PowerShell Web Access in action (WEB-access)

Дата публикации: 20 апреля 2015 года
https://technet.microsoft.com/ru-ru/windowsserver/dn705874.aspx
Я думаю, многим системным администраторам знакома проблема делегирования полномочий. На предыдущем месте работы я столкнулся с необходимостью предоставить возможность для второй линии техподдержки перезапускать определенные службы на серверах и восстанавливать резервные копии пользовательских площадок. Проблема усугублялась тремя факторами:

1. Вторая линия техподдержки в основной массе использовала Linux.
2. Не хотелось давать возможность подключения по RDP из соображений безопасности.
3. В качестве сервера резервного копирования использовался System Center Data Protection Manager, который требует вхождения в группу локальных администраторов для возможности восстановления резервных копий.

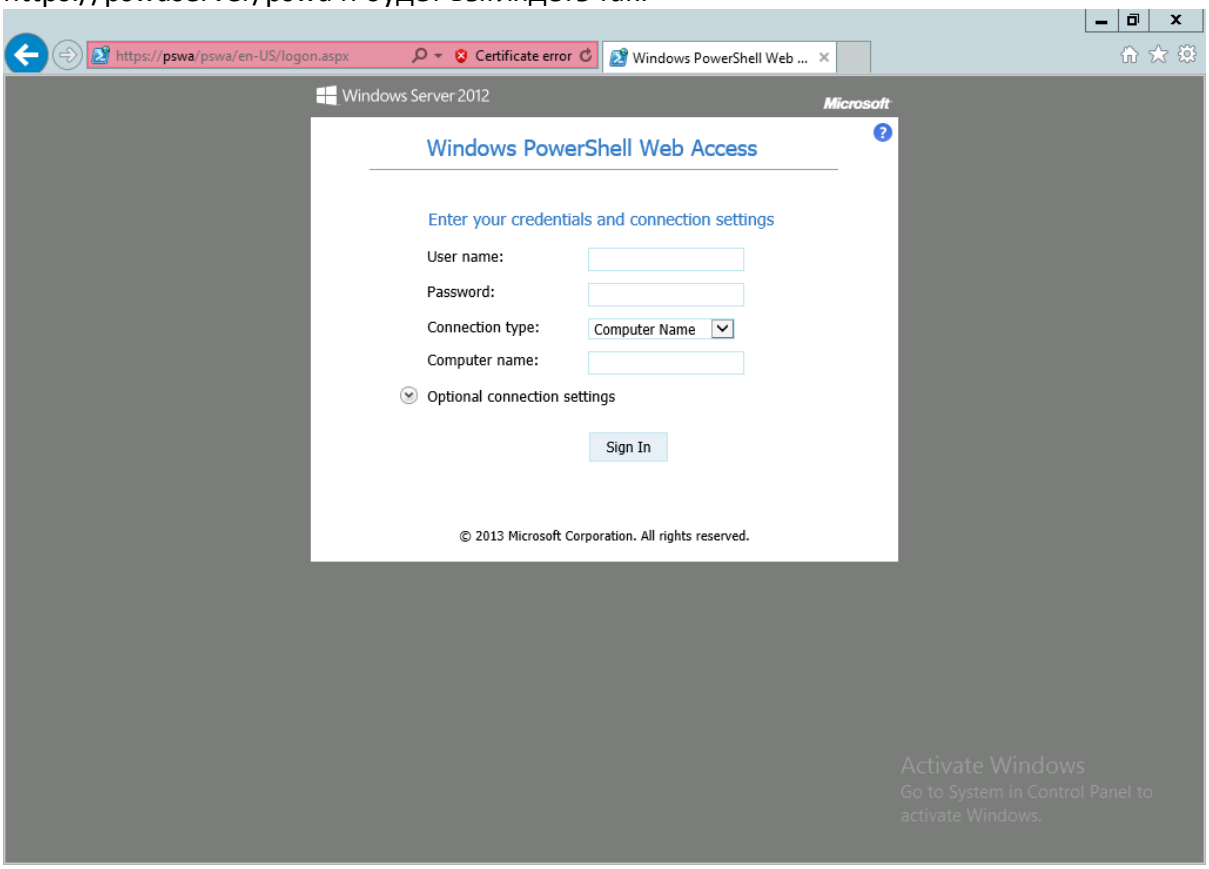
После некоторых раздумий было решено воспользоваться новой возможностью Windows Server 2012, а именно PowerShell Web Access (PSWA) с использованием настраиваемой конфигурации сессий PowerShell и включением, и настройкой CredSSP.
Установка и настройка PowerShell Web Access — процесс несложный.
Добавляем компонент:

```
Install-WindowsFeature -Name WindowsPowerShellWebAccess -IncludeManagementTools
```

Создаем веб-приложение PSWA:

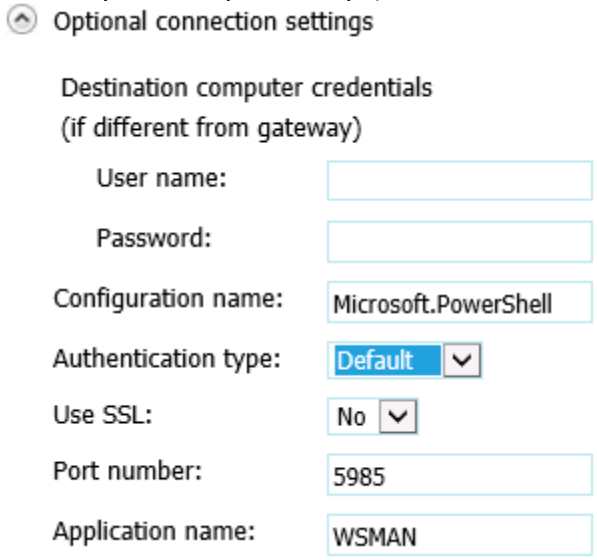
```
Install-PswaWebApplication -UseTestCertificate
```

(для тестов я использовал самоподписанный сертификат) — автоматически установятся все необходимые роли (IIS, .NET и т. д.). Приложение будет доступно по адресу https://pswaserver/pswa и будет выглядеть так:



Username\Password: для получения доступа к удаленной консоли PowerShell необходимо ввести логин/пароль.
Connection type: определяет тип адреса — FQDN или URI. FQDN используется для подключения к сессии powershell при помощи WinRM, URI — для подключения к сессии при помощи веб-прокси (примером может служить Microsoft Exchange). Справедливости ради стоит заметить, что в конечном итоге для подключения используется URI (см. ниже раздел Optional connection settings).

