Edge Store cleanup

Update on the Edge CVE-2026-11645 issue.

Root cause: the 10 affected hzn-itwin11-* hosts only had the default in-box Store version of Edge, not the managed Enterprise install. Because of that, they never had Edge's auto-update service running, so Edge was stuck on an old, vulnerable version (147.0.3912.60) with no way to update itself.

What we've done: pushed the Enterprise Edge MSI to all 10 hosts via Tanium. That installed the full managed browser and, critically, turned on the auto-update service. All 10 hosts are now confirmed on patched versions (150.0.4078.83 and 151.0.4129.59), both above the fix version (149.0.4022.69), and continuing to update on their own since.

What's left: updating the itwin11 golden image itself to include the Enterprise MSI instead of the default Store version. Without this, any new host recomposed from that image will come back with the same gap - no update mechanism - so fixing the image is what prevents this from recurring rather than just resolving it on the current 10 hosts.

Will follow up once the golden image update is complete.

To force the cleanup rather than waiting on it:

```
Get-AppxPackage -AllUsers -Name Microsoft.MicrosoftEdge.Stable | Where-Object { $_.Version -eq '147.0.3912.60' } | Remove-AppxPackage -AllUsers
```

Also remove the provisioned copy so it doesn't reappear for new user profiles on that host:

```
Get-AppxProvisionedPackage -Online | Where-Object { $_.DisplayName -eq 'Microsoft.MicrosoftEdge.Stable' -and $_.Version -eq '147.0.3912.60' } | Remove-AppxProvisionedPackage -Online
```

Here's a combined script that verifies the real (MSI) Edge version is patched, then cleans up any stale AppX Edge registrations below the fix version — rather than hardcoding `147.0.3912.60`, it compares against the fix threshold so it's reusable for future CVEs too.

```
# --- Config ---
$minSafeVersion = [version]"149.0.4022.69"
```

```
# --- Step 1: Verify the real (MSI-installed) Edge is patched ---
$edgeApp = Get-ItemProperty -Path @(
    "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*",
    "HKLM:\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"
) -ErrorAction SilentlyContinue | Where-Object { $_.DisplayName -eq "Microsoft Edge" } |
Select-Object -First 1

if (-not $edgeApp) {
    Write-Output "Microsoft Edge (MSI) not found via registry - cannot verify patch status."
    exit 1
}

$installedVersion = [version]$edgeApp.DisplayVersion
Write-Output "Installed Edge (MSI) version: $installedVersion"

if ($installedVersion -lt $minSafeVersion) {
    Write-Output "Edge MSI version is below the safe threshold ($minSafeVersion). Patch not
applied - exiting without cleanup."
    exit 2
}

Write-Output "Edge MSI is patched (>= $minSafeVersion). Proceeding to clean up stale
Store/AppX registrations."

# --- Step 2: Remove stale AppX Edge Stable packages below the safe version ---
$staleInstalled = Get-AppxPackage -AllUsers -Name "Microsoft.MicrosoftEdge.Stable" -
ErrorAction SilentlyContinue |
    Where-Object { [version]$_.Version -lt $minSafeVersion }

foreach ($pkg in $staleInstalled) {
    Write-Output "Removing stale installed AppX package: $($pkg.Version)"
    try {
        Remove-AppxPackage -Package $pkg.PackageFullName -AllUsers -ErrorAction Stop
        Write-Output "Removed installed package $($pkg.Version) successfully."
    } catch {
        Write-Output "Failed to remove installed package $($pkg.Version): $_"
    }
}

$staleProvisioned = Get-AppxProvisionedPackage -Online -ErrorAction SilentlyContinue |
```

```
    Where-Object { $_.DisplayName -eq "Microsoft.MicrosoftEdge.Stable" -and
[version]$_.Version -lt $minSafeVersion }

foreach ($pkg in $staleProvisioned) {
    Write-Output "Removing stale provisioned AppX package: $($pkg.Version)"
    try {
        Remove-AppxProvisionedPackage -Online -PackageName $pkg.PackageName -ErrorAction Stop
        Write-Output "Removed provisioned package $($pkg.Version) successfully."
    } catch {
        Write-Output "Failed to remove provisioned package $($pkg.Version): $_"
    }
}

if (-not $staleInstalled -and -not $staleProvisioned) {
    Write-Output "No stale AppX Edge Stable registrations found - nothing to clean up."
}

Write-Output "Edge patch verification and AppX cleanup complete."
exit 0
```

A couple of notes on why it's built this way:

- **Step 1 uses the registry Uninstall key directly** rather than the `Installed Applications` Tanium sensor, since the script needs to make its own pass/fail decision at runtime — it can't query Tanium's own sensor data from inside a package action.
- **It only proceeds to cleanup if the real Edge is confirmed patched** — this prevents a scenario where cleanup runs on a host that's actually still vulnerable (e.g., MSI install hasn't landed yet), since removing the AppX entry there wouldn't fix anything and would just erase a signal you might still want.
- Exit codes: `0` = patched and cleaned, `1` = Edge not found at all (investigate), `2` = Edge found but still on an old version (patch hasn't applied yeat — different problem, needs the MSI push, not cleanup).

Tanium package command line:

```
cmd /c powershell.exe -NoProfile -ExecutionPolicy Bypass -File "edge-cleanup-stale-appx.ps1"
```

Revision #4
Created 6 August 2026 04:53:10 by Guest
Updated 6 August 2026 05:38:49 by Guest