Budapesti Műszaki és Gazdaságtudományi Egyetem
Intelligens rendszerfelügyelet
(VIMIA370)

1P - Címtárak

Házi feladat

Papp Tamás (DPLEAE)
2011. május 22.

1 Bevezető

Ezt a dokumentációt Intelligens rendszerfelügyelet tárgy első pót-házi feladatának megoldásához készítettem.

1.1 A házi feladat célja

A házi feladat célja Windows címtárkezelési ismeretek elsajátítása és címtárkezelési feladatok megoldása Powershell használatának a segítségével.

A feladat megoldásához létre kellett hoznom egy Active Directory tesztkörnyezetet és magát a feladatkiírásnak megfelelő PS scriptet.

1.2 A script felépítése

A feladat megoldását több lépésre bontottam. Ezek a script bemeneti paramétereinek ellenőrzése, menedzselt UO-k megkeresése, hash-tábla építése a menedzserekről, csoportok transzformálása felhasználókká, jogosultságot sértő felhasználók összegyűjtése és a HTML kód előállítása, majd kiíratása fájlba.

1.2.1 Paraméterek ellenőrzése

A script két paraméter fogadására lett felkészítve és ezek mindegyike kötelező. Ezen paraméterek a feladatkiírásnak megfelelően: -Organization, -OutFile
Ha bármelyik hiányzik, akkor a script futása befejeződik.

```
if ((!$Organization) -or (!$OutFile))
{
        Write-Host "Mandatory parameters are missing." -
foregroundcolor yellow
        Write-Host "You must set -Organization, and -OutFile
parameters." -foregroundcolor yellow
        exit
}
```

Ha az –OutFile paraméterként megadott fájl létezik, akkor is befejeződik a script futása. Ezt a vizsgálatot a Test-Path cmdlet segítségével lehet elvégezni.

```
if (Test-Path $OutFile)
{
        Write-Host "OutFile can not be an existing file." $_ -
foregroundcolor yellow
        exit
}
```

Továbbá, ha olyan OU-t kap, ami nem létezik, akkor is hibajelzést ír ki a kimenetre. Ehhez már szükséges Active Directory lekérdezés használata, aminek működését a dokumentációm következő pontján részletezem. Ha ez a lekérdezés hibát dob, akkor nem található meg a megadott UO a rendszerben.

```
$path1="LDAP://"+$Organization
```

```
$objOU1 = New-Object System.DirectoryServices.DirectoryEntry($path1)

$objSearcher1 = New-Object
System.DirectoryServices.DirectorySearcher
$objSearcher1.SearchRoot = $objOU1
$objSearcher1.SearchScope = "Base"

try
{
      $results = $objSearcher1.FindOne()
} catch [Exception]
{
            Write-Host $_ -foregroundcolor yellow
         Write-Host "OU does not exist." -foregroundcolor yellow
         exit
}
```

1.2.2 Segédfüggvényeim

1.2.2.1 myGet-LDAPResult függvény

Ahhoz, hogy az AD-ben objektumokat tudjak keresni létrehoztam a myGet-LDAPResult segédfüggvényt.

```
function myGet-LDAPResult {
param ( [String]$OUnit, [String]$Filter, [String]$SearchScope,
[Array]$Proplist)
```

Paraméterei:

- OUnit: Az OU DN-je, amin belül keresni akarunk
- Filter: szűrési feltélek
- SearchScope: a keresés típusa (Base, OneLevel, SubTree)
- Proplist: a tulajdonságok listája, melyeket az egyes objektumokhoz le szeretnénk kérdezni

A függvény működése:

Ahhoz, hogy az Active Directory rendszerben keresni tudjak, az alábbi objektumokra volt szükségem:

```
$objOU = New-Object
System.DirectoryServices.DirectoryEntry("LDAP://"+$OUnit)
```

Ez az osztály egy Active Directory objektumot reprezentál. A konstruktor paramétereként megadhatunk egy gyökérobjektumot, ahonnan a keresést indítani szeretnénk.

```
$objSearcher = New-Object System.DirectoryServices.DirectorySearcher
$objSearcher.SearchRoot = $objOU
$objSearcher.PageSize = 1000
$objSearcher.Filter = $Filter
```

```
$objSearcher.SearchScope = $SearchScope
```

A tényleges kereséshez egy DirectorySearcher objektumot használtam. A keresés gyökereként megadtam az imént létrehozott DirectoryEntry objektumot. A Filter és SearchScope attribútumok értékét a paraméterként kapott értékek alapján állítottam be. Továbbá beállítottam, hogy a DirectorySearcher az objektumok mely attribútumait adja vissza a függvényem Proplist paramétere alapján.

```
#$l, csak h ne írja ki a return-t
foreach ($i in $Proplist){$l =
$objSearcher.PropertiesToLoad.Add($i)}

return $objSearcher.FindAll()
```

A keresést a DirectorySearcher FindAll() metódusával tudjuk elindítani. Ezt adja vissza a myGet-LDAPResult függvényem.

