Iron Scripter 2018: Prequel 8

The Puzzle

Greetings Iron Scripters. The countdown to Iron Scripter continues with only a few weeks left before the main event.

In this weeks challenge you're required to:

- Create a user on the local machine – named Bill Bennsson
- Create a second user on the local machine – named Andy Pandien
- Create a folder in the root of the C:\ drive. Folder name is SpecialFolder
- Create a file in the folder named SpecialFile.txt
- User Bill Bennsson should be able to read and write the contents of the file – including change the contents
- User Andy Pandien should be able only read the file
- NO other users should be able to access the file

All of the above tasks should be performed in a script or function that can be used on multiple machines if required.

Use best practice if it doesn't conflict with your faction's aims. The solution must be acceptable to your faction:

- Daybreak Faction - beautiful code
- Flawless Faction - flawless code
- Battle Faction - good enough to get the job done

Good luck and good coding.

The Commentary

This week's challenge needs a straight forward linear process to solve. Let's start by creating the users, folder and file.

```powershell
$password = Read-Host -Prompt 'Password' -AsSecureString

## create users
New-LocalUser -FullName 'Bill Bennsson' -Name 'BillB' -Password $password `
-Description 'User for Iron Scripter Prequel Puzzle 8'

New-LocalUser -FullName 'Andy Pandien' -Name 'AndyP' -Password $password `
-Description 'User for Iron Scripter Prequel Puzzle 8'

## create folder
New-Item -Path C:\ -Name SpecialFolder -ItemType Directory
New-Item -Path C:\SpecialFolder  -Name SpecialFile.txt -ItemType File
```

```
## put something in file
Get-Process | Out-File -FilePath C:\SpecialFolder\SpecialFile.txt
```

Windows 10 has a module - Microsoft.PowerShell.LocalAccounts – that you can use to manage local users and groups. If you don't have access to that module there are a couple of modules on the PowerShell gallery that might help. You could use the System.DirectoryServices.AccountManagement classes or even if all else fails and you can't think of anything else you could scrape the bottom of the barrel and revert to the legacy command net user.

Create a secure string, using Read-Host, that contains the password. I'm using the same password for both but Daybreak and Flawless factions might prefer to have separate passwords. Local users don't need as much information as Active Directory user accounts. I've added a description so that I remember to remove the accounts at some stage.

Creating the folder and file shouldn't have any surprises. I've piped the Get-Process output into the files just so that it's not empty. You can use any data that you want.

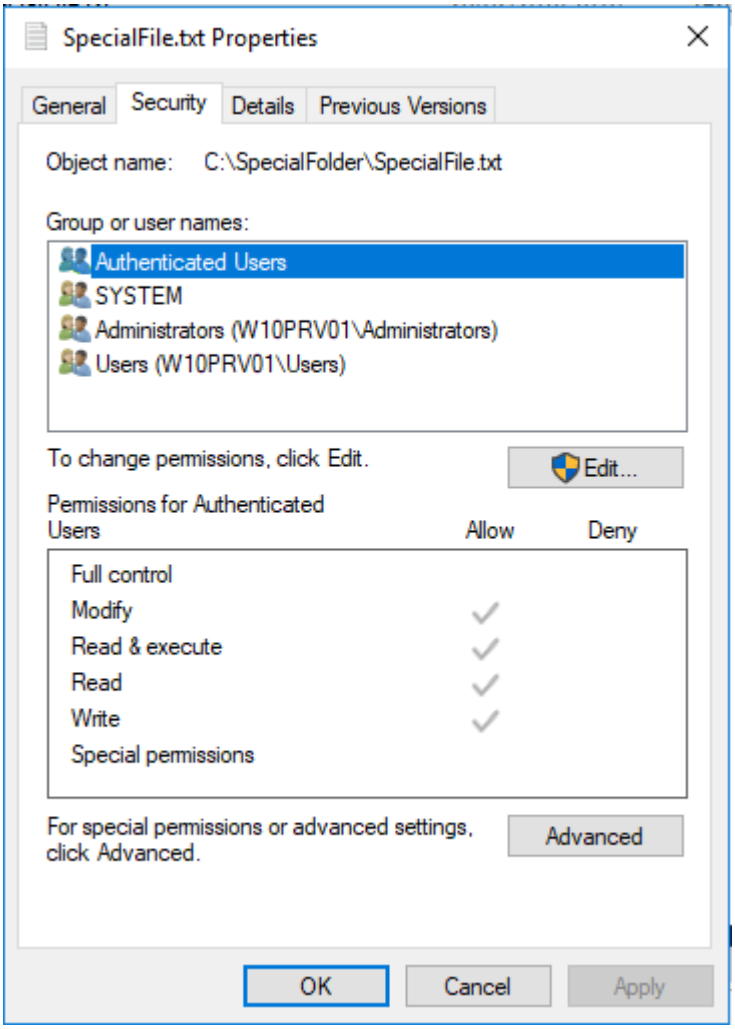
The fun comes when we start to look at the access controls. Starting with Get-Acl

```
PS>  Get-Acl -Path C:\SpecialFolder\SpecialFile.txt


    Directory: C:\SpecialFolder


Path            Owner                  Access
----            -----                  ------
SpecialFile.txt BUILTIN\Administrators BUILTIN\Administrators Allow  FullControl...
```

