JEA Security Considerations

07/10/20197 minutes to read

In this article

JEA helps you improve your security posture by reducing the number of permanent administrators on your machines. JEA uses a PowerShell session configuration to create a new entry point for users to manage the system. Users who need elevated, but not unlimited, access to the machine to do administrative tasks can be granted access to the JEA endpoint. Since JEA allows these users to run admin commands without having full admin access, you can then remove those users from highly privileged security groups.

Run-As account

Each JEA endpoint has a designated **run-as** account. This is the account under which the connecting user's actions are executed. This account is configurable in the session configuration file, and the account you choose has a significant bearing on the security of your endpoint.

Virtual accounts are the recommended way of configuring the **run-as** account. Virtual accounts are one-time, temporary local accounts that are created for the connecting user to use during the duration of their JEA session. As soon as their session is terminated, the virtual account is destroyed and can't be used anymore. The connecting user doesn't know the credentials for the virtual account. The virtual account can't be used to access the system via other means like Remote Desktop or an unconstrained PowerShell endpoint.

By default, virtual accounts belong to the local **Administrators** group on the machine. This gives them full rights to manage anything on the system, but no rights to manage resources on the network. When authenticating with other machines, the user context is that of the local computer account, not the virtual account.

Domain controllers are a special case since there isn't a local **Administrators** group. Instead, virtual accounts belong to **Domain Admins** and can manage the directory services on the domain controller. The domain identity is still restricted for use on the domain controller where the JEA session was instantiated. Any network access appears to come from the domain controller computer object instead.

In both cases, you may explicitly define which security groups the virtual account belongs to. This is a good practice when the task can be done without local or domain admin privileges. If you already have a security group defined for your admins, grant the virtual account membership to that group. Virtual account group membership is limited to local security groups on workstation and member servers. On domain controllers, virtual accounts must be members of domain security groups. Once the virtual account has been added to one or more security groups, it no longer belongs to the default groups (local or domain admins).

The following table summarizes the possible configuration options and resulting permissions for virtual accounts:

Computer type	Virtual account group configuration	Local user context

Computer type	Virtual account group configuration	Local user context
Domain controller	Default	Domain user, member of '*DOMAIN*\Domain Admins'
Domain controller	Domain groups A and B	Domain user, member of '*DOMAIN*\A', '*DOMAIN*\B'
Member server or workstation	Default	Local user, member of '*BUILTIN*\Administrators'
Member server or workstation	Local groups C and D	Local user, member of '*COMPUTER*\C' and '*COMPUTER*\D'

When you look at security audit events and application event logs, you see that each JEA user session has a unique virtual account. This unique account helps you track user actions in a JEA endpoint back to the original user who ran the command. Virtual account names follow the format `WinRM Virtual Users\WinRM_VA_<ACCOUNTNUMBER>_<DOMAIN>_<sAMAccountName>` For example, if user **Alice** in domain **Contoso** restarts a service in a JEA endpoint, the username associated with any service control manager events would be `WinRM Virtual Users\WinRM_VA_1_contoso_alice`.

Group-managed service accounts (gMSAs) are useful when a member server needs to have access to network resources in the JEA session. For example, when a JEA endpoint is used to control access to a REST API service hosted on a different machine. It's easy to write functions to invoke the REST APIs, but you need a network identity to authenticate with the API. Using a group-managed service account makes the second hop possible while maintaining control over which computers can use the account. The effective permissions of the gMSA are defined by the security groups (local or domain) to which the gMSA account belongs.

When a JEA endpoint is configured to use a gMSA, the actions of all JEA users appear to come from the same gMSA. The only way to trace actions back to a specific user is to identify the set of commands run in a PowerShell session transcript.

Pass-thru credentials are used when you don't specify a **run-as** account. PowerShell uses the connecting user's credential to run commands on the remote server. This requires you to grant the connecting user direct access to privileged management groups. This configuration is **not**recommended for JEA. If the connecting user already has admin privileges, they can avoid JEA and manage the system via other, unconstrained means. For more information, see the section below on how JEA doesn't protect against admins.

Standard run-as accounts allow you to specify any user account under which the entire PowerShell session runs. Session configurations using fixed **run-as**accounts (with the `-RunAsCredential` parameter) aren't JEA-aware. Role definitions no longer function as expected. Every user authorized to access the endpoint is assigned the same role.

You shouldn't use a **RunAsCredential** on a JEA endpoint because it's difficult to trace actions back to specific users and lacks support for mapping users to roles.