1.2.2.2 myGet-GroupMembers függvény

A myGet-GroupMembers függvény végzi egy adott csoport feloldását, illetve transzformálását felhasználókká. Ha egy csoporton belül szerepel egy, vagy több csoport, akkor feloldja azokat is. Egy OU-t akkor minősítek csoportnak, ha létezik beállított member attribútuma, melyet egy Active Directory lekérdezéssel érek el, ha létezik. A lekérdezés eredményeként egyszerre csak egy adott objektum member tulajdonságára van szükségünk. A Filter és SearchScope attribútumok értékeit ennek megfelelően állítottam be.

```
function myGet-GroupMembers {
param ( [String]$ou )
process {
      $path="LDAP://"+$ou
      $objOU = New-Object
System.DirectoryServices.DirectoryEntry($path)
      $objSearcher = New-Object
System.DirectoryServices.DirectorySearcher
      $objSearcher.SearchRoot = $objOU
      $objSearcher.PageSize = 1000
      $objSearcher.Filter = "(&(member=*))"
      $objSearcher.SearchScope = "Base"
      $colProplist = "member"
      #$l, csak h ne írja ki a return-t
      foreach ($i in $colPropList){$l =
$objSearcher.PropertiesToLoad.Add($i)}
      $results = $objSearcher.FindAll()

      $members = ""
      foreach ($item in $results)
      {
            $objItem = $item.Properties
            $members += $objItem.member
      }
```

Ha csoportot adtunk meg bemeneti paraméterként, akkor a FindAll() függvény eredményeként visszaadott érték tartalmazza a csoporthoz tartozó felhasználókat, illetve esetlegesen újabb csoportokat is. A kapott tagokat hozzáfűzöm a $members stringhez egymás után.

Ha nem csoportot adtunk meg bemeneti paraméterként, akkor a $members string hossza 0 marad.

Ha léteznek tagok, akkor előállítom ezek DN-jét ($user), majd meghívom rájuk egyesével rekurzívan a csoportfeloldást végző myGet-GroupMembers függvényt. Ezek eredményét stringként összefűzöm, majd a függvény visszatér ezzel az értékkel, vagyis az általa tartalmazott felhasználók listájával. Előbb utóbb, miden csoportból egy felhasználóhoz jutunk, amikor magával a paraméterként kapott felhasználó DN-jével tér vissza a függvény.

```
$memberusers = ""
if ($members.length -gt 0)
{
      while($members.IndexOf("CN=", 0) -eq 0)
      {
            if($members.IndexOf("CN=", 1) -gt 0)
            {
                  $user = $members.Substring(0,$members.IndexOf("
CN="))
                  $members = $members.Remove(0,$members.IndexOf("
CN=")+1)
                  $memberusers += myGet-GroupMembers -ou $user
                  $memberusers += " "
            } else
            {
                  $user = $members
                  $members = ""
                  $memberusers += myGet-GroupMembers -ou $user
                  return $memberusers
            }
      }
} else
{
      return $ou
}
```

1.2.3 Menedzselt UO-k megkeresése

A menedzselt OU-k megkeresését a korábban létrehozott myGet-LDAPResult függvényem megfelelő paraméterezésével és meghívásával valósítottam meg:

```
$colProplist = "name", "managedby", "distinguishedname"
$colResults = myGet-LDAPResult -OUnit $Organization -Filter
"(&(ou=*)(managedby=*))" -SearchScope "Subtree" -Proplist
$colProplist
```

- OUnit: a script paramétereként megadott OU
- Filter: "(&(ou=*)(managedby=*))" csak a managedby tulajdonsággal megadott OU-kat szeretnénk visszakapni
- SearchScope: Subtree, azaz az egész részfán keresünk
- Proplist: a tulajdonságok listája, melyeket az egyes objektumokhoz le szeretnénk kérdezni: név, managedby, DN

