Tanium Sensor: MD5 Hash of Installed Applications

This page documents the **Lab - MD5 Hash Installed Applications** Tanium sensor, which enumerates installed applications, identifies their associated executables, and returns the MD5 hash of each binary. This sensor is useful for software inventory, threat hunting, validation of executables, and hash-based security comparisons.

Overview

This Tanium sensor:

- Parses multiple registry uninstall locations for installed applications.
- Extracts each application's executable path using `DisplayIcon`.
- Handles cases where the executable name differs from the application name by locating a likely matching `.exe` in the corresponding directory.
- Computes the **MD5 hash** of each discovered executable.
- Returns results in the format:

Application Name | Executable Full Path | MD5 Hash

Registry Locations Queried

The sensor checks the following uninstall registry keys:

- `HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*`
- `HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*`
- `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*`

These cover both 64-bit and 32-bit applications, as well as per-user installs.

Key Functions

1. Get-InstalledApps

Extracts installed applications with both `DisplayName` and `DisplayIcon` present.

2. Clean-ExecutablePath

Normalizes the executable path by removing quotes, stripping trailing ",0" or similar entries, and verifying the file exists.

3. Get-FallbackExe

When the default executable resolves to an uninstaller:

- The function scans the installation directory.
- Attempts to match an `.exe` whose name closely resembles the application name.
- Helps resolve mismatched or misleading DisplayIcon values.

4. Get-MD5Hash

Generates the MD5 hash of a given executable. Returns `HashError` if the hash cannot be computed.

Script Code

```
##########################################################################################################
########################################
# Tanium Sensor: Lab - MD5 Hash Installed Applications
# Returns: DisplayName | Executable FullPath | MD5 Hash of Executable
# Parses registry Uninstall keys to locate installed applications and their respective
executable.
# Hashes target files and returns Application name, File Path, and MD5 hash.
# In some cases where the executable is named drastically different from the application, it
will typically default to the uninstaller.
```

```
##########################################################################################
########################################

function Get-InstalledApps {
    $registryPaths = @(
        'HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*',
        'HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*',
        'HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*'
    )

    foreach ($path in $registryPaths) {
        Get-ItemProperty -Path $path -ErrorAction SilentlyContinue | Where-Object {
$_.DisplayName -and $_.DisplayIcon } | ForEach-Object {
            [PSCustomObject]@{
                Name = $_.DisplayName
                DisplayIcon = $_.DisplayIcon
            }
        }
    }
}

function Clean-ExecutablePath {
    param ($iconPath)

    if (-not $iconPath) { return $null }

    $exe = $iconPath -replace '\"','' -replace ',\d+$',''
    $exe = $exe.Trim()

    # Check existence and extension
    if (Test-Path $exe -PathType Leaf) {
        if ($exe.ToLower().EndsWith(".exe")) {
            return $exe
        }
    }

    return $null
}
```

```
function Get-FallbackExe {
    param (
        [string]$uninstallerPath,
        [string]$appName
    )

    $dir = Split-Path $uninstallerPath -Parent
    if (-not (Test-Path $dir)) { return $null }

    # Normalize the app name: lowercase, no spaces, alphanumeric only
    $normalizedAppName = ($appName -replace '[^a-zA-Z0-9]', '').ToLower()

    $executables = Get-ChildItem -Path $dir -Filter *.exe -File -ErrorAction Stop
    foreach ($exe in $executables) {
        $normalizedExeName = ($exe.BaseName -replace '[^a-zA-Z0-9]', '').ToLower()
        if ($normalizedExeName -eq $normalizedAppName) {
            return $exe.FullName
        }
    }

    return $null
}

function Get-MD5Hash {
    param ($exePath)
    try {
        $hash = Get-FileHash -Path $exePath -Algorithm MD5 -ErrorAction Stop
        return $hash.Hash
    } catch {
        return "HashError"
    }
}

# Main
$seen = @{}
$apps = Get-InstalledApps

foreach ($app in $apps) {
    $exePath = Clean-ExecutablePath $app.DisplayIcon
```

```
    if ($exePath -and $exePath.ToLower().Contains("unin")) {
        $fallback = Get-FallbackExe -uninstallerPath $exePath -appName $app.Name
        if ($fallback) { $exePath = $fallback }
    }

    if ($exePath -and -not $seen.ContainsKey($exePath)) {
        $seen[$exePath] = $true
        $hash = Get-MD5Hash $exePath
        Write-Output "$($app.Name) | $exePath | $hash"
    }
}
```

Notes

- MD5 is commonly used for compatibility and fast lookups, but not suitable for cryptographic validation.
- Useful in Tanium for:
  - Software compliance
  - Threat/IOC matching
  - Detecting unauthorized file changes
- Sensor avoids duplicate hashes by tracking file paths.

Revision #1
Created 28 November 2025 23:36:06 by Christian Kaba
Updated 28 November 2025 23:44:11 by Christian Kaba