Guide to Named Pipes and Hunting for Cobalt Strike Pipes

By svch0st

Published: 2021-07-25 · Archived: 2026-04-05 16:18:48 UTC

Intro to Named Pipes

The way that helped me start to understand pipes is to think of them as like type of network socket that is created. It can be used to send and receive information between processes or even hosts.

As a rudimentary example, you can query the current pipes on your host:

```
Get-ChildItem \\.\pipe\
```

Now lets try creating one. Below is a basic script to create a named pipe using PowerShell:

```
try {
 $pipeName = "bad_pipe"
 $pipe = New-Object system.IO.Pipes.NamedPipeServerStream($pipeName)
 Write-Host "Listening on \\.\pipe\$pipeName"
 $pipe.WaitForConnection();
 $sr = new-object System.IO.StreamReader($pipe);
 $msg= $sr.ReadLine()
 Write-Host "I received a message: ", $msg
}
catch {
 Write-Host "Pipe Creation Failed..."
 $_
 return 0
}
```

Once running, it will listen for and data sent to it and write it to console. We can quickly test this by redirecting the stdout of a simple command:

```
echo "Sending data to pipe" >\\.\pipe\bad_pipe
```

See it in action here:

How Cobalt Strike uses Named Pipes

There is heaps of existing research on how Cobalt Strike utilises named pipes:

Including from the Cobalt Strike blog:

In this blog, Raphael Mudge (the creator of Cobalt Strike), notes some of the default pipe names. You can also customise the names of these pipes using Malleable C2 profiles.

Get svch0st's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

See a sample of regexes for pipe names I put together from default and custom profiles below:

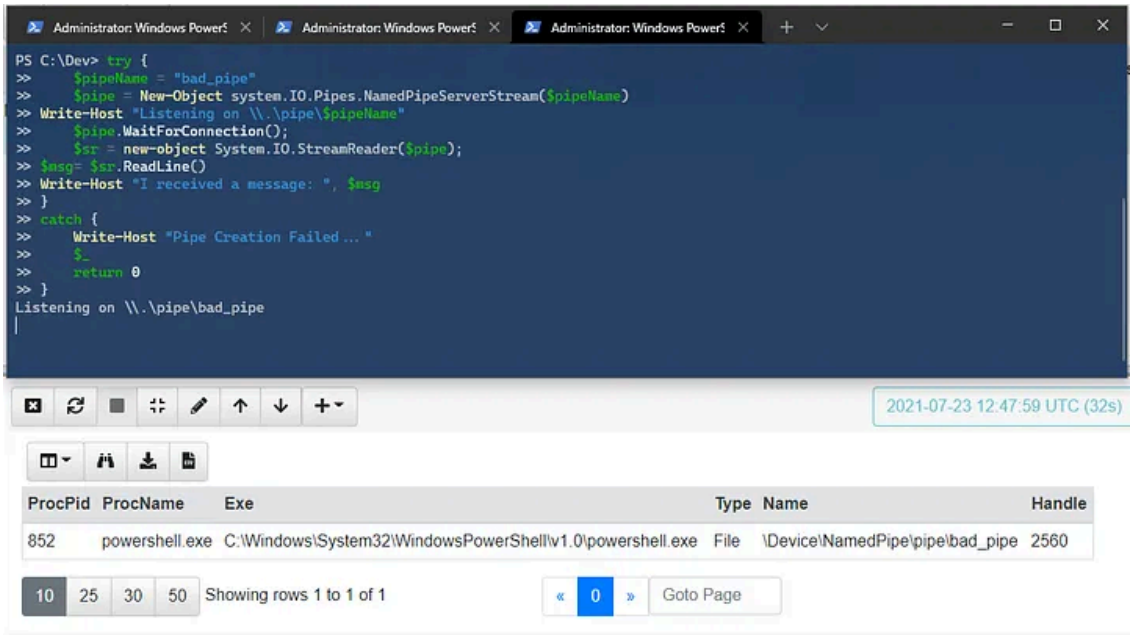
Using Velociraptor to Search for Malicious Named Pipes