1.2.4 Hash-tábla építése a menedzserekről

A programnak ezen a pontján rendelkezésünkre áll egy objektum lista a menedzselt OU-kről($colResults). Ezen objektumoknak a DN-jére és a managedby attribútumban beállított felhasználó, vagy csoport DN-jére van szükségünk. Mivel a kimeneti fájlban minden felhasználónak csak egyszer szabad szerepelnie, ezért hash-tábla használata mellett döntöttem. Az egyes kulcsok, az egyes menedzser felhasználók, vagy csoportok, a hozzájuk tartozó értékek pedig, az egy ilyen által menedzselt UO-kat tartalmazzák.

```
#$managers: hash-tábla
#kulcs: az OU-kon beállított managedby userek, vagy csoportok
#érték: a user, vagy csoport által menedzselt UO(-k) DN-je
$managers = @{}
foreach ($objResult in $colResults)
{
      $objItem = $objResult.Properties
      $key = '"'+$objItem.managedby+'"'
      if ($managers.ContainsKey($key) -eq $True)
      {
            $managers.Set_Item($key, $managers.get_Item($key) +
$objItem.distinguishedname)
      } else
      {
            $managers.Set_Item($key, $objItem.distinguishedname)
      }
}
```

1.2.5 Csoportok transzformálása felhasználókká

A feladatkiírás alapján csoportok esetén a csoport tagjaira el kell végezni a vizsgálatot, és nem a csoportot, hanem a felhasználót kell továbbra is jelenteni a kimenetben. Ehhez szükséges a csoportok feloldása, melyet a korábban részletezett myGet-GroupMembers függvény valósít meg.

```
#$justUsers: managers-hez hasonló hash-tábla, de csoportok helyett a
csoportba tartozó usereket tartalmazza
#A csoportok transzformálása felhasználókká
$justUsers = @{}
foreach ($key in $managers.Keys)
{
$value = $managers.get_Item($key)
$keymembers = myGet-GroupMembers -ou $key.Trim('"')
```

Ezen a ponton a $keymembers értékeként rendelkezésünkre áll egy adott UO összes felhasználó típusú menedzsere. A következő ciklus ezeket a felhasználókat veszi sorra. Addig fut, amíg van feldolgozatlan felhasználó.

```
while($keymembers.IndexOf("CN=", 0) -eq 0)
{
      if($keymembers.IndexOf("CN=", 1) -gt 0)
      {
            $user = $keymembers.Substring(0,$keymembers.IndexOf("
CN="))
            $keymembers = $keymembers.Remove(0,$keymembers.IndexOf("
CN=")+1)
      } else
      {
            $user = $keymembers
            $keymembers = ""
      }
```

$user értékeként rendelkezésünkre áll egy adott felhasználó. Ha még nem szerepel a $justUsers hash-táblában, akkor hozzáadjuk és beállítjuk az értékét az aktuális OU ($value) értékére, amelyikhez az AD-ból lekérdezett adatok alapján menedzselési joga van. Ha szerepel, akkor csak hozzáfűzzük, az új OU-t ($value) az eddigiekhez, de csak ha még nem szerepel köztük.

```
      if ($justUsers.ContainsKey($user) -eq $True)
      {
            $isContainValue = $False
            foreach($item in $justUsers.get Item($user))
            {
                  foreach($item2 in $value)
                  {
                        if ($item.ToString() -eq $item2)
                        {
                              $isContainValue = $True
                        }
                  }
            }
            if ($isContainValue -eq $False)
            {
                  $justUsers.Set_Item($user, $justUsers.get_Item($user) +
$value)
            }
      } else
      {
            $justUsers.Set_Item($user, $value)
      }
}
}
```

Végül a $justUsers hash-tábla kulcsaiként és értékeiként megkapjuk a redundanciát már nem tartalmazó listát az egyes felhasználókról, és arról, hogy melyik OU-khoz van hozzáférési joguk.