Computer Name: FQDN или URI, в зависимости от выбранного типа подключения.
Optional connection settings: позволяют выбрать дополнительные параметры подключения, такие как имя конфигурации, тип аутентификации, TCP-порт (если изменен стандартный порт WinRM) и имя приложения (из имени компьютера, порта и имени приложения получается URI точки подключения: **transport://server:port/ApplicationName**. В PSWA в качестве транспорта используются http или https):



Теперь создадим правила авторизации для пользователей (по умолчанию никаких правил нет):

1. Правило *For Admins,* позволяющее пользователям, входящим в группу **Domain Admins,** подключаться к любому компьютеру, используя любую конфигурацию сессий PowerShell:

```
Add-PswaAuthorizationRule -UserGroupName "AZURE\Domain Admins" -ComputerName * -ConfigurationName * -
RuleName "For Admins"
```

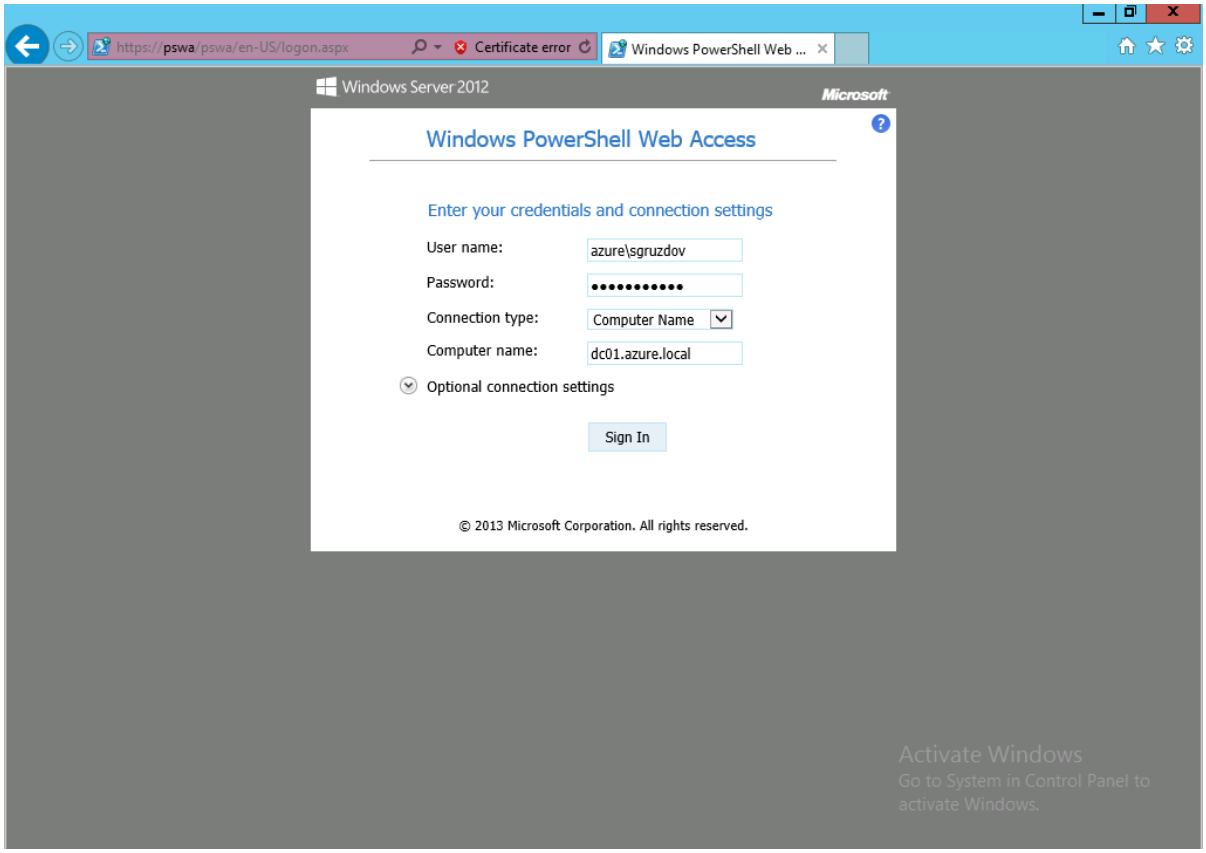
2. Правило *Restricted Access,* позволяющее пользователям, входящим в группу **HelpDesk,** подключаться к любому компьютеру, используя только конфигурацию **HelpDesk** (описание создания конфигурации будет дано ниже):

```
Add-PswaAuthorizationRule -UserGroupName "Azure\HelpDesk" -ComputerName * -ConfigurationName "HelpDesk" -
RuleName "Resticted access"
```

