Auditing and Reporting on JEA

07/10/20194 minutes to read 

In this article

After you've deployed JEA, you need to regularly audit the JEA configuration. Auditing helps you assess that the correct people have access to the JEA endpoint and their assigned roles are still appropriate.

Find registered JEA sessions on a machine

To check which JEA sessions are registered on a machine, use the Get-PSSessionConfiguration cmdlet.

PowerShell Copy

```
# Filter for sessions that are configured as
'RestrictedRemoteServer' to find JEA-like
session configurations
```

```
Get-PSSessionConfiguration | Where-Object {
$_.SessionType -eq 'RestrictedRemoteServer' }
```

Output

Copy

```
Name          : JEAMaintenance
PSVersion     : 5.1
StartupScript :
RunAsUser     :
Permission    : CONTOSO\JEA_DNS_ADMINS Acces-
sAllowed, CONTOSO\JEA_DNS_OPERATORS AccessAl-
lowed,
                CONTOSO\JEA_DNS_AUDITORS
AccessAllowed
```

The effective rights for the endpoint are listed in the **Permission** property. These users have the right to connect to the JEA endpoint. However, the roles and commands they have access to is determined by the **RoleDefinitions** property in the session configuration file that was used to register the endpoint. Expand the **RoleDefinitions** property to evaluate the role mappings in a registered JEA endpoint.

PowerShell

Copy

```
# Get the desired session configuration
$jea = Get-PSSessionConfiguration -Name 'JEA-
Maintenance'

# Enumerate users/groups and which roles they
have access to
$jea.RoleDefinitions.GetEnumerator() | Se-
lect-Object Name, @{
  Name = 'Role Capabilities'
```

```
  Expression = { $_.Value.RoleCapabilities }
}
```

Find available role capabilities on the machine

JEA gets role capabilities from the `.psrc` files stored in the **RoleCapabilities** folder inside a PowerShell module. The following function finds all role capabilities available on a computer.

PowerShell

Copy

```
function Find-LocalRoleCapability {
    $results = @()

    # Find modules with a "RoleCapabilities"
subfolder and add any PSRC files to the re-
sult set
    Get-Module -ListAvailable | ForEach-Ob-
ject {
        $psrcpath = Join-Path -Path $_.Mod-
uleBase -ChildPath 'RoleCapabilities'
        if (Test-Path $psrcpath) {
            $results += Get-ChildItem -Path
$psrcpath -Filter *.psrc
        }
    }

    # Format the results nicely to make it
easier to read
    $results | Select-Object @{ Name =
'Name'; Expression = {
$_.Name.TrimEnd('.psrc') }}, @{
```

```
        Name = 'Path'; Expression = {
$_.FullName }
    } | Sort-Object Name
}
```

Note

The order of results from this function is not necessarily the order in which the role capabilities will be selected if multiple role capabilities share the same name.

Check effective rights for a specific user

The Get-PSSessionCapability cmdlet enumerates all the commands available on a JEA endpoint based on a user's group memberships. The output of `Get-PSSessionCapability` is identical to that of the specified user running `Get-Command -CommandType All` in a JEA session.

PowerShell

Copy

```
Get-PSSessionCapability -ConfigurationName
'JEAMaintenance' -Username 'CONTOSO\Alice'
```

If your users aren't permanent members of groups that would grant them additional JEA rights, this cmdlet may not reflect those extra permissions. This happens when using just-in-time privileged access management systems to allow users to temporarily belong to a security group. Carefully evaluate the mapping of users to roles and capabilities to ensure that users only get the level of access needed to do their jobs successfully.

PowerShell event logs

If you enabled module or script block logging on the system, you can see events in the Windows event logs for each command a user runs in a JEA session. To find these events, open **Microsoft-Windows-PowerShell/Operational** event log and look for events with event ID **4104**.

Each event log entry includes information about the session in which the command was run. For JEA sessions, the event includes information about the **ConnectedUser** and the **RunAsUser**. The**ConnectedUser** is the actual user who created the JEA session. The **RunAsUser** is the account JEA used to execute the command.

Application event logs show changes being made by the **RunAsUser**. So having module and script logging enabled is required to trace a specific command invocation back to the **ConnectedUser**.

Application event logs

Commands run in a JEA session that interact with external applications or services may log events to their own event logs. Unlike PowerShell logs and transcripts, other logging mechanisms don't capture the connected user of the JEA session. Instead, those applications only log the virtual run-as user. To determine who ran the command, you need to consult a session transcript or correlate PowerShell event logs with the time and user shown in the application event log.

The WinRM log can also help you correlate run-as users to the connecting user in an application event log. Event ID **193** in the **Microsoft-Windows-Windows Remote Management/Operational** log records the security identifier (SID) and account name for both the connecting user and run as user for every new JEA session.

Session transcripts

If you configured JEA to create a transcript for each user session, a text copy of every user's actions are stored in the specified folder.

The following command (as an administrator) finds all transcript directories.

PowerShell Copy

```
Get-PSSessionConfiguration |
  Where-Object { $_.TranscriptDirectory -ne
$null } |
    Format-Table Name, TranscriptDirectory
```

Each transcript starts with information about the time the session started, which user connected to the session, and which JEA identity was assigned to them.

Copy

```
**********************
Windows PowerShell transcript start
Start time: 20160710144736
Username: CONTOSO\Alice
RunAs User: WinRM Virtual Users\WinRM VA_1_-
CONTOSO_Alice
Machine: SERVER01 (Microsoft Windows NT
10.0.14393.0)
[...]
```

The body of the transcript contains information about each command the user invoked. The exact syntax of the command used is unavailable in JEA sessions because of the way

commands are transformed for PowerShell remoting. However, you can still determine the effective command that was executed. Below is an example transcript snippet from a user running `Get-Service Dns` in a JEA session:

Copy

```
PS>CommandInvocation(Get-Service): "Get-Ser-
vice"
>> ParameterBinding(Get-Service):
name="Name"; value="Dns"
>> CommandInvocation(Out-Default): "Out-De-
fault"
>> ParameterBinding(Out-Default): name="In-
putObject"; value="Dns"

Running  Dns                  DNS Server
```

A **CommandInvocation** line is written for each command a user runs. **ParameterBindings** record each parameter and value supplied with the command. In the previous example, you can see that the parameter **Name** was supplied the with value **Dns** for the `Get-Service` cmdlet.

The output of each command also triggers a **CommandInvocation**, usually to `Out-Default`. The**InputObject** of `Out-Default` is the PowerShell object returned from the command. The details of that object are printed a few lines below, closely mimicking what the user would have seen.