1.2.6 Jogosultságot sértő felhasználók összegyűjtése

A feladatkiírásban szereplő házirend röviden:

- egy OU közvetlen gyerek entitásai (group, user) lehetnek menedzserek

- egy OU-ban található OU-kat a közvetlen szülő, vagy akár közvetett szülő entitásai is menedzselhetik

- egy OU szülő OU-ját a gyerekben található entitások nem menedzselhetik

Az ellenőrzés folyamata: az adott felhasználó DN-ből levágom a CN tagot, így megkapom, hogy melyik UO-ban található. Ha a kapott érték nem tartalmazza a felhasználó által menedzselt UO-t, akkor sérül a házirend. Ez alapján építem fel a $violatiors táblát, továbbá az egyes felhasználókhoz tartozó UO-kat
 tag-el választom el, a későbbi kiíratás megkönnyítése miatt.

```
#$violatiors: a jogosultságot sértő felhasználók, és a hozzájuk
tartozó UO DN-ek hash-táblája
$violatiors = @{}
foreach ($key in $justUsers.Keys)
{
      $values = $justUsers.get_Item($key)
      foreach ($value in $values)
      {
            if ($value.Contains($key.Substring($key.IndexOf(',')+1))
-eq $False)
            {
                  if ($violatiors.ContainsKey($key) -eq $True)
                  {
                        $violatiors.Set_Item($key,
$violatiors.get_Item($key) + "<br>" + $value)
                  } else
                  {
                        $violatiors.Set_Item($key, $value)
                  }
            }
      }
}
```

1.2.7 HTML kód előállítása

Kiíratás előtt felhasználói DN szerint rendezem a rendelkezése álló hash-táblát, majd ezen végigiterálva felépítem a megfelelő html kódot. Ha ez elkészült, akkor a tartalmát kiírom a megadott fájlban.

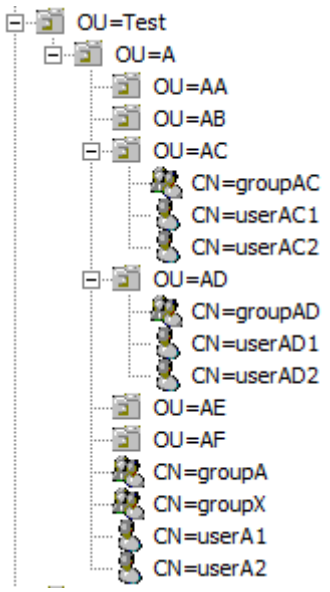
```
#$sortedResult: rendezett $violatiors tábla
$sortedResult = $violatiors.GetEnumerator() | Sort-Object Name
#$htmlresult: $sortedResult tömb transzformálása megfelelő html
kóddá
$htmlresult = "<html><body><table
border=1><tr><td>Manager</td><td>Violations</td></tr>"
foreach ($item in $sortedResult)
{
      $htmlresult +=
"<tr><td>"+$item.Name+"</td><td>"+$item.Value+"</td></tr>"
```

```
}
$htmlresult += "</table></body></html>"
$htmlresult > $OutFile
```

2 A megoldás tesztelése

2.1 Tesztkörnyezet kialakítása

Ahhoz, hogy a létrehozott script működését ellenőrizni tudjam, létrehoztam egy AD objektum-hierarchiát. Igyekeztem a lehető legtöbb esetet figyelembe venni. Ez a teszt-hierarchia a mellékelt képen látható:



Az egyes csoportok tagjai (a teljes DN helyett az átláthatóság érdekében csak felhasználó-, vagy csoportnevet írok):

- groupA: userA1, userA2
- groupAC: userAC1, userAC2
- groupAD: userAD1, userAD2
- groupX: userA1, userAC2, group AC, groupAD

Az egyes OU-k managedby attribútumainak értékei:

- AA: userA1
- AB: groupA
- AC: userAC1
- AD: groupAD
- AE: userAC1
- AF: groupX
- A: groupX

Az első 4 csoportban nem sérülnek a házirend szabályai, viszont az utolsó 3-ban (AE, AF) igen. AE-ben userAC1 nem lehet menedzser a feladatkiírásban szereplő házirendszabályok alapján. Továbbá AF-ben és A-ban userAC2, userAC1, userAD1 és userAD2 sem lehet menedzser.

2.2 Paraméterek tesztelése

2.2.1 Hiányzó paraméterek

Parancs:

```
PS>.\Check-AdminRights.ps1
```

Eredmény:

```
Mandatory parameters are missing.
You must set -Organization, and -OutFile parameters.
```

Parancs:

```
PS>.\Check-AdminRights.ps1 -Organization "OU=Test,DC=irfhf,DC=local"
```

Eredmény:

```
Mandatory parameters are missing.
You must set -Organization, and -OutFile parameters.
```

Parancs:

```
PS>.\Check-AdminRights.ps1 -OutFile letezofajl.html
```

Eredmény:

```
Mandatory parameters are missing.
You must set -Organization, and -OutFile parameters.
```

Mindhárom teszteset az elvárt eredményt hozta.