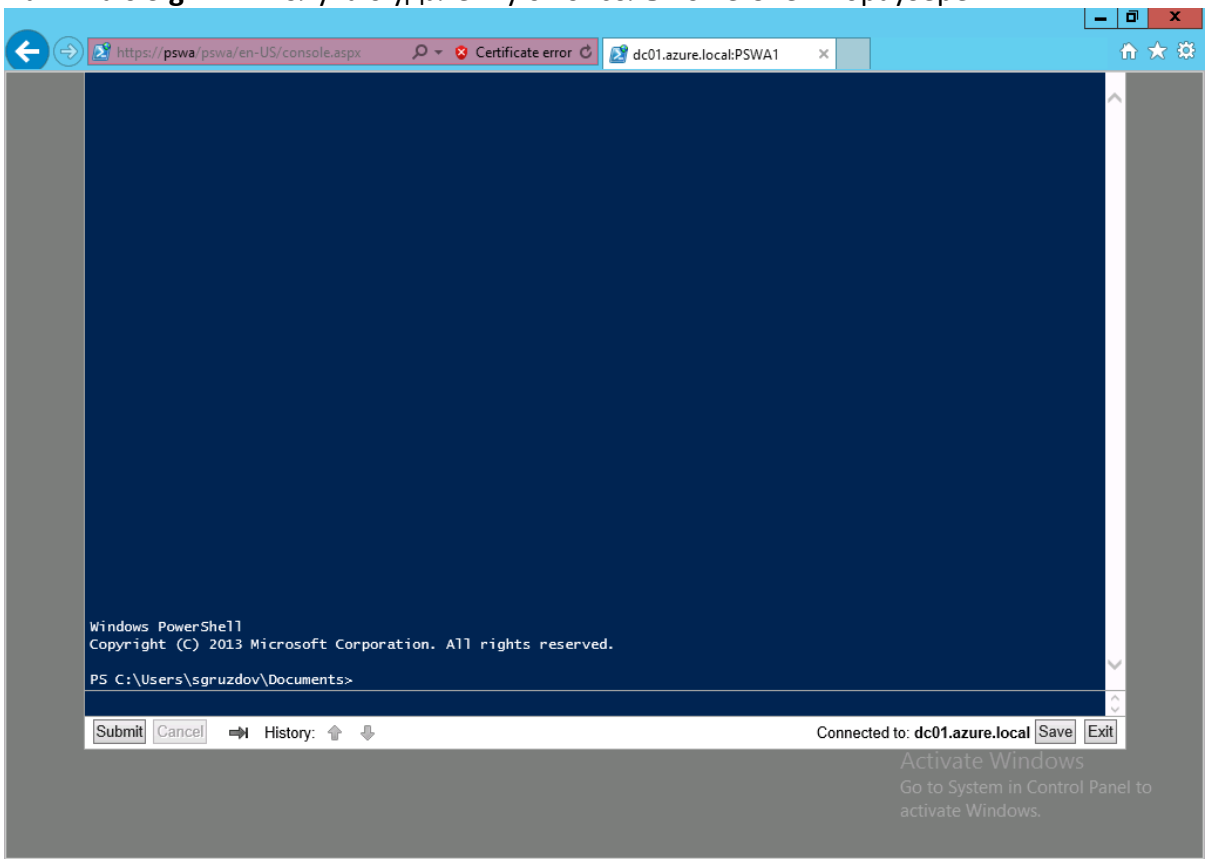
Попробуем подключиться к какому-нибудь серверу. Стоит заметить, что в Windows Server 2012 и выше PowerShell Remoting разрешен по умолчанию, а вот для серверов 2008 R2 и младше нужно предварительно выполнить:

```
Enable-PSRemoting –Force
```

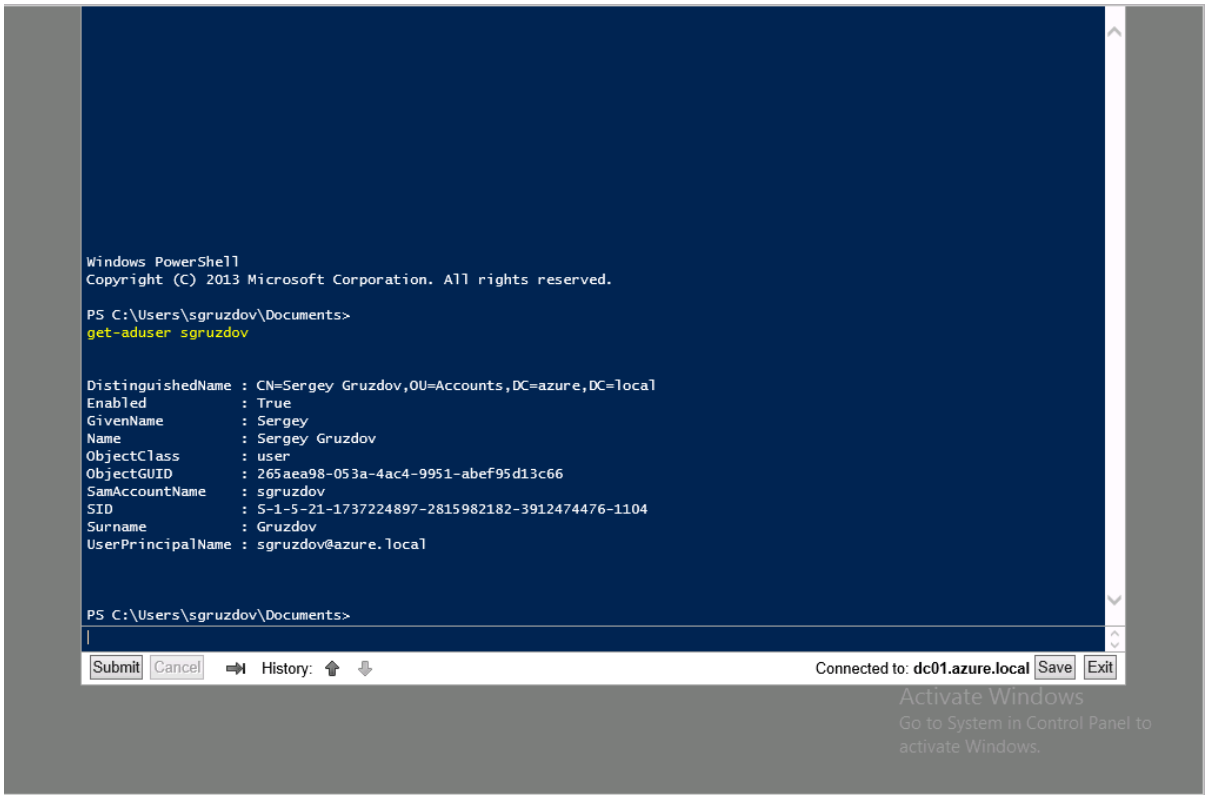
Я подключаюсь к контроллеру домена, используя свою учетную запись, входящую в группу **Domain Admins**:



Нажимаю **Sign In** и получаю удаленную консоль PowerShell в браузере:



Попробуем выполнить какой-нибудь командлет:



Все настроено и работает.
Теперь перейдем к самому интересному: созданию ограниченных сессий PowerShell и кастомизации начальной страницы портала PSWA.

