第 12 章　学以致用

是时候学以致用了。在本章，我们不会尝试教你任何新东西，而是使用你所学到的知识完成一个完整的示例。所以本示例必须是一个具有实际意义的示例。我们首先设定一个任务，在我们找出如何完成该任务之后，你可以跟随我们的进程。本章是本书内容的一个缩影，因为除了告诉你如何完成工作之外，我们还希望你意识到：你可以自学成才。

12.1　定义任务

首先，我们先假设你正在使用 Windows 10 或 Windows Server 2012 R2，或更新版本，这些版本的操作系统已经安装了 PowerShell v5 或更新版本。我们完成的示例在早期的 Windows 与 PowerShell 版本中或许也可以执行，但我们在 Windows 10 与 PowerShell v5 中已经测试成功。我们知道该示例无法在 Linux 或 macOS 中工作，这是由于这些操作系统没有像 Windows 中那样的用户权限。

我们的目标是使用 PowerShell 在本地系统中修改一些默认的用户 privileges，这并不等同于权限，而是某个用户或组能够执行的一些系统范围的任务。

12.2　发现命令

完成任何任务的第一步都是找出哪一个命令可以完成任务，基于你安装的组件，你得到的结果或许与我们不同，但重要的是解决问题的过程本身。因为我们知道我们希望修改用户权限，因此我们把“privileges”作为关键字。

```
PS C:\> help *privilege*

Name                                Category  Module
```

```
----                              --------  ------
Update-Help                       Cmdlet    Microsoft.PowerShell.Core
ConvertTo-Csv                     Cmdlet    Microsoft.PowerShell.U...
Import-Counter                    Cmdlet    Microsoft.PowerShell.D...
about_Remote_Requirements         HelpFile
about_Remote_Troubleshooting      HelpFile
```

嗯。这并没有帮助。命令中并没有任何信息看起来与 privileges 有关。好的，让我们尝试另一种方式——这次，关注命令本身而不是帮助文件，使用更加宽泛的搜索关键字。

```
PS C:\> get-command -noun *priv*
PS C:\>
```

好的，并没有名称中包含 priv 的命令。让人失望！现在我们不得不查看 PowerShell Gallery 中可能会有什么。我们会意识到该步骤依赖于已经安装的 PowerShell Package Manager。这是 PowerShell v5 的一部分，但也可以在老版本 PowerShell 安装获取。访问 http://powershellgallery.com 获得下载链接。

```
PS C:\> find-module *privilege* | format-table -auto

Version Name          Repository Description
-------               ---- ---------- -----------
0.3.0.0 PoshPrivilege PSGallery Module designed to use allow easi...
```

看上去有戏！让我们安装该模块。

```
PS C:\> install-module poshprivilege

You are installing the module(s) from an untrusted repository. If you
 trust this repository, change its InstallationPolicy value by
running the Set-PSRepository cmdlet.
Are you sure you want to install software from
'https://go.microsoft.com/fwlink/?LinkID=397631&clcid=0x409'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend
[?] Help(default is "N"): y
```

此时你应该小心。虽然 PowerShell Gallery 由微软运营，但微软并不验证其他人发布的代码。因此我们应该停下来，查看我们刚刚安装的代码，在继续往下之前确保代码没有任何问题。该模块的作者是 MVP：Boe Prox，并且我们相信他。

现在让我们查看刚刚获取的代码。

```
PS C:\> get-command -module PoshPrivilege | format-table -auto

CommandType Name              Version Source
----------- ----              ------- ------
Function    Add-Privilege     0.3.0.0 PoshPrivilege
```

```
Function    Disable-Privilege 0.3.0.0 PoshPrivilege
Function    Enable-Privilege  0.3.0.0 PoshPrivilege
Function    Get-Privilege     0.3.0.0 PoshPrivilege
Function    Remove-Privilege  0.3.0.0 PoshPrivileg
```

好的，这看起来就直观很多。我们需要添加或启用权限（privilege），因此我们只需要找出使用哪个 cmdlet。

12.3 学习如何使用命令

幸运的是，Boe 在该模块中包含了帮助文件。那些编写模块但不编写帮助文件的都是坏人。本书的姊妹篇：*Learn PowerShell Toolmaking in A Month of Lunches* (Manning, 2012)，我们阐述了如何编写模块以及如何向模块中加入帮助文件——加入帮助文件是正确的事。让我们来看 Boe 是否做了正确的事。

```
PS C:\> help Add-Privilege

NAME
    Add-Privilege
SYNOPSIS
    Adds a specified privilege for a user or group

SYNTAX
    Add-Privilege [[-AccountName] <String>] [-Privilege]
    <Privileges[]> [-WhatIf] [-Confirm] [<CommonParameters>]
```

