Sign in

redcanaryco / **atomic-red-team** Public

Notifications | Fork 2.8k | Star 9.7k

Code | Issues 6 | Pull requests 4 | Actions | Wiki | Security | Insights

atomic-red-team / atomics / T1006 / **T1006.md**

`56 lines (30 loc) · 2.08 KB`

Preview | Code | Blame | Raw

T1006 - Direct Volume Access

Description from ATT&CK

> Adversaries may directly access a volume to bypass file access controls and file system monitoring. Windows allows programs to have direct access to logical volumes. Programs with direct access may read and write files directly from the drive by analyzing file system data structures. This technique bypasses Windows file access controls as well as file system monitoring tools. (Citation: Hakobyan 2009)
> Utilities, such as NinjaCopy, exist to perform these actions in PowerShell. (Citation: Github PowerSploit Ninjacopy)

Atomic Tests

- Atomic Test #1 - Read volume boot sector via DOS device path (PowerShell)

Atomic Test #1 - Read volume boot sector via DOS device path (PowerShell)

This test uses PowerShell to open a handle on the drive volume via the `\\.\` DOS device path specifier and perform direct access read of the first few bytes of the volume. On success, a hex dump of the first 11 bytes of the volume is displayed.

For a NTFS volume, it should correspond to the following sequence (NTFS partition boot sector):

```
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   EB 52 90 4E 54 46 53 20 20 20 20                 ëR?NTFS
```

Supported Platforms: Windows

auto_generated_guid: 88f6327e-51ec-4bbf-b2e8-3fea534eab8b

Inputs:

Name	Description	Type	Default Value
volume	Drive letter of the volume to access	String	C:

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
$buffer = New-Object byte[] 11
$handle = New-Object IO.FileStream "\\.\#{volume}", 'Open', 'Read', 'ReadWrite'
$handle.Read($buffer, 0, $buffer.Length)
$handle.Close()
Format-Hex -InputObject $buffer
```