Создание ограниченных сессий

Для примера создадим сессию, к которой могут подключаться пользователи, входящие в группу **HelpDesk,** и сделаем им доступным единственный командлет **Create-MarketingUser**:

```
$HelpDeskSDDL = GetSSDLForADGroup -GroupName "AZURE\HelpDesk";
$cred = Get-Credential("AZURE\sgruzdov")
Register-PSSessionConfiguration -Name "HelpDesk" -StartupScript "c:\scripts\startup.ps1" -
SecurityDescriptorSddl $HelpDeskSDDL -RunAsCredential $Cred -Force;
```

Данный код создает конфигурацию сессии **HelpDesk**, правом на подключение к которой обладают пользователи, входящие в группу **AZURE\HelpDesk,** и которая будет выполняться в контексте пользователя «AZURE\sgruzdov», входящего в группу доменных администраторов, независимо от введенных учетных данных.

Функция для получения SDDL группы:

```
Function GetSSDLForADGroup
{
   param([String]$GroupName = [String]::Empty, [Switch]$AllowDefaultSDDL)
  try
   {
    #retreave the default SSDL as a base
    $defaultSSDL = (Get-Item WsMan:\localhost\service\rootSDDL).Value
    $isContainer = $false
    $isDS = $false
    $SecurityDescriptor = New-Object -
TypeName Security.AccessControl.CommonSecurityDescriptor $isContainer, $isDS, $defaultSSDL

      if (!$AllowDefaultSDDL)
      {
       #remove the default groups (Administrators and Interactive User)
       while($SecurityDescriptor.DiscretionaryAcl.Count -gt 0)
       {
        $ssdl = $SecurityDescriptor.DiscretionaryAcl[0]
        $SecurityDescriptor.DiscretionaryAcl.RemoveAccess(
```

```
            [System.Security.AccessControl.AccessControlType]::Allow,
            $ssdl.SecurityIdentifier,
            $ssdl.AccessMask,
            $ssdl.InheritanceFlags,
            $ssdl.PropagationFlags) | Out-Null
        }
       }

    #get the SID for the specified Group and add it to the SSDL
    $AdminGroup = New-Object Security.Principal.NTAccount $GroupName
    $AdminGroupSid = $AdminGroup.Translate([Security.Principal.SecurityIdentifier]).Value

    $SecurityDescriptor.DiscretionaryAcl.AddAccess([System.Security.AccessControl.AccessControlType]::Allow,
      $AdminGroupSid, 268435456, #full control all operations
      [System.Security.AccessControl.InheritanceFlags]::None,
      [System.Security.AccessControl.PropagationFlags]::None) | Out-Null

    return $SecurityDescriptor.GetSddlForm("All")
 }
 catch [Exception]
  {
  Write-Error -Message "Failed To Generate SSDL (review inner exception):`n $_.Message" -Exception $_.Exception;
 }
}
```

Для настройки параметров сессии используется Startup-скрипт **Startup.ps1**, который мы сохраняем в каталоге **C:\Scripts**:

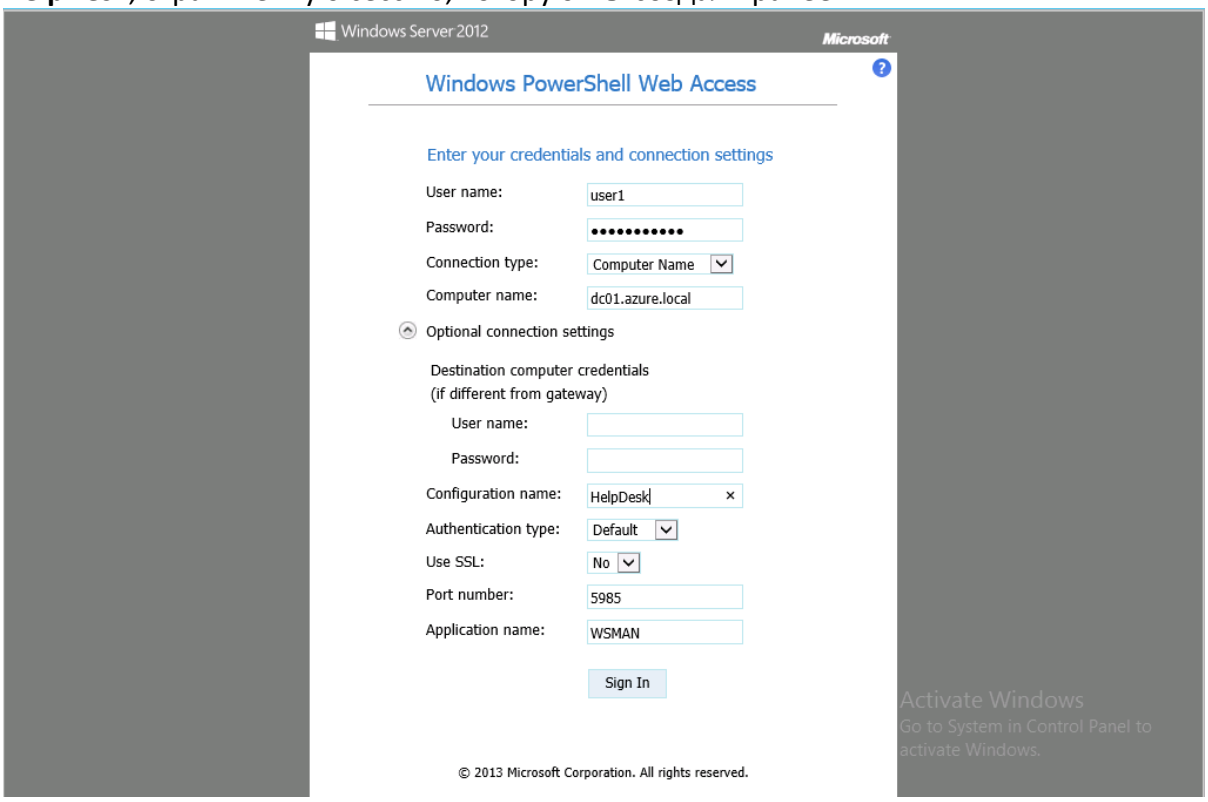
```
Write-Host "Setting environment..." -NoNewline;

# устанавливаем видимость всех команд как “Private”. Некоторые команды сделать приватными невозможно. К ним о
тносятся: Get-Command, Exit-PSSession, Get-FormatData, Get-Help, Measure-Object, Out-Default, Select-Object

foreach ($command in Get-Command)
{
   $command.Visibility = "private"
}

# устанавливаем видимость всех переменных как “Private”
foreach ($variable in Get-Variable)
{
   $variable.Visibility = "private"
}

$ExecutionContext.SessionState.Applications.Clear()
$ExecutionContext.SessionState.Scripts.Clear()
$ExecutionContext.SessionState.LanguageMode = "FullLanguage" #в режиме FullLanguage разрешено пользоваться перем
енными для сохранения результата выполнения командлетов, возможен также режим RestrictedLanguag, в  котором п
ользоваться переменными запрещено

$InitialSessionState = [Management.Automation.Runspaces.InitialSessionState]::CreateRestricted("remoteserver");

foreach ($proxy in $InitialSessionState.Commands | where { $_.Visibility -eq "Public"})
{
 $cmdlet = Get-Command -Type cmdlet -ErrorAction silentlycontinue $proxy.name
 if ($cmdlet)
 {
  $alias = Set-Alias "$($proxy.name)" "$($cmdlet.ModuleName)\$($cmdlet.Name)" -PassThru
  $alias.Visibility = "Private"
 }
 Set-Item "function:global:$($proxy.Name)" $proxy.Definition
}
```

```
# подгружаем модуль, содержащий наши функции — Create—MarketingUser
Import-Module "C:\Scripts\functions.psm1" -DisableNameChecking;

# устанавливаем видимость командлетов, загруженных из модуля functions как public
$allowedCmdlets = dir function:\ | ?{$_.ModuleName -eq "functions"};
$allowedCmdlets | foreach { (Get-Command $_.Name).Visibility = "Public" };

Write-Host " done";
```

