🎯 Scenario: UAC Bypass via `fodhelper.exe`

Context:
An alert wasn't triggered, but EDR flagged strange use of `fodhelper.exe`. Your job is to investigate if this was a legitimate process or part of a privilege escalation attempt.

🔍 Step 1: Define Hypothesis (Threat Hunting Phase 1)

Hypothesis: An attacker exploited `fodhelper.exe` to bypass UAC and execute a malicious payload with elevated privileges.

MITRE ATT&CK Mapping:

- T1548.002 – Bypass User Account Control
- T1059 – Command and Scripting Interpreter (PowerShell)

🛠️ Step 2: Environment Setup

You should ideally have:

- Windows VM with **Sysmon** installed
- **Sysmon Config** (from SwiftOnSecurity or Olaf Hartong)
- Logs being ingested by ELK or Splunk (or viewed locally with Event Viewer)

📊 Step 3: Hunt for Execution of `fodhelper.exe`

Sysmon Event ID to Check:

- **Event ID 1** = Process Creation

Sample Query (Splunk):

```
index=sysmon EventCode=1
Image="*fodhelper.exe"
```

In Kibana (ELK):

```
event.code:"1" AND process.name:"fodhelper.exe"
```

Look for:

- Parent process → `explorer.exe`
- CommandLine → blank or oddly structured

- Child processes → PowerShell, `cmd.exe`, `reg.exe`, etc.

🧠 Step 4: Expand to Registry Manipulation

Attackers abuse registry to redirect `fodhelper.exe` to their payload.

Registry Path:

```
HKCU\Software\Classes\ms-settings\shell\open\command
```

Check for:

- A `(Default)` key pointing to a suspicious EXE
- `DelegateExecute` value set to blank

Splunk Query for Registry Changes (Event ID 13):

```
index=sysmon EventCode=13
TargetObject="*ms-settings\\shell\\open\\command"
```

Indicators:

- `cmd.exe`, `powershell.exe`, or custom binary
- Unusual timestamps (off-hours)

🔗 Step 5: Correlate with Other Events

Now pivot:

- What ran immediately after `fodhelper.exe`?
- Any **network activity**? (Sysmon Event ID 3)
- Did it **spawn a reverse shell**?
- Was **lsass.exe** accessed? (Event ID 10 = Process Access)

🧰 Step 6: Use Sigma Rules (Generic Hunting)

Sigma Rule: Bypass UAC via fodhelper

```
title: UAC Bypass Using Fodhelper
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\fodhelper.exe'
```

```
    ParentImage|endswith: '\explorer.exe'
  condition: selection
level: high
```

Convert to:

- **Splunk**
- **Elastic**
- or use tools like Sigmac to translate automatically.

✅ Step 7: Take Action

If confirmed:

- Contain the affected host
- Capture artifacts (memory, persistence, network logs)
- Add detection rules (in SIEM/EDR)
- Map attack to MITRE ATT&CK and report

🧪 Optional Lab: Simulate the Attack Yourself

On your Windows VM:

```
# Create the registry keys for UAC bypass
New-Item "HKCU:\Software\Classes\ms-settings\shell\open\command" -Force
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-
settings\shell\open\command" -Name "DelegateExecute" -Value ""
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-
settings\shell\open\command" -Name "(default)" -Value
"C:\Temp\backdoor.exe"

# Trigger UAC bypass
Start-Process fodhelper.exe
```

Check the logs after this execution to **hunt your own actions**.

📘 Summary

Element	Example
Tactic	Privilege Escalation
Technique	T1548.002 - Bypass UAC via fodhelper
Tools	Sysmon, ELK, Splunk, Sigma
Logs	Event ID 1 (Process), 13 (Registry), 3 (Network)

Element	Example
Key Artifacts	`fodhelper.exe`, registry keys in `ms-settings`, PowerShell
Detection Rule	Sigma + MITRE-aligned custom queries
