delegacja na DOMENIE; false => delegacja na OU=pAbc

$UsersForDelegation = @('puz1','puz2')

$AclUserForpAbc  = 'puz2'

$ApplyAclForpAbc = $true


$DomainDN  = (Get-ADRootDSE).DefaultNamingContext  # np. DC=rol00,DC=edu,DC=pl

$pAbcOUPath = "OU=pAbc,$DomainDN"

$DelegationTarget = if ($DelegateOnDomain) { $DomainDN } else { $pAbcOUPath }
```

Co i dlaczego:

- $DelegateOnDomain wybiera **obiekt delegacji**: domena (cały naming context) lub konkretna OU.
- $UsersForDelegation to lista kont, którym nadajesz **pakiet praw delegacji**.
- $AclUserForpAbc to konto, które dostanie **ACL** na OU=pAbc.
- $ApplyAclForpAbc pozwala warunkowo pominąć część ACL (dobry na demo i regres).
- $DomainDN, $pAbcOUPath, $DelegationTarget to **łańcuchy DN** używane do tworzenia ścieżek **LDAP://…** dla późniejszej manipulacji ACL przez **DirectoryEntry**.

Wyjaśnienie: Zmienne odzwierciedlają parametry z pliku odpowiedzi. Sprawdzamy istnienie OU=pAbc oraz kont puz1/puz2 (widoczność SID).

D. Walidacja obiektów w AD

```
Get-ADOrganizationalUnit -LDAPFilter '(ou=pAbc)' -SearchBase $DomainDN | Select-Object DistinguishedName
foreach ($sam in ($UsersForDelegation + $AclUserForpAbc | Select-Object -Unique)) {
    Get-ADUser -Identity $sam | Select-Object SamAccountName, SID
}
```

- Get-ADUser -Identity akceptuje SAMAccountName; zwracamy minimum, czyli SamAccountName, SID (SID przyda się do ACL).

- Select-Object -Unique eliminuje duplikaty (gdyby $AclUserForpAbc był też w $UsersForDelegation).

Uwagi: Jeśli konto nie istnieje, cmdlet zgłosi błąd - należy wcześniej wykonać skrypt przygotowawczy, który tworzy konta (cw14.1c2.txt).

E. ETAP B - Delegowanie uprawnień dla puz1 i puz2 (na domenie)

B1. Pobierz obiekt delegacji (domena lub OU=pAbc) i jego listę ACL:

```
$deDeleg = New-Object System.DirectoryServices.DirectoryEntry("LDAP://$DelegationTarget")
```

Tworzy .NET-owy obiekt DirectoryEntry do bezpośredniej pracy z wskazanym obiektem w AD (domena lub OU).

Dlaczego DirectoryEntry: daje dostęp do właściwości ObjectSecurity i metod Add/Set/CommitChanges, czyli do niskopoziomowej listy ACL, tak jak robi to kreator Delegate Control pod spodem.

```
$sec = $deDeleg.ObjectSecurity
```

Pobiera ActiveDirectorySecurity, czyli kolekcję reguł ACL przypiętą do obiektu.

Na tej kolekcji wykonujesz operacje: AddAccessRule, RemoveAccessRule, SetAccessRule.

Pamiętaj: kolekcja w pamięci jest tylko lokalną kopią - dopiero CommitChanges() na DirectoryEntry zapisuje do AD.

B2. Zdefiniuj pakiet praw odpowiadający kreatorowi Delegate Control i dodaj wpisy dla puz1/puz2:

ActiveDirectoryRights + AccessControlType + ActiveDirectorySecurityInheritance

W instrukcji tworzysz jeden zestaw praw:

```
$rights = [System.DirectoryServices.ActiveDirectoryRights]::ReadProperty `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::WriteProperty `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::CreateChild `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::DeleteChild `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::ListChildren `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::GenericRead `
      -bor [System.DirectoryServices.ActiveDirectoryRights]::GenericWrite
```

Co to znaczy:

- ReadProperty / WriteProperty > odczyt/zapis atrybutów obiektu.
- CreateChild / DeleteChild > tworzenie/usuwanie obiektów podrzędnych (np. użytkowników, grup, OU).
- ListChildren > możliwość wyświetlenia dzieci (elementy wewnątrz kontenera).

- GenericRead / GenericWrite > szersze zbiory uprawnień, które obejmują m.in. odczyt atrybutów i modyfikację właściwości.

Symbol -bor: to bitowe OR łączące flagi w pojedynczy zestaw.

Uwaga dydaktyczna: Ten pakiet odtwarza typowe wybory w kreatorze Delegate Control.

F) Mapowanie użytkownika na SID i tworzenie reguły ACL

Pobranie SID:

```
foreach ($sam in $UsersForDelegation) {
    $sid  = (Get-ADUser -Identity $sam -Properties SID).SID
    $sidO = New-Object System.Security.Principal.SecurityIdentifier($sid)
```

ACL operuje na SID, nie na nazwach — dzięki temu wpis jest jednoznaczny (nawet gdy zmienisz nazwę konta).

SecurityIdentifier to .NET-owa reprezentacja SID z wygodnymi metodami i konwersjami.

Tworzenie reguły:

```
    $rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
        $sidO,
        $rights,
        [System.Security.AccessControl.AccessControlType]::Allow,
        [System.DirectoryServices.ActiveDirectorySecurityInheritance]::All
    )
    $sec.AddAccessRule($rule)
}
```

ActiveDirectoryAccessRule reprezentuje pojedynczy wpis ACL (kto, jakie prawa, Allow/Deny, jak się dziedziczy).

AccessControlType::Allow > przyznajemy uprawnienia (alternatywnie byłoby Deny).

Inheritance::All > wpis może być dziedziczony w dół (np. na obiekty w OU), zależnie od tego, jak ustawione jest blokowanie dziedziczenia w danym obiekcie.

AddAccessRule dodaje wpis do kolekcji — nie zapisuje go jeszcze do AD.

B3. Zapisz zmiany w AD:

Zapis:

```
$deDeleg.ObjectSecurity = $sec
$deDeleg.CommitChanges()
```

Podmieniasz kolekcję ACL w DirectoryEntry i commitujesz.

Ważne: jeśli nie zrobisz CommitChanges(), nic nie zostanie utrwalone w AD.

B4. Zweryfikuj, że wpisy ACL dla puz1/puz2 istnieją:

```
($deDeleg.ObjectSecurity.Access |
    Where-Object { $_.IdentityReference -match 'puz1|puz2' } |
    Select-Object IdentityReference, ActiveDirectoryRights, AccessControlType, InheritanceType)
```

- ObjectSecurity.Access zwraca wszystkie wpisy ACL (także odziedziczone).
- Where-Object -match 'puz1|puz2' filtruje wpisy dla tych użytkowników.
- Select-Object wypisuje to, co najważniejsze: kto (Identity), które rights, czy Allow/Deny, oraz InheritanceType.

Pułapki:

Wpisy mogą pochodzić z dziedziczenia, więc nie zawsze zobaczysz dokładnie to, co dodałeś - uczniowie powinni rozróżniać Effective permissions od explicit.

F. ETAP C — ACL dla puz2 na OU=pAbc (Read, Write, Create Child, Delete Child)

C1. Pobierz OU=pAbc i aktualną listę ACL:

```
$deAbc  = New-Object System.DirectoryServices.DirectoryEntry("LDAP://$pAbcOUPath")
$secAbc = $deAbc.ObjectSecurity
```

C2. Przygotuj cztery prawa i dodaj regułę dla puz2, następnie zapisz:

```
$rightsAbc = [System.DirectoryServices.ActiveDirectoryRights]::ReadProperty `
        -bor [System.DirectoryServices.ActiveDirectoryRights]::WriteProperty `
        -bor [System.DirectoryServices.ActiveDirectoryRights]::CreateChild `
        -bor [System.DirectoryServices.ActiveDirectoryRights]::DeleteChild
$sid2    = (Get-ADUser -Identity $AclUserForpAbc -Properties SID).SID
$sid2Obj = New-Object System.Security.Principal.SecurityIdentifier($sid2)
$ruleAbc = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
   $sid2Obj, $rightsAbc,
   [System.Security.AccessControl.AccessControlType]::Allow,
   [System.DirectoryServices.ActiveDirectorySecurityInheritance]::All
)
$secAbc.AddAccessRule($ruleAbc)
$deAbc.ObjectSecurity = $secAbc
$deAbc.CommitChanges()
```

Kluczowe różnice względem delegacji na domenie:

- Tu **nie ma** GenericRead/Write ani ListChildren - tylko **cztery** prawa, które precyzyjnie definiują możliwości w obrębie OU.
- To daje uczniowi jasny obraz: **jak minimalny zestaw praw wpływa na operacje** (tworzenie/usuwanie i edycję atrybutów).

C3. Weryfikacja wpisu dla puz2 na OU=pAbc:

```
($deAbc.ObjectSecurity.Access |
    Where-Object { $_.IdentityReference -match $AclUserForpAbc } |
    Select-Object IdentityReference, ActiveDirectoryRights, AccessControlType, InheritanceType)
```

G. Test działania (opcjonalny – na poświadczeniach puz2)

Tworzenie przykładowego konta w OU=pAbc przy użyciu poświadczeń puz2 (sprawdza CreateChild i WriteProperty):

```
$cred = Get-Credential -Message 'Podaj dane użytkownika puz2'
New-ADUser -Name 'TestUserFromPuz2' -Path $pAbcOUPath -SamAccountName 'testPuz2' -Enabled $true -Credential $cred
```

Get-Credential

Wyświetla bezpieczne okno wprowadzania poświadczeń (tu: puz2).

Dobre praktyki:

Nie zapisuj haseł w skryptach; używaj SecureString i Credential object.

New-ADUser … -Credential $cred

Tworzy nowego użytkownika w OU=pAbc na uprawnieniach puz2, co wprost testuje CreateChild i częściowo WriteProperty.

Pułapki:

W zależności od polityk domeny mogą być wymagane dodatkowe atrybuty: hasło, UPN, itd.

Jeśli operacja się nie powiedzie, sprawdź czy puz2 ma dokładnie te prawa na OU oraz czy dziedziczenie nie jest blokowane.

Usunięcie testowego konta (sprawdza DeleteChild):