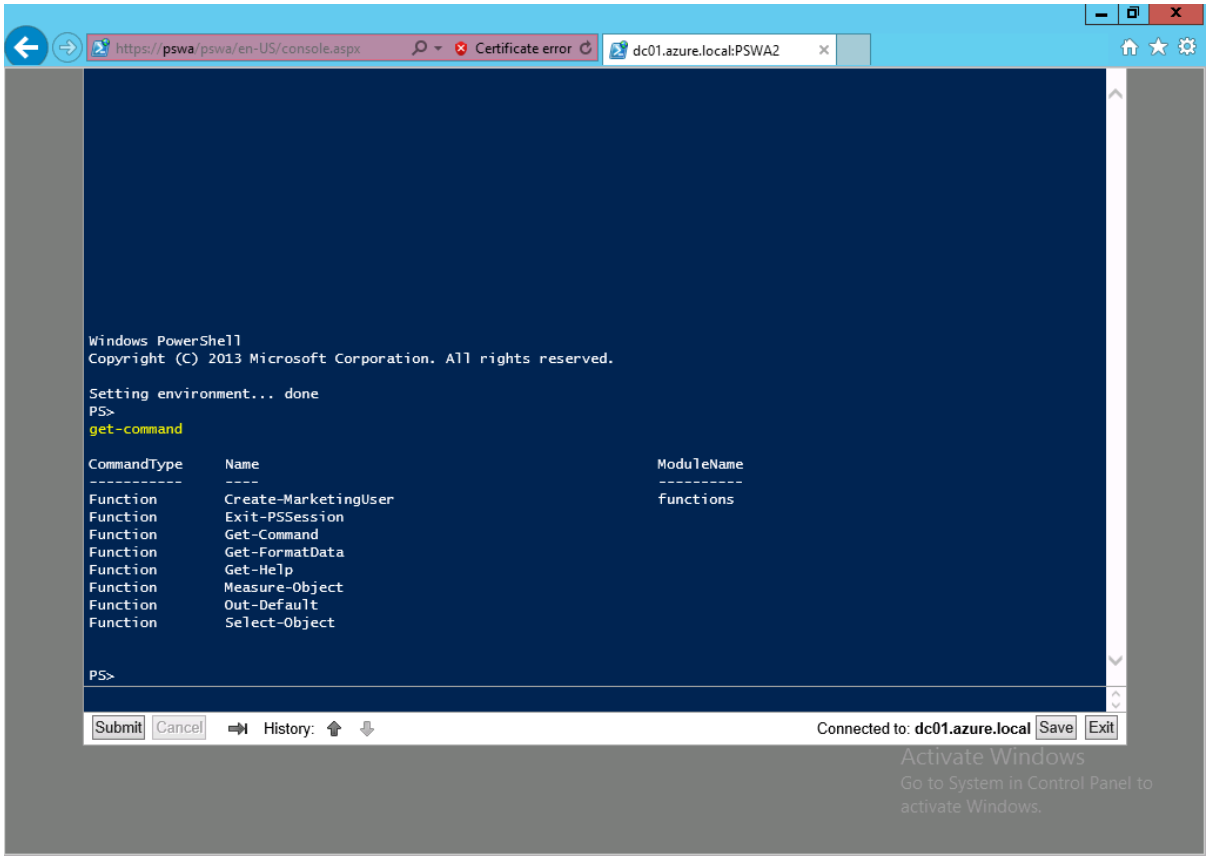
Данный скрипт устанавливает видимость всех командлетов в «Private», тем самым делая их недоступными, подгружает модуль **functions.psm1**, содержащий функции, которые будут доступны в контексте сессии подключившемуся пользователю и будут исполняться с правами учетной записи RunAs, и устанавливает видимость функций из **functions.psm1** в «Public»:

```
function Create-MarketingUser
{
   param([string]$user)

   New-ADUser $user

   Get-ADUser $user
}
```

ВАЖНО!*Необходимо выполнить на каждом сервере, куда будет доступ у группы HelpDesk.*
Для теста я создал пользователя **user1,** входящего в группу **HelpDesk**. Подключаемся к контроллеру домена, указывая в качестве дополнительного параметра **Configuration name: HelpDesk**, ограниченную сессию, которую мы создали ранее.

