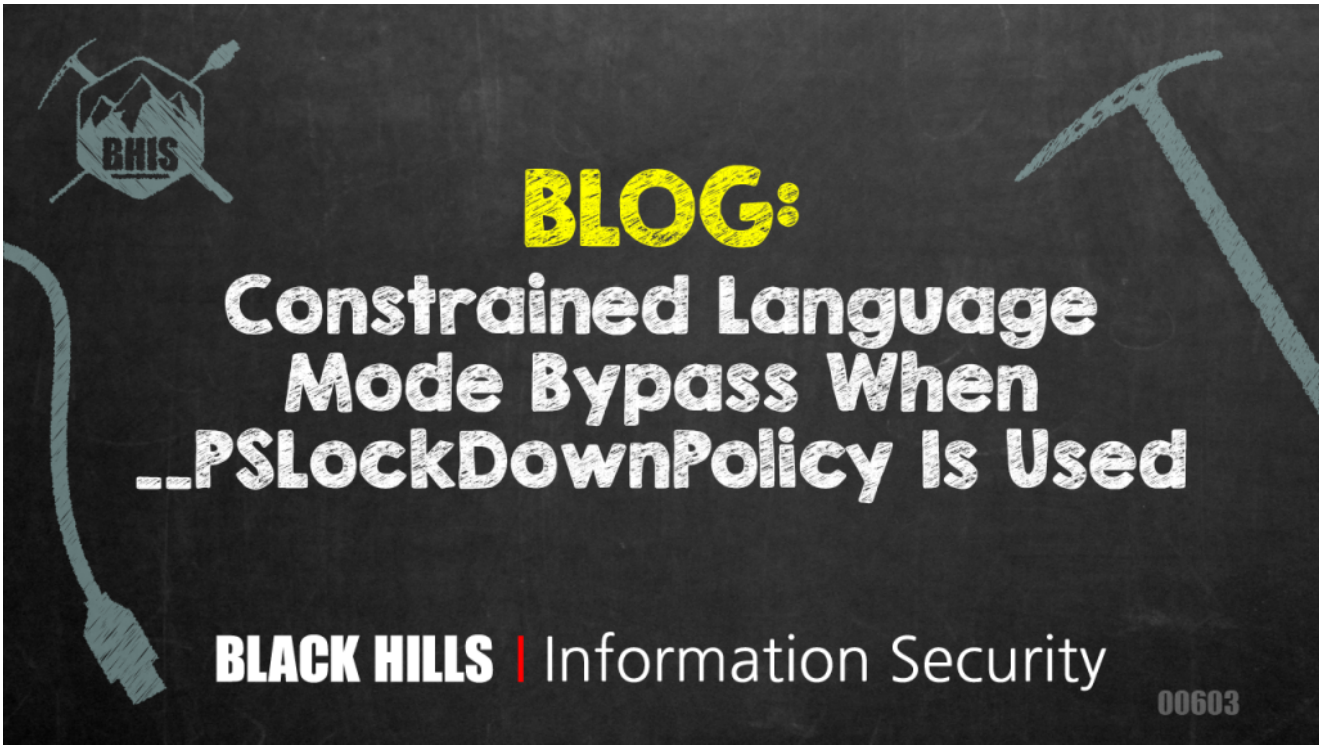
Constrained Language Mode Bypass When __PSLockDownPolicy Is Used

blackhillsinfosec.com/constrained-language-mode-bypass-when-pslockdownpolicy-is-used

27 Sep 2022

Carrie Roberts //


BHIS
BLOG:
Constrained Language Mode Bypass When __PSLockDownPolicy Is Used
BLACK HILLS | Information Security
00603

PowerShell’s Constrained Language (CLM) mode limits the functionality available to users to reduce the attack surface. It is meant to be used in conjunction with application control solutions like Device Guard User Mode Code Integrity. If CLM is enabled without proper application control settings, it is not an effective security solution.