```
Remove-ADUser -Identity 'testPuz2' -Credential $cred -Confirm:$false
```

Usuwa testowy obiekt na uprawnieniach puz2, co testuje DeleteChild.

Dobre praktyki:

- -Confirm:$false wyłącza interaktywne potwierdzanie; na lekcji można zostawić -Confirm:$true, żeby wymusić świadomy klik.
- Po teście warto posprzątać, aby nie zostawiać „śmieci” w AD.

J) Najczęstsze problemy i jak je diagnozować

1. **Brak modułu ActiveDirectory** > nie działa Import-Module.
   - Zainstaluj RSAT / rola AD DS, sprawdź, czy sesja ma dostęp do DC.
2. **OU=pAbc nie istnieje** > Get-ADOrganizationalUnit nic nie zwraca.
   - Utwórz OU zgodnie z „przygotowaniem” (cw14.1c2.txt), sprawdź lokalizację OU.
3. **SID użytkownika pusty lub błąd** > konto nie istnieje albo brak replikacji.
   - Zweryfikuj Get-ADUser -Identity puz2 -Properties SID.

4. **Brak efektu po dodaniu reguły ACL** > zapomniane CommitChanges().
   - Zawsze ustaw ObjectSecurity i wołaj CommitChanges().
5. **Operacje New/Remove-ADUser się nie udają na poświadczeniach puz2**
   - Sprawdź, czy wpis ACL jest **Allow** (nie Deny) i czy **Inheritance** = All.
   - Sprawdź, czy OU nie ma włączonego „protect from accidental deletion" w sposób, który blokuje usunięcia.

K) Dobre praktyki (dla administracji i dla ucznia)

- **Używaj precyzyjnych zestawów praw** (tak jak w części C: tylko 4 prawa), zamiast dawać zbiorcze „GenericWrite". Zmniejsza to ryzyko **nadmiernych uprawnień**.
- **Zawsze weryfikuj** wpisy ACL w PS i w GUI (Security > Advanced), żeby porównać **teorię** z **praktyką**.
- **Komentuj kod** i **blokuj** przypadkowe usunięcia obiektów produkcyjnych; w labie możesz to wyłączyć dla celów ćwiczeń.
- **Testuj na poświadczeniach** delegowanych kont (-Credential). To najpewniejszy sposób sprawdzenia skuteczności wpisów ACL.

H. Notatki do zeszytu (skrót)

• Delegacja na domenie dla puz1/puz2 to wpisy ACL (Read/Write, Create/Delete Child, Generic Read/Write, ListChildren).

• ACL dla puz2 na OU=pAbc: ReadProperty, WriteProperty, CreateChild, DeleteChild.

• Wpisy ACL sprawdzisz w PS: ObjectSecurity.Access lub w GUI: Security > Advanced.

• Dziedziczenie ustawione na All – prawa dotyczą także obiektów podrzędnych.

• Po zmianach zawsze wykonuj weryfikację (PS i/lub test operacji).

zgłoszenie 1

Podsumowanie:
Po wykonaniu wszystkich czynności z powyższej instrukcji przeczytaj ponownie z zrozumieniem cel ogólny i cele szczegółowe, które znajdują się na pierwszej stronie instrukcji. Jeżeli one zostały niezrealizowane to powtarzaj wykonie tej instrukcji w szkole lub/i w domu do momentu zrealizowania.

Mini-słownik (PowerShell ISE / AD ACL)

Pojęcia ogólne
• DN (Distinguished Name) – pełna ścieżka obiektu AD, np. OU=pAbc,DC=rol00,DC=edu,DC=pl.

• Ścieżka LDAP – format "LDAP://<DN>", używany przez DirectoryEntry do odwołań do obiektów w AD.

• DirectoryEntry – .NET API do bezpośredniej pracy z obiektem AD (właściwości, ObjectSecurity, CommitChanges()).

• ActiveDirectorySecurity (ObjectSecurity) – kolekcja reguł ACL (DACL) przypięta do obiektu AD.

• ActiveDirectoryAccessRule – pojedynczy wpis ACL: kto (SID), jakie prawa (ActiveDirectoryRights), typ (Allow/Deny), dziedziczenie.

• SID (Security Identifier) – unikatowy identyfikator konta używany w ACL; stabilniejszy niż nazwa użytkownika.

ActiveDirectoryRights (użyte w instrukcji)
• ReadProperty – odczyt atrybutów obiektu.

• WriteProperty – modyfikacja (zapis) atrybutów obiektu.

• CreateChild – tworzenie obiektów podrzędnych (np. user, group, OU) w kontenerze/OU.

• DeleteChild – usuwanie obiektów podrzędnych w kontenerze/OU.

• ListChildren – wyświetlanie (listowanie) obiektów podrzędnych wewnątrz kontenera.

• GenericRead – zestaw uprawnień zapewniający odczyt (obejmuje m.in. odczyt właściwości).

• GenericWrite – zestaw uprawnień zapewniający zapis (obejmuje m.in. zapis właściwości).

AccessControlType
• Allow – jawne przyznanie uprawnień (stosowane w ćwiczeniu).

• Deny – jawne odebranie/odmowa uprawnień (używać ostrożnie; ma priorytet nad Allow).

ActiveDirectorySecurityInheritance
• All – dziedziczenie wpisu na obiekty podrzędne zgodnie z ustawieniami dziedziczenia (stosowane w ćwiczeniu).

• None – wpis dotyczy tylko bieżącego obiektu (bez dziedziczenia).

• Inne warianty (Children, Descendents, SelfAndChildren) – rzadziej używane w tym ćwiczeniu; pozwalają doprecyzować zakres dziedziczenia.

Cmdlety Active Directory użyte w instrukcji
• Get-ADDomain – zwraca obiekt bieżącej domeny (np. DNSRoot, NetBIOSName).

• Get-ADRootDSE – zwraca metadane katalogu; DefaultNamingContext to DN domeny.

• Get-ADOrganizationalUnit – pobiera OU; parametry: -LDAPFilter, -SearchBase.

• Get-ADUser – pobiera użytkownika; wartość SID dostępna po -Properties SID.

• New-ADUser – tworzy konto użytkownika (wymaga odpowiednich parametrów i uprawnień).

• Remove-ADUser – usuwa konto użytkownika.

• Get-Credential – pobiera poświadczenia w bezpieczny sposób (SecureString).

Operatory i konstrukcje PowerShell

• -bor – bitowe OR do łączenia wielu flag ActiveDirectoryRights w jeden zestaw.

• foreach (…) { … } – iteracja po elementach (np. po liście użytkowników).

• Where-Object -match 'puz1|puz2' – filtrowanie wpisów ACL po nazwie tożsamości (wyrażenie regularne).

• Select-Object – wybór najważniejszych właściwości do wyświetlenia (np. Rights, AccessControlType, InheritanceType).

• CommitChanges() – utrwalenie zmian w AD (bez tego zmiany pozostają tylko w pamięci).

Dobre praktyki w pracy z ACL (dla ucznia i administratora)

• Minimalne uprawnienia: nadawaj tylko to, co jest potrzebne (np. cztery prawa na OU=pAbc zamiast GenericWrite).

• Weryfikacja dwutorowa: sprawdzaj ObjectSecurity.Access w PowerShell i Security > Advanced w GUI.

• Testuj na poświadczeniach delegowanych kont (Get-Credential + New-/Remove-ADUser).

• Dokumentuj zmiany: zapisuj w zeszycie cel, zestaw praw, obiekt docelowy, wynik weryfikacji.