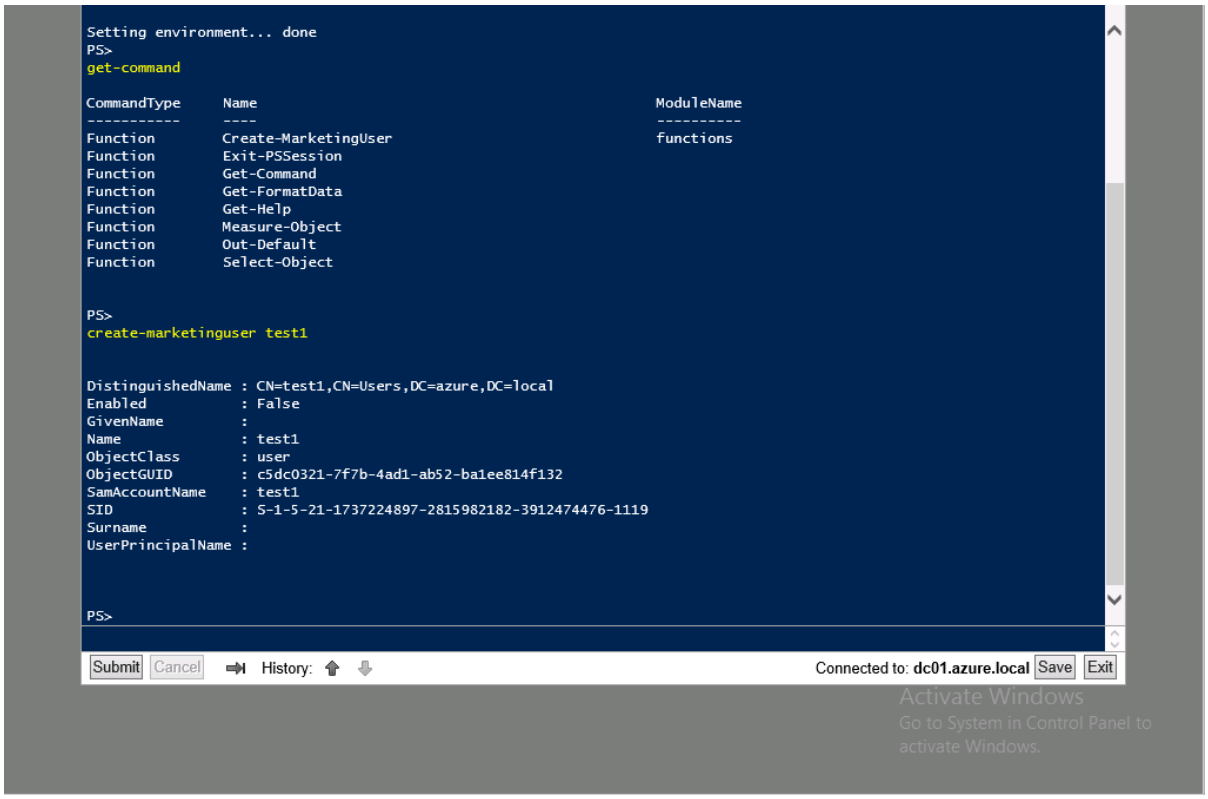


Выполняем **Get-Command**:



Как видно, пользователю недоступно ничего, кроме разрешенного явно — функции, импортированной из модуля **functions.psm1** (хотя сессия исполняется под привилегированной учетной записью).
Теперь попробуем выполнить следующую команду:

```
Create-MarketingUser test1
```



Учетная запись создалась, несмотря на то что подключившийся пользователь не имеет прав на создание учетных записей в AD.

Кастомизация страницы портала PSWA

В данной части я расскажу, как можно изменить первую страницу PowerShell Web Access, а именно: оставить только поля **User name**, **Password**, **Computer name**, и, в зависимости от имени пользователя, подставить необходимое значение **Configuration name:**
HelpDesk для **azure\user1**
SuperHelpDesk для **azure\user2**
Взяв за основу скрипт Startup.ps1, я создал два скрипта для каждой конфигурации: StartupHelpDesk.ps1 и StartupSuperHelpDesk.ps1, изменил строку

```
Write-Host "Setting environment..." -NoNewline;
```

на

```
Write-Host "Setting environment for HelpDesk..." -NoNewline;
```

и

```
Write-Host "Setting environment for SuperHelpDesk..." -NoNewline;
```

соответственно.
Используя приведенный выше код, я создал на контроллере домена еще одну сессию **SuperHelpDesk**, доступ к которой разрешен пользователям, входящим в группу SuperHelpDesk, и создал еще одно правило доступа **Hard Restricted Access**:

```
Add-PswaAuthorizationRule -UserGroupName "Azure\SuperHelpDesk" -ComputerName * -
ConfigurationName "SuperHelpDesk" -RuleName "Hard Resticted access"
```

Для того чтобы отключить панель **Optional connection settings**, необходимо исправить файл **logon.aspx**, расположенный в **C:\Windows\Web\PowerShellWebAccess\wwwroot\en-US**. Данный файл представляет собой веб-форму ASP .NET. Открываем в любом текстовом редакторе и вносим правку:

1. Скрываем список выбора типа подключения. Для этого находим блок:

```
<span class="field">
       <label for="connection-type" id="connectionTypeLabel">Connection type:</label>
       <select class="connectionType" id="connectionTypeSelection" name="connection-
type" clientidmode="Static" runat="server">
         <option id="computerNameOption" value="computer-name" selected="selected">Computer Name</option>
         <option id="connectionUriOption" value="connection-uri">Connection URI</option>
       </select>
      </span>
```

и изменяем первую строку на:

```
<span class="field" style="display: none;">
```