2.2.2 Ha létezik a megadott fájl

Parancs:

```
PS>.\Check-AdminRights.ps1 -Organization "OU=Test,DC=irfhf,DC=local" -OutFile letezofajl.html
```

Eredmény:

```
OutFile can not be an existing file.
```

2.2.3 Nem létező OU-t adunk meg

Parancs:

```
PS>.\Check-AdminRights.ps1 -Organization "OU=Nemletezo,DC=irfhf,DC=local" -OutFile result.html
```

Eredmény:

```
There is no such object on the server.


OU does not exist.
```

A teszteset az elvárt eredményt hozta.

2.3 Teljes funkcionalitás tesztelése

Script tesztelése:

```
PS>.\Check-AdminRights.ps1 -Organization "OU=Test,DC=irfhf,DC=local" -OutFile result.html
```

A létrehozott result.html fájl tartalma böngészőben nézve:

Manager	Violations
CN=userAC1,OU=AC,OU=A,OU=Test,DC=irfhf,DC=local	OU=A,OU=Test,DC=irfhf,DC=local OU=AF,OU=A,OU=Test,DC=irfhf,DC=local OU=AE,OU=A,OU=Test,DC=irfhf,DC=local
CN=userAC2,OU=AC,OU=A,OU=Test,DC=irfhf,DC=local	OU=A,OU=Test,DC=irfhf,DC=local OU=AF,OU=A,OU=Test,DC=irfhf,DC=local
CN=userAD1,OU=AD,OU=A,OU=Test,DC=irfhf,DC=local	OU=A,OU=Test,DC=irfhf,DC=local OU=AF,OU=A,OU=Test,DC=irfhf,DC=local
CN=userAD2,OU=AD,OU=A,OU=Test,DC=irfhf,DC=local	OU=A,OU=Test,DC=irfhf,DC=local OU=AF,OU=A,OU=Test,DC=irfhf,DC=local

A script futása a korábban részletezett elvárt eredményt hozta.

