3 ways to generate passwords in Powershell

arminreiter.com/2021/07/3-ways-to-generate-passwords-in-powershell

July 31, 2021

Powershell is great for automation and if one wants to reset passwords or create new users, it can be useful to generate the passwords in the Powershell.

The following contains 3 ways how to generate passwords in Powershell:

1. A simple version using the built-in System.Web.Security.Membership
2. Simple random password generator with predefined character set
3. Password Generator that allows to specify the amount of uppercase, lowercase, numeric and special characters

Simple, built-in Password Generator

```
function Get-RandomPassword {

param (

[Parameter(Mandatory)]

[int] $length,

[int] $amountOfNonAlphanumeric = 1

)

Add-Type -AssemblyName 'System.Web'

return [System.Web.Security.Membership]::GeneratePassword($length, $amountOfNonAlphanumeric)

}

Get-RandomPassword 8
```

Generates passwords such as:

```
# Get-RandomPassword 8 0

kz^RawKy

# Get-RandomPassword 8 1

/vmt@pO|

# Get-RandomPassword 8 2

6@=(9Dx[
```

```
# Get-RandomPassword 8 4

fI@W=p${

# Get-RandomPassword 8 7

@+[#$/+M

# Get-RandomPassword 8 8

!&amp;@^*:}*
```

Simple Random Password Generator with predefined character set

```
function Get-RandomPassword {

param (

[Parameter(Mandatory)]

[int] $length

)

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

$charSet =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'.ToCharArray()

$rng = New-Object System.Security.Cryptography.RNGCryptoServiceProvider

$bytes = New-Object byte[]($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)

}

Get-RandomPassword 8
```

lets generate a few passwords via:

```
for($i=0;$i-lt8;$i++) { Get-RandomPassword 8 }

OhVGsUdx

Blg2uK3u

gj9qKDH4

bbsTr25Q

zQ4w2jTK

aMAFMols

26BvUY7E

6GR6fZM5
```

PassGen – Generate passwords with specific requirements

```
function Get-RandomPassword {

param (

[Parameter(Mandatory)]

[ValidateRange(4,[int]::MaxValue)]

[int] $length,

[int] $upper = 1,

[int] $lower = 1,

[int] $numeric = 1,

[int] $special = 1

)

if($upper + $lower + $numeric + $special -gt $length) {

throw "number of upper/lower/numeric/special char must be lower or equal to length"

}

$uCharSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

$lCharSet = "abcdefghijklmnopqrstuvwxyz"

$nCharSet = "0123456789"

$sCharSet = "/*-+,!?=()@;:._"
```

```
$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 }

$charSet = $charSet.ToCharArray()

$rng = New-Object System.Security.Cryptography.RNGCryptoServiceProvider

$bytes = New-Object byte[]($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)

$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-RandomPassword $length $upper $lower $numeric $special

}

return $password

}

Get-RandomPassword 8
```

Generates passwords such as:

length 8, 1 upper, 1 lower, 1 number, 1 special

Get-RandomPassword 8

!64mQbcY

at least 4 upper case characters

Get-RandomPassword 8 4

kEyW8IB/

length 12, at least 2 upper case, 2 lower case, 1 number

Get-RandomPassword 12 2 2 1

USj;,Y5MKyME

length 12, at least 5 upper case, 5 numbers

Get-RandomPassword 12 -upper 5 -numeric 5

722HE32WWDy!

length 8, at least 4 lower case, 4 numbers

Get-RandomPassword 8 0 4 4 0

9ub0v47y

length 8, 2 upper, 2 lower, 2 number, 2 special

Get-RandomPassword 8 2 2 2 2

0=oeS.F4

length 20, 1 upper, 1 lower, 1 number, 1 special

Get-RandomPassword 20 1 1 1 1

ma/dS@Z+ghuvCfEv=_5V