Runbook: Secure EC2 RDP Access via AWS Secrets Manager in Multi-Account AWS Organization (Windows Only)

Objective

To build a secure, cross-account system in AWS that allows users to remotely log into Windows EC2 instances via RDP, using credentials stored in Secrets Manager in a central "Managerial" account, and accessed from invited member accounts in an AWS Organization. The system will also:

- Use PowerShell automation to select from a list of EC2 secrets
- Dynamically RDP into selected instances
- Apply fine-grained access control via IAM roles, SCPs, resource policies, and network security groups

Architecture Overview

Account A (e.g., K1) and Account B access secrets from a central Managerial Account via STS AssumeRole. Secrets Manager stores EC2 login credentials, and users run a PowerShell script to select and connect to instances over RDP (Windows only).

1. Store Secrets in AWS Secrets Manager (Managerial Account)

Each secret contains:

```
{
 "username": "Administrator",
 "password": "YourSecurePassword123",
 "ipv4": "52.15.123.101",
 "dns": "ec2-52-15-123-101.compute-1.amazonaws.com"
}
```

2. Configure Resource Policy for Each Secret

Attach a resource policy to allow cross-account access from an IAM role in an Org account, e.g.:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowOrgRolesAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::637423537225:role/RDPAccessRole"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "o-ab12cd345e"
        }
      }
    }
  ]
}
```

3. Create an IAM Role in Each Member Account

Role name: RDPAccessRole

Trust Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:user/K1"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Permissions Policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "secretsmanager:GetSecretValue",
   "Resource": "arn:aws:secretsmanager:us-east-1:111111111111:secret:ec2/rdp/*"
  }
 ]
}
```

4. Set SCP (Service Control Policy) to Allow GetSecretValue

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": ["secretsmanager:GetSecretValue"],
   "Resource": "*"
  }
 ]
}
```

5. Configure EC2 Security Groups

Inbound Rule: RDP (TCP 3389) from known IPs or 0.0.0.0/0 for testing.

6. PowerShell Script to Pull Secret + RDP

The following PowerShell script allows the user to select a secret and RDP into the corresponding instance:

Run this command for your IAM account used to configure AWS Cli to assume the role created.

```
aws sts assume-role \
 --role-arn arn:aws:iam::637423537225:role/RDPAccessRole \
 --role-session-name k1-access-session
```

NB: PowerShell should be able to allow scripts to be run

Run this command in an elevated PowerShell mode:

```
Set-ExecutionPolicy Unrestricted
```

This allows allows unsigned script execution with a warning for the scripts downloaded from the internet.

Save the below script to your desired location on your computer. Use the extension ***.ps1***

```
# Step 1: Retrieve all secrets from AWS Secrets Manager
try {
   $secretList = aws secretsmanager list-secrets `
      --query 'SecretList[].Name' `
      --output text

   if (-not $secretList) {
      Write-Error "No secrets found in Secrets Manager."
      exit
   }

   $secretArray = $secretList -split '\s+'
}
catch {
   Write-Error "Failed to list secrets: $_"
   exit
}

# Step 2: Prompt user to select a secret
$SecretName = $secretArray | Out-GridView -Title "Select a Secret to Use" -OutputMode Single

if (-not $SecretName) {
   Write-Warning "No secret selected. Exiting..."
   exit
}

# Step 3: Retrieve the secret value and parse the credentials
try {
   $secretJson = aws secretsmanager get-secret-value `
      --secret-id $SecretName `
      --query SecretString `
      --output text | ConvertFrom-Json
}
catch {
   Write-Error "Failed to retrieve secret value: $_"
   exit
```

```
}

$username = $secretJson.username
$password = $secretJson.password
$RdpHost  = $secretJson.hostname  # Expecting 'hostname' key in secret bundle

if (-not $username -or -not $password -or -not $RdpHost) {
   Write-Error "One or more required values (username, password, hostname) are missing in the selected secret."
   exit
}

# Step 4: Store credentials in Windows Credential Manager
cmdkey /generic:$RdpHost /user:$username /pass:$password

# Step 5: Create RDP file
$rdpFilePath = "$env:TEMP\rdp-$($RdpHost -replace '[^a-zA-Z0-9]', '_').rdp"
$rdpContent = @"
full address:s:$RdpHost
username:s:$username
prompt for credentials:i:0
authentication level:i:2
enablecredsspsupport:i:1
"@

$rdpContent | Out-File -FilePath $rdpFilePath -Encoding ASCII

# Step 6: Launch RDP session
Start-Process "mstsc.exe" "/v:$RdpHost"
```

☑ Tested Environment

- ✔ Windows 10/11
- ✔ AWS CLI configured via AssumeRole
- ✔ EC2 instances running Windows Server
- ✔ PowerShell 5.1+