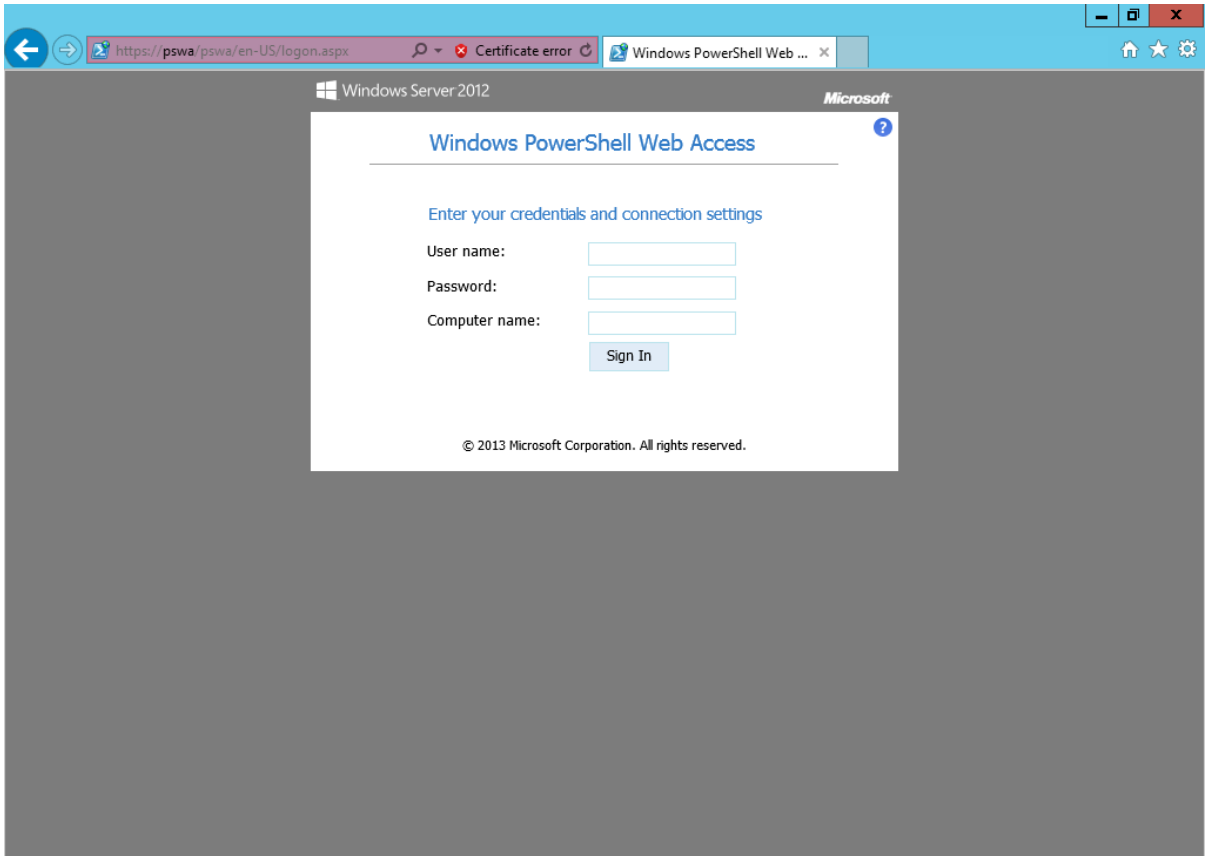
2. Скрываемкнопку Optional connection settings. Для этого находим блок:

```
<div class="show-advanced" >
 <button id="advancedOptionsButton" type="button">
  <img id="arrowIcon" src="../images/down_arrow.png" width="15" height="15" alt="open" />
 </button>
  <label for="advancedOptionsButton" id="advancedOptions">Optional connection settings
  </label>
</div>
```

и исправляем первую строку на

```
<div class="show-advanced" style="display: none;" >
```

ВАЖНО! Перед открытием необходимо сменить владельца файла с **Trusted installer** и изменить права для группы **Administrators**, иначе сохранить файл будет невозможно.
Обновляем страницу и видим, что на ней осталось всего три поля:



Следующий шаг — подставить необходимое значение **Configuration name** в зависимости от имени пользователя. Для этого находим в файле **logon.aspx** функцию, выполняющуюся после нажатия на **Sign in** и проверяющую (изменяющую) значения:

```
S("form").submit(function () {
.....
}
```

В самом начале функции имя пользователя из формы сохраняется в переменную (этим мы воспользуемся позже):

```
var username = S("#userNameTextBox").val().trim();
```

Находим блок кода:

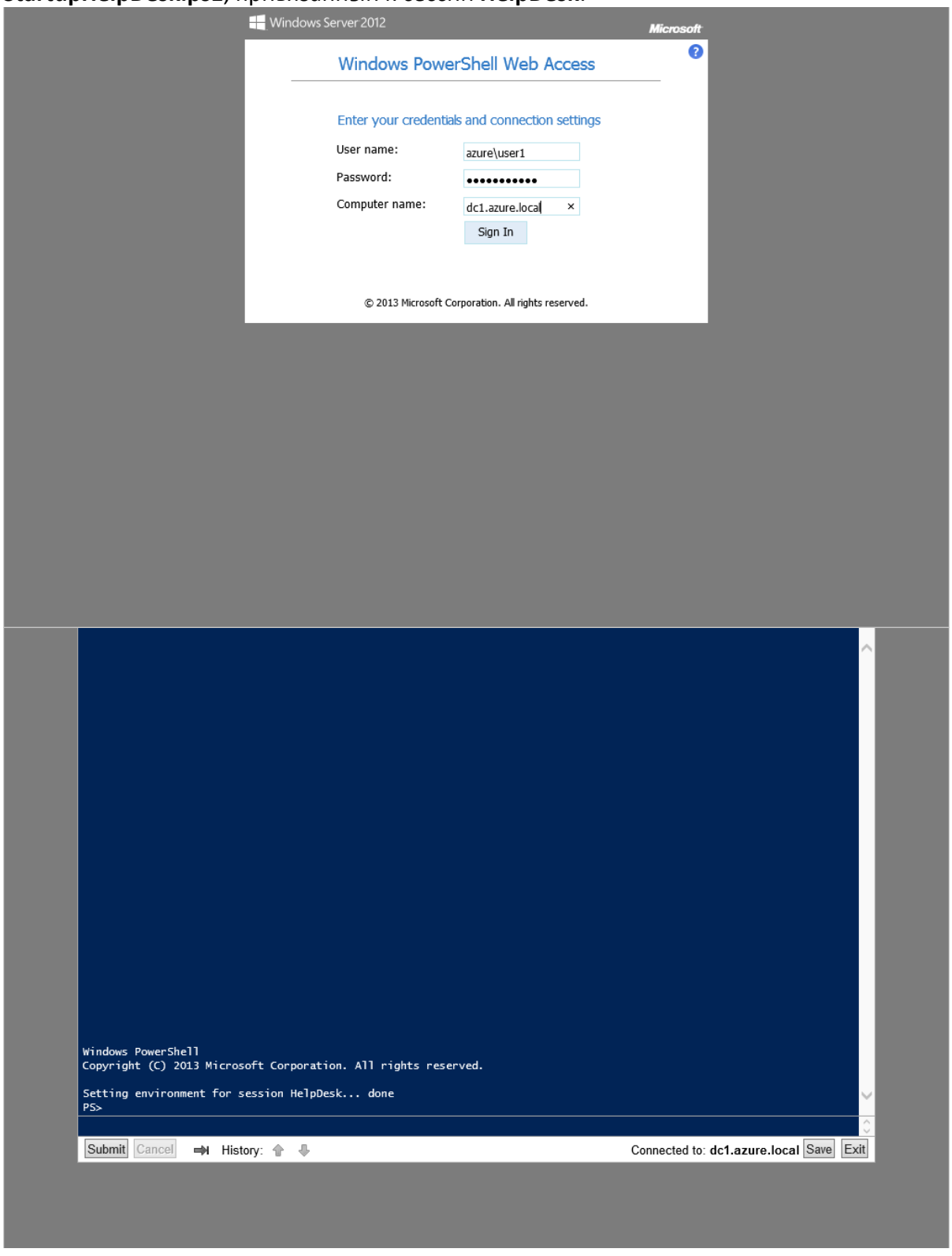
```
if (S("div.error").isVisible()) {
        // Need to clear the passwords when clientside validation fails
        S("#passwordTextBox").val("");
        S("#altPasswordTextBox").val("");
        return false;
      }
```

