.. /Fsi.exe

AWL bypass

64-bit FSharp (F#) Interpreter included with Visual Studio and DotNet Core SDK.

Paths:
C:\Program Files\dotnet\sdk\<version>\FSharp\fsi.exe
C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\Common7\IDE\CommonExtensions\Microsoft\FSharp\fsi.exe

Resources:
- https://twitter.com/NickTyrer/status/904273264385589248
- https://bohops.com/2020/11/02/exploring-the-wdac-microsoft-recommended-block-rules-part-ii-wfc-fsi/

Acknowledgements:
- Nick Tyrer (@NickTyrer)
- Jimmy (@bohops)

Detections:
- Elastic: defense_evasion_unusual_process_network_connection.toml
- Elastic: defense_evasion_network_connection_from_windows_binary.toml
- BlockRule: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules
- IOC: Fsi.exe execution may be suspicious on non-developer machines
- Sigma: proc_creation_win_lolbin_fsharp_interpreters.yml

AWL bypass

1. Execute F# code via script file

```
fsi.exe c:\path\to\test.fsscript
```

Use case: Execute payload with Microsoft signed binary to bypass WDAC policies
Privileges required: User
Operating systems: Windows 10 2004 (likely previous and newer versions as well)
ATT&CK® technique: T1059: Command and Scripting Interpreter

2. Execute F# code via interactive command line

```
fsi.exe
```

Use case: Execute payload with Microsoft signed binary to bypass WDAC policies
Privileges required: User
Operating systems: Windows 10 2004 (likely previous and newer versions as well)
ATT&CK® technique: T1059: Command and Scripting Interpreter