他做了正确的事！我们并没有打印出完整的帮助，但我们绝对是完整阅读了帮助。因此看上去我们可以通过-AccountName 参数提供一个用户或组名称，然后我们可以通过名称指定一个或多个权限。事情是，我们并不知道权限有是什么。但模块提供了 Get 命令，让我们试一下。

```
PS C:\> Get-Privilege

Computername         Privilege                        Accounts
------------         ---------                        --------
DESKTOP-GDI89IG      SeAssignPrimaryTokenPrivilege    {NT AUTHORIT...
DESKTOP-GDI89IG      SeAuditPrivilege                 {NT AUTHORIT...
DESKTOP-GDI89IG      SeBackupPrivilege                {BUILTIN\Bac...
DESKTOP-GDI89IG      SeBatchLogonRight                {BUILTIN\Per...
DESKTOP-GDI89IG      SeChangeNotifyPrivilege          {BUILTIN\Bac...
DESKTOP-GDI89IG      SeCreateGlobalPrivilege          {NT AUTHORIT...
DESKTOP-GDI89IG      SeCreatePagefilePrivilege        {BUILTIN\Adm...
```

输出结果有好几屏幕，但的确给我们提供了可用的权限列表，不意外的是，还列出了拥有权限的用户名称。因此让我们尝试添加一个权限。

```
PS C:\> Add-Privilege -AccountName Administrators -Privilege SeDenyBatchLogonRight
```

非常简单。让我们看该命令是否生效。

```
PS C:\> Get-Privilege -Privilege SeDenyBatchLogonRight

Computername          Privilege                    Accounts
------------          ---------                    --------
DESKTOP-GDI89IG       SeDenyBatchLogonRight        {BUILTIN\Adm...
```

能够生效！

现在，我们可以承认本任务并没有完成。但任务本身并不是本章的重点。重点是如何找到解决办法。我们做了什么？

（1）我们搜索包含特定关键字的本地帮助文件。当我们的搜索词在本地帮助文件中并没有任何匹配的命名名称时，PowerShell 的确对所有的帮助文件内容做了一次搜索。这很有用，因为帮助如果提到 `privilege`，我们就已经找到了命令。

（2）我们转到搜索特定的命令名称。这可以帮助找到那些没有安装帮助文件的命令。理想情况是，命令应该总是带有帮助文件，但我们并不生活在一个理想世界，所以我们总是需要额外的步骤。

（3）本地一无所获，我们搜索 PowerShell Gallery，并找到一个看上去可能的模块。我们安装了该模块并查看了该模块的命令。

（4）由于模块作者是一个好人并提供了帮助，我们能够找出如何运行命令获取权限列表。这帮助我们了解命令的属性是如何组织的，以及命令期望的值。我们总是以 `Get` 命令作为开始，如果存在 `Get` 命令，则查看得到的结果是什么样子。

（5）使用我们当前已经收集到的信息，我们就能够实现我们需要的变更。

12.4 自学的一些技巧

再次说明，本书的目的是教会你如何自学——这也是本章希望阐明的。下面是一些技巧。

- 不要害怕使用帮助并确保阅读示例。我们不止一次强调过这一点，但好像没人愿意听。我们仍然会看到很多学生在我们眼皮底下使用 Google 寻找示例。为什么那么害怕帮助文档？如果你都愿意读别人的博客了，为什么不先尝试在帮助文档中阅读示例？
- 请注意，在屏幕上，每一点信息可能都非常重要——请不要跳过不是你目前正在寻找的信息。你很容易这样做，但请不要这么做。要查看每一部分信息，并尝试发现该信息的用处，以及使用该信息能够推算出什么。

- 不要害怕失败，希望你有一台虚拟机，然后在虚拟里实验 PowerShell。学生们经常会问类似这类问题："如果我做了这个和那个，会发生什么？"我们的回答往往是"不知道，自己试试"。在虚拟机做实验是一个好办法，最坏的情况也只不过是将虚拟机回滚到某个快照点，对吧？所以无论做什么，都请试一试。
- 如果尝试一种方法不奏效，不要挠墙——请尝试其他方法。

随着时间的流逝，所有的事情都会变得简单。请耐心并保持练习——但同时在学习过程中不忘思考。

12.5 动手实验

注意：对于本次动手实验来说，你需基于 Windows 8 或 Windows Server 2012，运行 PowerShell v3 或更新版本 PowerShell 的计算机。

下面该轮到你了。我们假设你正在使用虚拟机或其他你可以假借学习的名义搞乱的环境。请不要在生产环境和运行关键系统的计算机上进行实验。

Windows 8 和 Windows Server 2012（或更新版本）包含一个使用文件共享的模块。你的任务是创建一个名称为"LABS"的目录，并共享该目录。为了练习的简便，先假设该目录和共享不存在。先不用管 NTFS 的权限问题，但请确保共享目录的权限设置为"所有人"拥有读/写权限，并且本地管理员拥有完全控制权。由于共享的主要是文件，你获取希望为文档设置共享缓存。你的脚本还应该展示新建的共享及其权限。

12.6 动手实验答案

```
#创建目录
New-item -Path C:\Labs -Type Directory | Out-Null

#创建共享
$myShare = New-SmbShare -Name Labs -Path C:\Labs\ `
-Description "MoL Lab Share" -ChangeAccess Everyone `
-FullAccess Administrators -CachingMode Documents

#获取共享权限
$myShare | Get-SmbShareAccess
```