That doesn't tell us a lot so let's dig a bit deeper

```
PS>  Get-Acl -Path C:\SpecialFolder\SpecialFile.txt | select -ExpandProperty Access


FileSystemRights  : FullControl
AccessControlType : Allow
IdentityReference : BUILTIN\Administrators
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : FullControl
AccessControlType : Allow
IdentityReference : NT AUTHORITY\SYSTEM
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : ReadAndExecute, Synchronize
AccessControlType : Allow
IdentityReference : BUILTIN\Users
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : Modify, Synchronize
AccessControlType : Allow
IdentityReference : NT AUTHORITY\Authenticated Users
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None
```

That matches what you'd see using file explorer:



The way I usually approach these problems is to set the additional access controls I want and then remove the unwanted ones.

BillB is supposed to have read/write permissions and AndyP just read. Other users will have their access revoked.

The process of adding permissions involves getting the current ACL; building an access rule for each permission and adding it to the ACL then applying the new ACL.

The possible permissions are defined in System.Security.AccessControl.FileSystemRights:

```
PS> [enum]::GetNames('System.Security.AccessControl.FileSystemRights')
ListDirectory
ReadData
WriteData
CreateFiles
CreateDirectories
AppendData
ReadExtendedAttributes
WriteExtendedAttributes
Traverse
ExecuteFile
DeleteSubdirectoriesAndFiles
ReadAttributes
WriteAttributes
Write
Delete
ReadPermissions
Read
ReadAndExecute
Modify
ChangePermissions
TakeOwnership
```

```
Synchronize
FullControl
```

You need to add the rules individually:

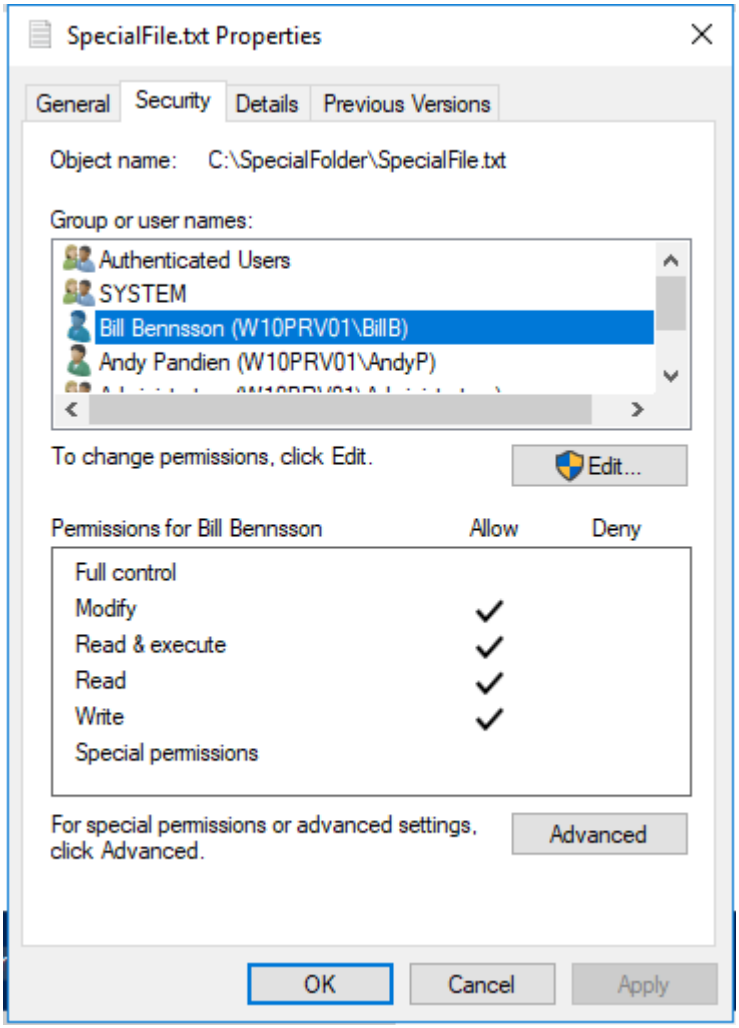
```
## get current ACL
$acl = Get-Acl -Path C:\SpecialFolder\SpecialFile.txt

$user = 'BillB'
$rights = 'ReadAndExecute', 'Write', 'Modify', 'Read'

foreach ($right in $rights) {
  $RuleBillB = New-Object -TypeName System.Security.AccessControl.FileSystemAccessRule($user,
$right, 'Allow')
  $acl.AddAccessRule($RuleBillB)
}

$user = 'AndyP'
$rights = 'Read'

foreach ($right in $rights) {
  $RuleAndyP = New-Object -TypeName System.Security.AccessControl.FileSystemAccessRule($user,
$right, 'Allow')
  $acl.AddAccessRule($RuleAndyP)
}

Set-Acl -Path C:\SpecialFolder\SpecialFile.txt -AclObject $acl
```

