Add/Update User with PowerShell

Discussion

We needed a way to automatically update a local user on a bunch of systems which were not on an Active Directory configuration. We had remote access, and the ability to run PowerShell scripts as an administrator.

It should not be interactive at all.

The following is **insecure** as both the AES256 key and the password encrypted by it are in the script itself, so treat the script with the same security as you would the password itself. It obscures the password instead of actually hiding it; one step above actually putting the password in the file in plain text. It is secure against someone just reading it, but any script kiddie can hack the password in no time.

My initial, simplistic code did not take into account that when you create a secure passsword with no encryption key, it uses a key attached to the machine and user (maybe just the user). The result is that it will work just fine when run by the user on the same machine that generated it, but fails on other machines.

Note: I put a lot of comments in the scripts to explain what is going on. The actual code is very small.

Generate password hash

The first step is to generate a password, and encrypt it with a randomly generated key. This will save the results to two files, aes.key and encrypted_password.txt.

Run this script one time and put the contents of the key and the encrypted passwords files into updateUser.ps1. Each time this is run, a new key is generated.

makepass.ps1

```
# script to create a secure password and save it encrypted
# This will accept a password input from the user, generate a random key,
# and save both the key and the encrypted password to files.
# This was createed to be used with the updateUser.ps1 script
# with the help of Copilot.


# Ensure the script is run with administrative privileges
# Requires -Version 5.1
# Requires -RunAsAdministrator

# Prompt for password
```

```
$password = Read-Host "Enter password" -AsSecureString

# Generate a random key and save it
# Note: a new random key is generated each time this script runs.
# we save teh contents of the key to a file named aes.key
# This key should be kept secure and not shared publicly.
$key = New-Object Byte[] 32
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($key)
Set-Content -Path "aes.key" -Value ($key | ForEach-Object {
$_.ToString() })

# Export the secure string using the key
# The password will be encrypted using AES-256 with the generated key
# Note: The encrypted password will be saved in a file named
encrypted_password.txt
$password | ConvertFrom-SecureString -Key $key | Set-Content
"encrypted_password.txt"
```

Download and Modify script

Download the following Powershell file and edit in your favorite text editor. Paste the output of the previouss code into this script where the script has “Very Long Hex String from above” (keep the quotes around it).

Adjust the following to your needs

- $username: Replace with the username to update/create
- $key: Replace the comma separated integers with the 32 integers in the file aes.key
- $securePassword: Replace *contents of encrypted_password.txt* with the contents of the file encrypted_password.txt
- $fullName: This is the diplay name of the user
- $description: An optional Description of the user (defaults to Added by script)
- $localGroup: Group to add the user so (only one group)

updateUser.ps1

```
# Script to create or update a local user with a secure password
# This script checks if a local user exists, creates it if not, and
sets the password.
# It also ensures the user is part of the Administrators group.
# Requires -Version 5.1
# Requires -RunAsAdministrator

# this is insecure because both the key and the encrypted password are
stored in plaintext within the script.
# and can be reversed to obtain the original password.
# This script is intended to be used with the makepass.ps1 script,
```

```
which generates a secure password and key
# and saves it encrypted.

# For security, consider this password to be obscured, NOT secured.

# Define variables
# you must define $username, $key, and $securePassword variables before
running this script.


# Ensure the username is valid and does not contain special characters
$userName = "Enter Username Here" # Replace with the actual username
you want to create or update

# Replace with your actual key, contents of aes.key file
# The keyfile has 32 bytes, newline separated. Replace newlines with
commas,
# and paste below.
# Example key:
96,255,122,11,73,230,146,112,214,11,216,253,111,225,240,99,181,82,26,48
,245,158,216,219,236,151,62,127,98,155,136,68
# Ensure the key is a byte array of 32 bytes
# Note: The key must be exactly 32 bytes for AES-256 encryption
$key =
[Byte[]](1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
,25,26,27,28,29,30,31,32)

# Enter the contents of the file encrypted_password.txt here, within
quotes.
# This should be the output from ConvertFrom-SecureString using the
same key.
# Example: '76492d1116743f0423413b.....16050a5345AzADgANQBjAA==' (the
dots indicate a truncated string)
# Note: The encrypted key will end with '==' if it is base64 encoded.
$securePassword = 'contents of encrypted_password.txt' | ConvertTo-
SecureString -Key $key

$fullName = "" # Full name for the user, defaults to username if empty
$description = "" # Description for the user, defaults to "User created
by script" if empty
$localGroup = "" # Group to add the user to, defaults to "Users" if
empty. Use Administrators for admin access.

# if $fullName is empty or null
if (-not $fullName) {
    $fullName = $userName
}

# if $description is empty or null
if (-not $description) {
    $description = "User created by script"
```

```
}

# if $localGroup is empty or null
if (-not $localGroup) {
    $localGroup = "Users" # Default group if not specified
}

# Check if user exists, create if not
if (-not (Get-LocalUser -Name $userName -ErrorAction SilentlyContinue))
{
    try {
        New-LocalUser -Name $userName -Password $securePassword -
FullName $fullName -Description $description -ErrorAction Stop
    } catch {
        Write-Error "Failed to create user '$userName': $_"
        exit 1
    }
}

# Set the password (update if user exists)
try {
    Set-LocalUser -Name $userName -Password $securePassword
} catch {
    Write-Error "Failed to set password for user '$userName': $_"
    exit 1
}

# Ensure user is in correct group
if (-not (Get-LocalGroupMember -Group $localGroup -Member $userName -
ErrorAction SilentlyContinue)) {
    try {
        Add-LocalGroupMember -Group $localGroup -Member $userName -
ErrorAction Stop
    } catch {
        Write-Error "Failed to add user '$userName' to group
'$localGroup': $_"
        exit 1
    }
}
# Output success message
Write-Host "User '$userName' has been created or updated successfully
with the specified password." -ForegroundColor Green
```

Run the code

You can run the code by opening PowerShell as Administrator, then copying/pasting directly into the window. This avoids the need to specifically allow power shell script execution. The same code can be used on multiple machines.

Enhancements

- Do not send the script over any public media like e-mail. You can safely send the $key line, or the $securePassword line, but not both.
- Multiple groups could be set up by changing group to an array and then looping through them.
- ~~You can easily change the group to add to. I'd suggest replacing Administrators (two instances near bottom) with a variable, then define the variable at the top.~~
- ~~FullName and Description likewise could be set up in variables if this script will be used multiple times~~

Links

- This was all built on Linux (Devuan) using Microsoft Code and PowerShell
  - https://code.visualstudio.com/docs/setup/linux
  - https://learn.microsoft.com/en-us/powershell/scripting/install/install-debian?view=powershell-7.5#installation-on-debian-11-or-12-via-the-package-repository
- Script was generated with the help of Copilot. While I had a majority of it written beforehand, Copilot became a shortcut to doing the whole AES thing.
  - https://copilot.microsoft.com/
- https://www.danielengberg.com/powershell-script-add-user-to-local-admin-group/
- https://stackoverflow.com/questions/49595003/checking-if-a-local-user-account-group-exists-or-not-with-powershell
- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/convertto-securestring?view=powershell-7.5
- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localuser?view=powershell-5.1
- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/convertto-securestring
- https://community.spiceworks.com/t/use-powershell-securestring-with-windows-system-account/974434
- https://stackoverflow.com/questions/7109958/saving-credentials-for-reuse-by-powershell-and-error-convertto-securestring-ke

Also, thanks to DavidN for tightening it up a little for me.