```
name: RDP

on:
  workflow_dispatch:

jobs:
  secure-rdp:
    runs-on: windows-latest
    timeout-minutes: 3600

    steps:
      - name: Configure Core RDP Settings
        run: |
          # Enable Remote Desktop and disable Network Level Authentication (if needed)
          Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' `
                           -Name "fDenyTSConnections" -Value 0 -Force
          Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp' `
                           -Name "UserAuthentication" -Value 0 -Force
          Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp' `
                           -Name "SecurityLayer" -Value 0 -Force

          # Remove any existing rule with the same name to avoid duplication
          netsh advfirewall firewall delete rule name="RDP-Tailscale"

          # For testing, allow any incoming connection on port 3389
          netsh advfirewall firewall add rule name="RDP-Tailscale" `
            dir=in action=allow protocol=TCP localport=3389

          # (Optional) Restart the Remote Desktop service to ensure changes take effect
          Restart-Service -Name TermService -Force

      - name: Create RDP User with Secure Password
        run: |
          Add-Type -AssemblyName System.Security
          $charSet = @{
              Upper   = [char[]](65..90)      # A-Z
              Lower   = [char[]](97..122)     # a-z
              Number  = [char[]](48..57)      # 0-9
              Special = ([char[]](33..47) + [char[]](58..64) +
                        [char[]](91..96) + [char[]](123..126)) # Special characters
          }
          $rawPassword = @()
```

```
        $rawPassword += $charSet.Upper | Get-Random -Count 4
        $rawPassword += $charSet.Lower | Get-Random -Count 4
        $rawPassword += $charSet.Number | Get-Random -Count 4
        $rawPassword += $charSet.Special | Get-Random -Count 4
        $password = -join ($rawPassword | Sort-Object { Get-Random })
        $securePass = ConvertTo-SecureString $password -AsPlainText -Force
        New-LocalUser -Name "RDP" -Password $securePass -AccountNeverExpires
        Add-LocalGroupMember -Group "Administrators" -Member "RDP"
        Add-LocalGroupMember -Group "Remote Desktop Users" -Member "RDP"

        echo "RDP_CREDS=User: RDP | Password: $password" >> $env:GITHUB_ENV

        if (-not (Get-LocalUser -Name "RDP")) {
          Write-Error "User creation failed"
          exit 1
        }

    - name: Install Tailscale
      run: |
        $tsUrl = "https://pkgs.tailscale.com/stable/tailscale-setup-1.82.0-amd64.msi"
        $installerPath = "$env:TEMP\tailscale.msi"

        Invoke-WebRequest -Uri $tsUrl -OutFile $installerPath
        Start-Process msiexec.exe -ArgumentList "/i", "`"$installerPath`"", "/quiet", "/norestart"
-Wait
        Remove-Item $installerPath -Force

    - name: Establish Tailscale Connection
      run: |
        # Bring up Tailscale with the provided auth key and set a unique hostname
        & "$env:ProgramFiles\Tailscale\tailscale.exe" up --authkey=${{
secrets.TAILSCALE_AUTH_KEY }} --hostname=gh-runner-$env:GITHUB_RUN_ID

        # Wait for Tailscale to assign an IP
        $tsIP = $null
        $retries = 0
        while (-not $tsIP -and $retries -lt 10) {
          $tsIP = & "$env:ProgramFiles\Tailscale\tailscale.exe" ip -4
          Start-Sleep -Seconds 5
          $retries++
        }

        if (-not $tsIP) {
          Write-Error "Tailscale IP not assigned. Exiting."
```

```
        exit 1
    }
    echo "TAILSCALE_IP=$tsIP" >> $env:GITHUB_ENV

- name: Verify RDP Accessibility
  run: |
    Write-Host "Tailscale IP: $env:TAILSCALE_IP"

    # Test connectivity using Test-NetConnection against the Tailscale IP on port 3389
    $testResult = Test-NetConnection -ComputerName $env:TAILSCALE_IP -Port 3389
    if (-not $testResult.TcpTestSucceeded) {
        Write-Error "TCP connection to RDP port 3389 failed"
        exit 1
    }
    Write-Host "TCP connectivity successful!"

- name: Maintain Connection
  run: |
    Write-Host "`n=== RDP ACCESS ==="
    Write-Host "Address: $env:TAILSCALE_IP"
    Write-Host "Username: RDP"
    Write-Host "Password: $(echo $env:RDP_CREDS)"
    Write-Host "==================`n"

    # Keep runner active indefinitely (or until manually cancelled)
    while ($true) {
        Write-Host "[$(Get-Date)] RDP Active - Use Ctrl+C in workflow to terminate"
        Start-Sleep -Seconds 300
    }
```