Так как к моменту исполнения данного кода все проверки выполнены, после этого блока кода можно вставить свой код, а именно подставить необходимое значение **Configuration name** в зависимости от имени пользователя:

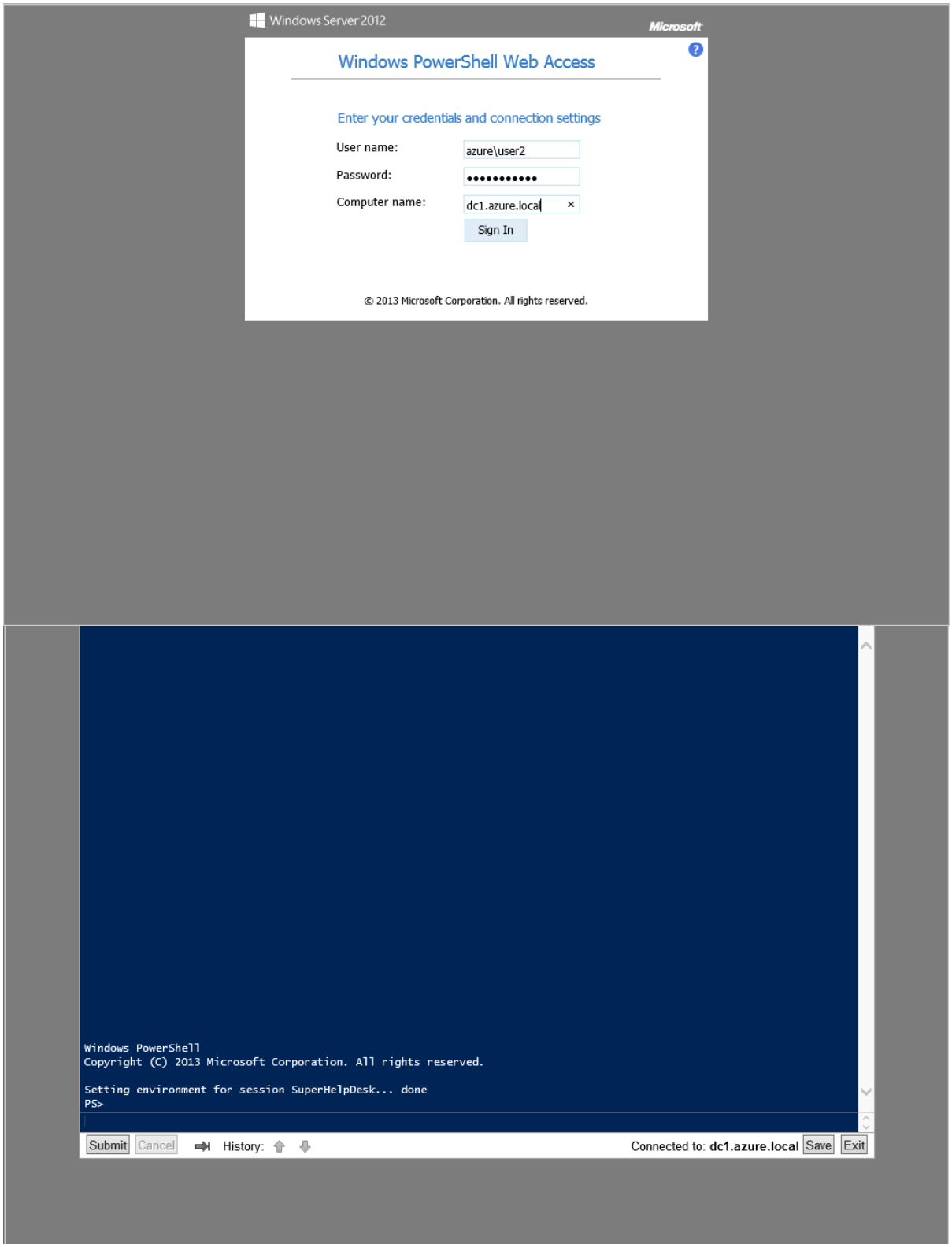
```
switch (username) {
        case "azure\\user1":
          S("#configurationNameTextBox").val("HelpDesk");
          break;

        case "azure\\user2":
          S("#configurationNameTextBox").val("SuperHelpDesk");
          break;
      }
```

Данный кодподставит дляпользователя **azure\user1Configuration name** равным **HelpDesk**, для **azure\user2** — **SuperHelpDesk**. Сохраняем **logon.aspx**, обновляем страницу в браузере и

выполняем вход под пользователем **azure\user1.** Видим, что выполнился скрипт **StartupHelpDesk.ps1**, привязанный к сессии **HelpDesk**:



Выполняем вход под пользователем **azure\user2** и видим, что выполнился скрипт **StartupSuperHelpDesk.ps1**, привязанный к сессии **SuperHelpDesk**:



Таким образом, столь экзотическая для наших широт функция, как PowerShell Web Access, может с успехом использоваться для делегирования административных полномочий, в том числе и в гетерогенных инфраструктурах.

Автор статьи: Сергей Груздов.