WinRM Endpoint ACL

As with regular PowerShell remoting endpoints, each JEA endpoint has a separate access control list (ACL) that controls who can authenticate with the JEA endpoint. If improperly configured, trusted users may not be able to access the JEA endpoint and untrusted users may have access. The WinRM ACL doesn't affect the mapping of users to JEA roles. Mapping is controlled by the **RoleDefinitions** field in the session configuration file used to register the endpoint.

By default, when a JEA endpoint has multiple role capabilities, the WinRM ACL is configured to allow access to all mapped users. For example, a JEA session configured using the following commands grants full access to `CONTOSO\JEA_Lev1` and `CONTOSO\JEA_Lev2`.

PowerShell

Copy

```powershell
$roles = @{ 'CONTOSO\JEA_Lev1' = 'Lev1Role'; 'CONTOSO\JEA_Lev2' = 'Lev2Role' }
New-PSSessionConfigurationFile -Path '.\jea.pssc' -SessionType RestrictedRemoteServer -RoleDefinitions $roles -RunAsVirtualAccount
Register-PSSessionConfiguration -Path '.\jea.pssc' -Name 'MyJEAEndpoint'
```

You can audit user permissions with the Get-PSSessionConfiguration cmdlet.

PowerShell

Copy

```
Get-PSSessionConfiguration -Name 'MyJEAEnd-
point' | Select-Object Permission
```

Output

Copy

```
Permission
----------
CONTOSO\JEA_Lev1 AccessAllowed
CONTOSO\JEA_Lev2 AccessAllowed
```

To change which users have access, run either `Set-PSSessionConfiguration -Name 'MyJEAEndpoint' -ShowSecurityDescriptorUI` for an interactive prompt or `Set-PSSessionConfiguration -Name 'MyJEAEndpoint' -SecurityDescriptorSddl <SDDL string>` to update the permissions. Users need at least *Invoke* rights to access the JEA endpoint.

It's possible to create a JEA endpoint that doesn't map a defined role to every user that has access. These users can start a JEA session, but only have access to the default cmdlets. You can audit user permissions in a JEA endpoint by running `Get-PSSessionCapability`. For more information, see Auditing and Reporting on JEA.

Least privilege roles

When designing JEA roles, it's important to remember that the virtual and group-managed service accounts running behind the scenes can have unrestricted access to the local machine. JEA role capabilities help limit the commands and applications that can be run in that privileged context. Improperly designed roles can allow dangerous commands that may permit a user to break out of the JEA boundaries or obtain access to sensitive information.

For example, consider the following role capability entry:

PowerShell Copy

```powershell
@{
    VisibleCmdlets = 'Microsoft.PowerShell.-
Management\*-Process'
}
```

This role capability allows users to run any PowerShell cmdlet with the noun **Process** from the**Microsoft.PowerShell.Management** module. Users may need to access cmdlets like `Get-Process` to see what applications are running on the system and `Stop-Process` to kill applications that aren't responding. However, this entry also allows `Start-Process`, which can be used to start up an arbitrary program with full administrator permissions. The program doesn't need to be installed locally on the system. A connected user could start a program from a file share that gives the user local admin privileges, runs malware, and more.

A more secure version of this same role capability would look like:

PowerShell Copy

```powershell
@{
    VisibleCmdlets = 'Microsoft.PowerShell.-
Management\Get-Process', 'Microsoft.Power-
Shell.Management\Stop-Process'
}
```

Avoid using wildcards in role capabilities. Be sure to regularly audit effective user permissions to understand which commands are accessible to the user.

JEA doesn't protect against admins

One of the core principles of JEA is that it allows non-admins to do some admin tasks. JEA doesn't protect against users who already have administrator privileges. Users who belong **Domain Admins**, local **Administrators**, or other highly privileged groups can circumvent JEA's protections via another means. For example, they could sign in with RDP, use remote MMC consoles, or connect to unconstrained PowerShell endpoints. Also, local admins on a system can modify JEA configurations to allow additional users or change a role capability to extend the scope of what a user can do in their JEA session. It's important to evaluate your JEA users' extended permissions to see if there are other ways to gain privileged access to the system.

A common practice is to use JEA for regular day-to-day maintenance and have a just-in-time, privileged access management solution that allows users to temporarily become local admins in emergency situations. This helps ensure users aren't permanent admins on the system, but can get those rights if, and only when, they complete a workflow that documents their use of those permissions.