One method for enabling CLM the wrong way is using the __PSLockDownPolicy environment variable. This is what Microsoft has to say about that:

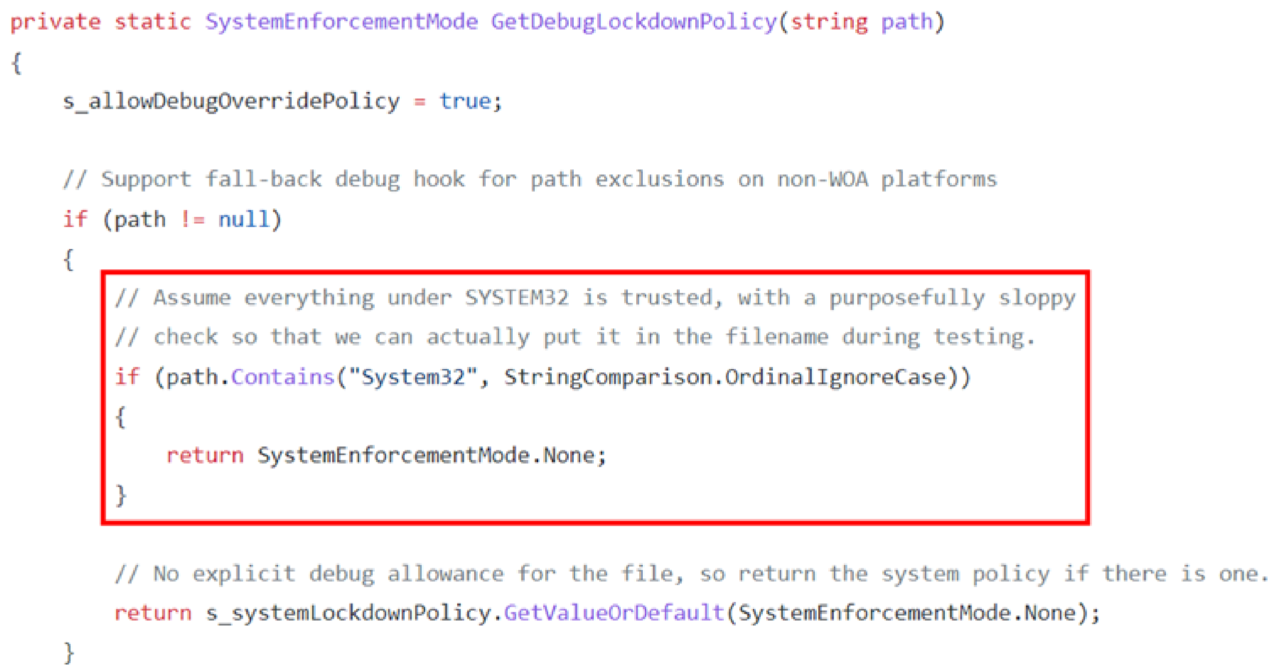
> As part of the implementation of Constrained Language, PowerShell included an environment variable for debugging and unit testing called __PSLockdownPolicy. While we have never documented this, some have discovered it and described this as an enforcement mechanism. This is unwise because an attacker can easily change the environment variable to remove this enforcement. In addition, there are also file naming conventions that enable FullLanguage mode on a script, effectively bypassing Constrained Language.
>
> *reference*

A malicious user with admin privileges could simply remove the environment variable, but what about a user without admins privs? At the end of the quote above, there is a very intriguing statement.

> In addition, there are also file naming conventions that enable FullLanguage mode on a script, effectively bypassing Constrained Language.

There are file naming conventions to enable Full Language mode? Do tell — inquiring minds want to know!

I’m preparing a 16-hour course on “PowerShell For InfoSec” where I will be covering this topic, and I didn’t feel comfortable making such a statement without actually knowing how to do it. So I put Google Search through the paces trying to find the magic file naming convention with no luck. Ultimately, I bit the bullet and actually looked at the PowerShell source code and now I share the magic with you. reference



