

Генерация пароля заданной длины и сложности с помощью PowerShell

windowsnotes.ru/powershell-2/generaciya-parolya-zadannoj-dliny-i-slozhnosti-s-pomoshhyu-powershell

18 августа 2024 г.

18.08.2024

Рубрики: PowerShell

Если необходимо сбросить пароли или создать большое количество новых пользователей, это легко можно сделать с помощью Powershell. Сегодня мы рассмотрим 3 способа создания паролей в PowerShell.

- Простой генератор паролей, выдающий произвольный пароль заданной длины;
- Генератор паролей с predetermined набором символов;
- Генератор паролей, позволяющий задать количество прописных, строчных, числовых и специальных символов.

Простой генератор паролей

Самый простой вариант, с использованием встроенного генератора паролей. В этом варианте используем метод `GeneratePassword` класса

`System.Web.Security.Membership`:

```
function Get-Password {  
    param (  
        [Parameter(Mandatory)]  
        [int] $length, #password length  
        [int] $special = 1 #special  
    )  
    Add-Type -AssemblyName 'System.Web';  
    $password = [System.Web.Security.Membership]::GeneratePassword($length,  
        $special);  
    return $password;  
}
```

В итоге получаем произвольный пароль заданной длины. Но методу `GeneratePassword` можно передать только 2 аргумента — длину и минимальное количество нестандартных символов. В ситуации, когда есть жесткие требования к сложности пароля (например парольная политика AD) этот способ может не подойти.

```
PS C:\WINDOWS\system32> Get-Password 8  
&joK}tB$  
  
PS C:\WINDOWS\system32> Get-Password 12 4  
4K:##T=ih1kn  
  
PS C:\WINDOWS\system32> |
```

Генератор паролей с predetermined набором символов

Здесь мы используем класс RandomNumberGenerator, который реализует криптографический генератор случайных чисел. Пароль будем создавать с помощью метода Create:

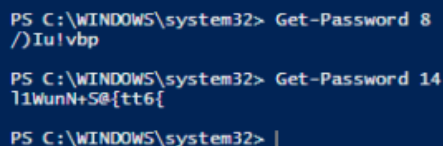
```
function Get-Password {
param (
[Parameter(Mandatory)]
[int] $length #password length
)

#password charset
$charSet =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789{ }+~
[*=@:)}$^%(_!&#?>/|. '.ToCharArray();

#generate password
$rng = [System.Security.Cryptography.RandomNumberGenerator]::Create();
$bytes = New-Object byte[]($length);
$rng.GetBytes($bytes);
$result = New-Object char[]($length);

for ($i = 0 ; $i -lt $length ; $i++) {
$result[$i] = $charSet[$bytes[$i] % $charSet.Length];
}
return (-join $result);
}
```

В этом варианте мы можем контролировать используемый при генерации пароля набор символов, но результат все так же непредсказуем.



```
PS C:\WINDOWS\system32> Get-Password 8
/)Iu!vbp
PS C:\WINDOWS\system32> Get-Password 14
11WunN+S@{tt6{
PS C:\WINDOWS\system32> |
```

Генератор паролей с контролируруемыми параметрами

Используем тот же подход, что и в предыдущем способе, но немного его доработаем, включим в него проверку на соответствие заданным параметрам:

```
function Get-Password {
param (
[Parameter(Mandatory)]
[ValidateRange(4, [int]::MaxValue)]
[int] $length, #length
[int] $upper = 1, #number of uppercase
[int] $lower = 1, #number of lowercase
[int] $numeric = 1, #number of numeric
[int] $special = 1 #number of special
)
```

```

if($upper + $lower + $numeric + $special -gt $length) {
throw "number of upper/lower/numeric/special char must be lower or equal to
length";
}
# password charset
$uCharSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; #uppercase
$lCharSet = "abcdefghijklmnopqrstuvwxyz"; #lowercase
$nCharSet = "0123456789"; #numeric
$sCharSet = "/*-+,!?=()@;:._"; #special
$charSet = "";
if($upper -gt 0) { $charSet += $uCharSet }
if($lower -gt 0) { $charSet += $lCharSet }
if($numeric -gt 0) { $charSet += $nCharSet }
if($special -gt 0) { $charSet += $sCharSet }

#password generate
$charSet = $charSet.ToCharArray();
$rng = [System.Security.Cryptography.RandomNumberGenerator]::Create();
$bytes = New-Object byte[]($length);
$rng.GetBytes($bytes);

$result = New-Object char[]($length);
for ($i = 0 ; $i -lt $length ; $i++) {
$result[$i] = $charSet[$bytes[$i] % $charSet.Length]
}
$password = (-join $result);

# password validation
$valid = $true;
if($upper -gt ($password.ToCharArray() | Where-Object {$_ -cin
$uCharSet.ToCharArray() }).Count) { $valid = $false }
if($lower -gt ($password.ToCharArray() | Where-Object {$_ -cin
$lCharSet.ToCharArray() }).Count) { $valid = $false }
if($numeric -gt ($password.ToCharArray() | Where-Object {$_ -cin
$nCharSet.ToCharArray() }).Count) { $valid = $false }
if($special -gt ($password.ToCharArray() | Where-Object {$_ -cin
$sCharSet.ToCharArray() }).Count) { $valid = $false }

if(!$valid) {
$password = Get-Password $length $upper $lower $numeric $special;
}
return $password;
}

```

Вот теперь мы можем полностью контролировать генерируемый пароль, например исключить из него спецсимволы, или использовать только буквы в нижнем регистре.

```
PS C:\WINDOWS\system32> Get-Password 8
@6sXk4GM

PS C:\WINDOWS\system32> Get-Password 12 2 2 2 0
wAE1gxg4sT0e

PS C:\WINDOWS\system32> Get-Password 18 0 17 0 0
ueshhny1gjamroyho

PS C:\WINDOWS\system32> |
```