Start by getting the current rules. Define the user and the permissions. Loop through the permissions adding them to the access rule set. Repeat for the second user. Use Set-Acl to apply the new permissions.

The permissions are now:

```
PS>  Get-Acl -Path C:\SpecialFolder\SpecialFile.txt | select -ExpandProperty Access


FileSystemRights  : Modify, Synchronize
AccessControlType : Allow
IdentityReference : W10PRV01\BillB
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : Read, Synchronize
AccessControlType : Allow
IdentityReference : W10PRV01\AndyP
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : FullControl
AccessControlType : Allow
IdentityReference : BUILTIN\Administrators
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : FullControl
AccessControlType : Allow
IdentityReference : NT AUTHORITY\SYSTEM
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : ReadAndExecute, Synchronize
AccessControlType : Allow
IdentityReference : BUILTIN\Users
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : Modify, Synchronize
AccessControlType : Allow
IdentityReference : NT AUTHORITY\Authenticated Users
IsInherited       : True
InheritanceFlags  : None
PropagationFlags  : None
```

The permissions for BillB look odd so let's check file explorer.



Modify implies Read, Write and Read & execute so it's all good.

One problem is that these permissions are inherited so can't be removed directly.

```
## remove access rights
# get current ACL
$acl = Get-Acl -Path C:\SpecialFolder\SpecialFile.txt

## protect ACL from inheritance
## remove inherited access rules
$acl.SetAccessRuleProtection($true, $false)

Set-Acl -Path C:\SpecialFolder\SpecialFile.txt -AclObject $acl
```

What you need to do is take off the inheritance using SetAccessRuleProtection(). The first parameter protects the ACL from inheritance and the second removes the inherited access rules leaving the permissions as:

```
PS>  Get-Acl -Path C:\SpecialFolder\SpecialFile.txt | select -ExpandProperty Access


FileSystemRights  : Modify, Synchronize
AccessControlType : Allow
IdentityReference : W10PRV01\BillB
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

FileSystemRights  : Read, Synchronize
AccessControlType : Allow
IdentityReference : W10PRV01\AndyP
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None
```

Put it all together and you have this script

```
$password = Read-Host -Prompt 'Password' -AsSecureString

## create users
New-LocalUser -FullName 'Bill Bennsson' -Name 'BillB' -Password $password `
-Description 'User for Iron Scripter Prequel Puzzle 8'

New-LocalUser -FullName 'Andy Pandien' -Name 'AndyP' -Password $password `
-Description 'User for Iron Scripter Prequel Puzzle 8'

## create folder
New-Item -Path C:\ -Name SpecialFolder -ItemType Directory
New-Item -Path C:\SpecialFolder  -Name SpecialFile.txt -ItemType File

## put something in file
Get-Process | Out-File -FilePath C:\SpecialFolder\SpecialFile.txt

## ADD required premissions
## get current ACL
$acl = Get-Acl -Path C:\SpecialFolder\SpecialFile.txt

$user = 'BillB'
$rights = 'ReadAndExecute', 'Write', 'Modify', 'Read'

foreach ($right in $rights) {
  $RuleBillB = New-Object -TypeName System.Security.AccessControl.FileSystemAccessRule($user,
$right, 'Allow')
  $acl.AddAccessRule($RuleBillB)
}

$user = 'AndyP'
$rights = 'Read'

foreach ($right in $rights) {
  $RuleAndyP = New-Object -TypeName System.Security.AccessControl.FileSystemAccessRule($user,
$right, 'Allow')
  $acl.AddAccessRule($RuleAndyP)
}

## REMOVE inherited rights

## protect ACL from inheritance
## remove inherited access rules
$acl.SetAccessRuleProtection($true, $false)

Set-Acl -Path C:\SpecialFolder\SpecialFile.txt -AclObject $acl
```

Battle faction may want to parameterise the script but probably not much more.

Daybreak faction will want to parameterize the script and format the code so that it looks good. Maybe add it to one of the modules created in an earlier puzzle.

In addition, Flawless faction will want to add all the bells and whistles - anything that ensures the code executes flawlessly.

Enjoy! Puzzle 9 will be available around the time you're reading this.