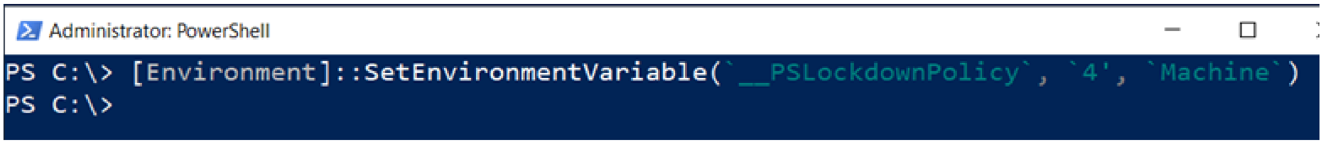
```
private static SystemEnforcementMode GetDebugLockdownPolicy(string path)
{
    s_allowDebugOverridePolicy = true;

    // Support fall-back debug hook for path exclusions on non-WOA platforms
    if (path != null)
    {
        // Assume everything under SYSTEM32 is trusted, with a purposefully sloppy
        // check so that we can actually put it in the filename during testing.
        if (path.Contains("System32", StringComparison.OrdinalIgnoreCase))
        {
            return SystemEnforcementMode.None;
        }

        // No explicit debug allowance for the file, so return the system policy if there is one.
        return s_systemLockdownPolicy.GetValueOrDefault(SystemEnforcementMode.None);
    }
```

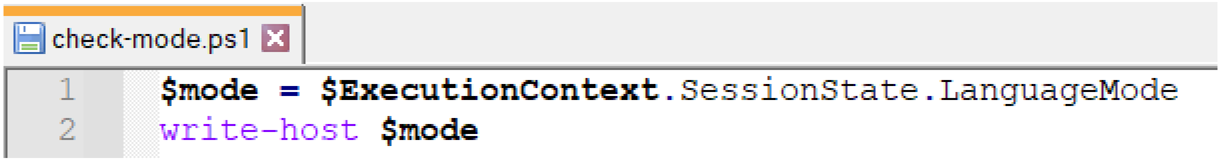
And there we have it. We just need to have “System32” somewhere in the path of the PowerShell script that we want to run in Full Language mode and it will do it. Let’s test it out. First, from an administrative PowerShell prompt, enable CLM using the environment variable (aka “the wrong way).

[Environment]::SetEnvironmentVariable(‘__PSLockdownPolicy‘, ‘4’, ‘Machine‘)



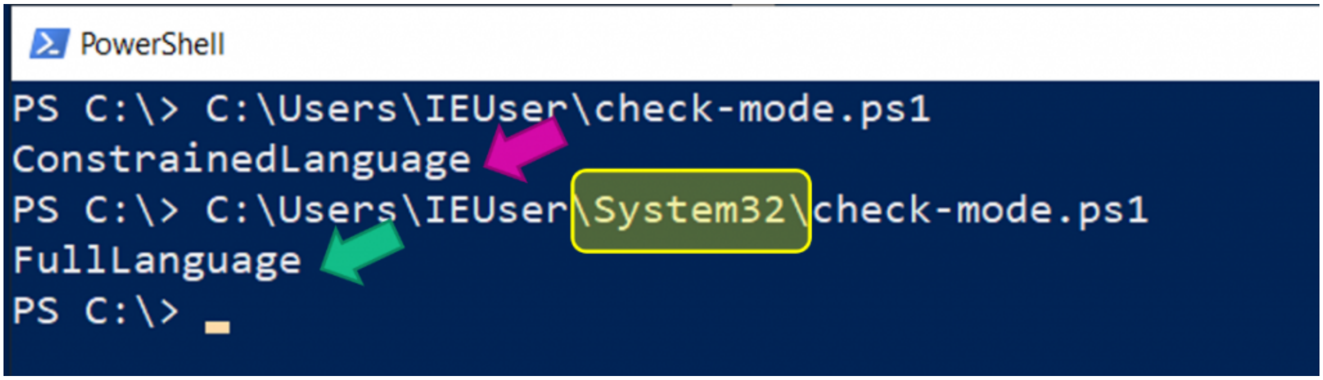
```
PS C:\> [Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '4', 'Machine')
PS C:\>
```

Now, we will use this super simple script just to print out the current language mode.



```
$mode = $ExecutionContext.SessionState.LanguageMode
write-host $mode
```

Let’s run the script first from a path that does not contain “System32” and then again from a path that does.



```
PS C:\> C:\Users\IEUser\check-mode.ps1
ConstrainedLanguage
PS C:\> C:\Users\IEUser\System32\check-mode.ps1
FullLanguage
PS C:\> _
```

And there you have it; we can easily run any script in full language mode in this case, even without administrative access.

Keep this trick in mind the next time you run into CLM on a pentest to help ensure the organization has implemented it correctly.