3 A teljes elkészített script

```
# Name:   Check-AdminRights.ps1
# Author: Papp Tamás
# Date:   2011.05.21.
# Desc:
# Params:
#      -Organization   <OU DN>                  :Mandatory
#      -OutFile          <output file name>     :Mandatory

param(
    [String] $Organization,
      [String] $OutFile
)

function myGet-GroupMembers {
param ( [String]$ou )
process {
      $path="LDAP://"+$ou
      $objOU = New-Object System.DirectoryServices.DirectoryEntry($path)
      $objSearcher = New-Object System.DirectoryServices.DirectorySearcher
      $objSearcher.SearchRoot = $objOU
      $objSearcher.PageSize = 1000
      $objSearcher.Filter = "(&(member=*))"
      $objSearcher.SearchScope = "Base"
      $colProplist = "member"
      #$l, csak h ne írja ki a return-t
      foreach ($i in $colPropList){$l =
$objSearcher.PropertiesToLoad.Add($i)}
      $results = $objSearcher.FindAll()

      $members = ""
      foreach ($item in $results)
      {
            $objItem = $item.Properties
            $members += $objItem.member
      }

      $memberusers = ""
      if ($members.length -gt 0)
      {
```

```
            while($members.IndexOf("CN=", 0) -eq 0)
            {
                  if($members.IndexOf("CN=", 1) -gt 0)
                  {
                        $user = $members.Substring(0,$members.IndexOf("
CN="))
                        $members = $members.Remove(0,$members.IndexOf("
CN=")+1)
                        $memberusers += myGet-GroupMembers -ou $user
                        $memberusers += " "
                  } else
                  {
                        $user = $members
                        $members = ""
                        $memberusers += myGet-GroupMembers -ou $user
                        return $memberusers
                  }
            }
      } else
      {
            return $ou
      }
}
}

function myGet-LDAPResult {
param ( [String]$OUnit, [String]$Filter, [String]$SearchScope,
[Array]$Proplist)
process {
      $objOU = New-Object
System.DirectoryServices.DirectoryEntry("LDAP://"+$OUnit)

      $objSearcher = New-Object System.DirectoryServices.DirectorySearcher
      $objSearcher.SearchRoot = $objOU
      $objSearcher.PageSize = 1000
      $objSearcher.Filter = $Filter
      $objSearcher.SearchScope = $SearchScope

      $colProplist = $Proplist
      #$l, csak h ne írja ki a return-t
      foreach ($i in $colPropList){$l =
$objSearcher.PropertiesToLoad.Add($i)}

      return $objSearcher.FindAll()
      }
}

if ((!$Organization) -or (!$OutFile))
{
        Write-Host "Mandatory parameters are missing." -foregroundcolor
yellow
        Write-Host "You must set -Organization, and -OutFile parameters." -
foregroundcolor yellow
        exit
}

if (Test-Path $OutFile)
{
        Write-Host "OutFile can not be an existing file." $_ -
foregroundcolor yellow
        exit
```

```
}


$path1="LDAP://"+$Organization
$objOU1 = New-Object System.DirectoryServices.DirectoryEntry($path1)

$objSearcher1 = New-Object System.DirectoryServices.DirectorySearcher
$objSearcher1.SearchRoot = $objOU1
$objSearcher1.SearchScope = "Base"

try
{
      $results = $objSearcher1.FindOne()
} catch [Exception]
{
            Write-Host $_ -foregroundcolor yellow
        Write-Host "OU does not exist." -foregroundcolor yellow
        exit
}


$colProplist = "name", "managedby", "distinguishedname"
$colResults = myGet-LDAPResult -OUnit $Organization -Filter
"(&(ou=*)(managedby=*))" -SearchScope "Subtree" -Proplist $colProplist
if ($colResults -eq $null)
{
        Write-Host "There are no managedby property adjusted." -
foregroundcolor yellow
        exit
}

#$managers: hash-tábla
#kulcs: az OU-kon beállított managedby userek, vagy csoportok
#érték: a user, vagy csoport által menedzselt UO(-k) DN-je
$managers = @{}
foreach ($objResult in $colResults)
{
      $objItem = $objResult.Properties
      $key = '"'+$objItem.managedby+'"'
      if ($managers.ContainsKey($key) -eq $True)
      {
            $managers.Set_Item($key, $managers.get_Item($key) +
$objItem.distinguishedname)
      } else
      {
            $managers.Set_Item($key, $objItem.distinguishedname)
      }
}

#$justUsers: managers-hez hasonló hash-tábla, de csoportok helyett a
csoportba tartozó usereket tartalmazza
#A csoportok transzformálása felhasználókká
$justUsers = @{}
foreach ($key in $managers.Keys)
{
$value = $managers.get_Item($key)
$keymembers = myGet-GroupMembers -ou $key.Trim('"')
while($keymembers.IndexOf("CN=", 0) -eq 0)
{
      if($keymembers.IndexOf("CN=", 1) -gt 0)
      {
```

```
            $user = $keymembers.Substring(0,$keymembers.IndexOf(" CN="))
            $keymembers = $keymembers.Remove(0,$keymembers.IndexOf("
CN=")+1)
      } else
      {
            $user = $keymembers
            $keymembers = ""
      }
      if ($justUsers.ContainsKey($user) -eq $True)
      {
            $isContainValue = $False
            foreach($item in $justUsers.get_Item($user))
            {
                  foreach($item2 in $value)
                  {
                        if ($item.ToString() -eq $item2)
                        {
                              $isContainValue = $True
                        }
                  }
            }
            if ($isContainValue -eq $False)
            {
                  $justUsers.Set_Item($user, $justUsers.get_Item($user) +
$value)
            }
      } else
      {
            $justUsers.Set_Item($user, $value)
      }
}
}

#$violatiors: a jogosultságot sértő felhasználók, és a hozzájuk tartozó UO
DN-ek hash-táblája
$violatiors = @{}
foreach ($key in $justUsers.Keys)
{
      $values = $justUsers.get_Item($key)
      foreach ($value in $values)
      {
            if ($value.Contains($key.Substring($key.IndexOf(',')+1)) -eq
$False)
            {
                  if ($violatiors.ContainsKey($key) -eq $True)
                  {
                        $violatiors.Set_Item($key,
$violatiors.get_Item($key) + "<br>" + $value)
                  } else
                  {
                        $violatiors.Set_Item($key, $value)
                  }
            }
      }
}

#$sortedResult: rendezett $violatiors tábla
$sortedResult = $violatiors.GetEnumerator() | Sort-Object Name
#$htmlresult: $sortedResult tömb transzformálása megfelelő html kóddá
$htmlresult = "<html><body><table
border=1><tr><td>Manager</td><td>Violations</td></tr>"
```

```
foreach ($item in $sortedResult)
{
      $htmlresult +=
"<tr><td>"+$item.Name+"</td><td>"+$item.Value+"</td></tr>"
}
$htmlresult += "</table></body></html>"
$htmlresult > $OutFile
```