When a process uses a named pipe, it creates a handle. Below is a sample of VQL that will walk through all running processes and pull the handles of the process. It will then search for any handles that match the regex `bad_pipe`.

```
LET pipeRegex = 'bad_pipe'
LET processes = SELECT Pid AS ProcPid, Name AS ProcName, Exe
FROM pslist()
WHERE ProcPid > 0SELECT * FROM foreach(
row=processes,
query={
    SELECT ProcPid, ProcName, Exe, Type, Name, Handle
    FROM handles(pid=ProcPid)
    WHERE Name =~ pipeRegex
})
```

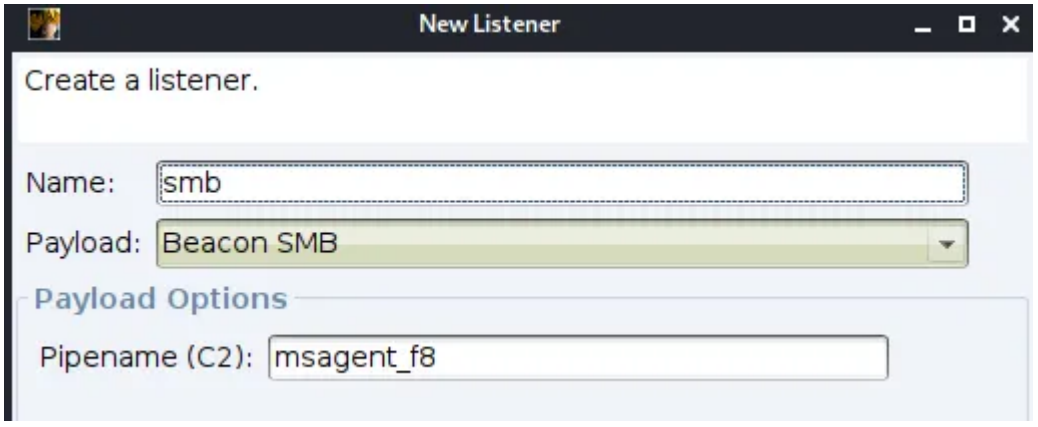
Using the example created before, I left the named pipe open and ran the VQL above in a notebook which returned the following result:

Press enter or click to view image in full size

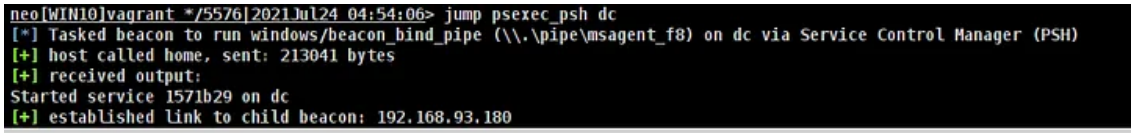


It recorded what process was using the pipe as well as the pipe name! Using the regex of some of the default named pipes lets put all this to the test.

In Cobalt Strike, the interface for creating a new SMB listener the default pipe name was `msagent_f8` which matches what we learnt before. I ran `jump psexec_psh` to laterally move to a different host.



Press enter or click to view image in full size



If we jump into Velociraptor, I created an artefact to search for any handles that match the regex outlined previously. You can see we have the process details as well as the pipe name of the SMB beacon.

This was a good start and found named pipes such as the SMB beacon that stay open for a long period of time, but it doesn't catch the transient named pipes.

Of course, if you are lucky enough to have Sysmon deployed to the network already, you can easily monitor for these same named pipes as shown below:

Edit: I’m currently researching the possibility of monitoring named pipes with ETW and using Velociraptor further.

Thanks,

@svch0st

Source: https://svch0st.medium.com/guide-to-named-pipes-and-hunting-for-cobalt-strike-pipes-dc46b2c5f575