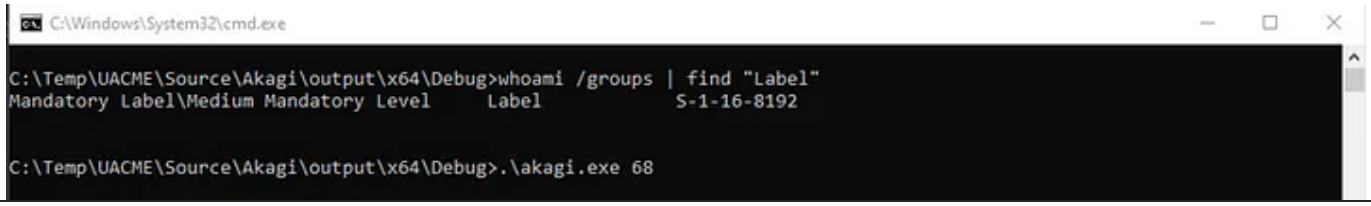
FalconFriday — Detecting UAC Bypasses — 0xFF16

detection rules for several of these UAC bypasses that will allow detection of techniques that are not detected by default using MDE.

TL;DR for red teams: While many functional UAC bypass techniques are available, many of them allow for relatively easy detection. It might be beneficial to research your own techniques instead of relying on widely known techniques that can be easily detected.

The techniques in UACME are simply numbered 1 to 69 and can be invoked by running the “akagi” tool available in the Github repository and providing the technique number as the first command line argument. An additional argument providing the binary to launch can be provided but in our tests we omitted it, causing the default (which is “cmd.exe”) to be launched.



We also investigated whether or not the techniques are detected by Microsoft Defender for Endpoint (MDE) out of the box. This yields 6 techniques that are detected using the 'UAC bypass was detected' alert (techniques 33, 34, 56, 59, 62, and 67), and one technique that is detected using the 'Behavior:Win32/UACBypassExp.F!sdclt' alert (technique 53).

The other 5 functional techniques are not detected by MDE out of the box and require custom detection rules to be detected. Below we go through these techniques, investigate how they work and how they can be detected.

- 43 — {D2E7041B-2927–42fb-8E9F-7CE93B6DC937} — colorui.dll
- 65 — {E9495B87-D950–4AB5–87A5-FF6D70BF3E90} — wscui.cpl

With an excellent tool called OleViewDotNet (released by James Forshaw of Google Project Zero), we can view the properties of COM objects. If we use this tool to view one of the entries above we can see that these COM objects indeed provide elevation using “Auto Approval”.

```
DeviceProcessEvents
| where InitiatingProcessFileName =~ "dllhost.exe"
| where ProcessIntegrityLevel == "High"
| where InitiatingProcessCommandLine has_any ("E9495B87-D950-4AB5-
87A5-FF6D70BF3E90", "3E5FC7F9-9A51-4367-9063-A120244FBEC7",
"D2E7041B-2927-42fb-8E9F-7CE93B6DC937")
```

Another approach would be to create a list of the CLSIDs of COM objects that are known to spawn child processes with high integrity under normal circumstances and look for process creation from other CLSIDs. This would also allow identifying newly discovered Elevated COM interfaces in the

```
DeviceProcessEvents
| where InitiatingProcessFileName =~ "wsreset.exe"
| where ProcessIntegrityLevel == "High"
```

. . .

Technique 61 — ChangePK / SLUI Registry tampering

obviously a lot of ground to cover implementing a good detection capability. Keep an eye out for our next FalconFriday articles where we will dive into other areas.

Falconfriday Uac Defender For Endpoint

 --