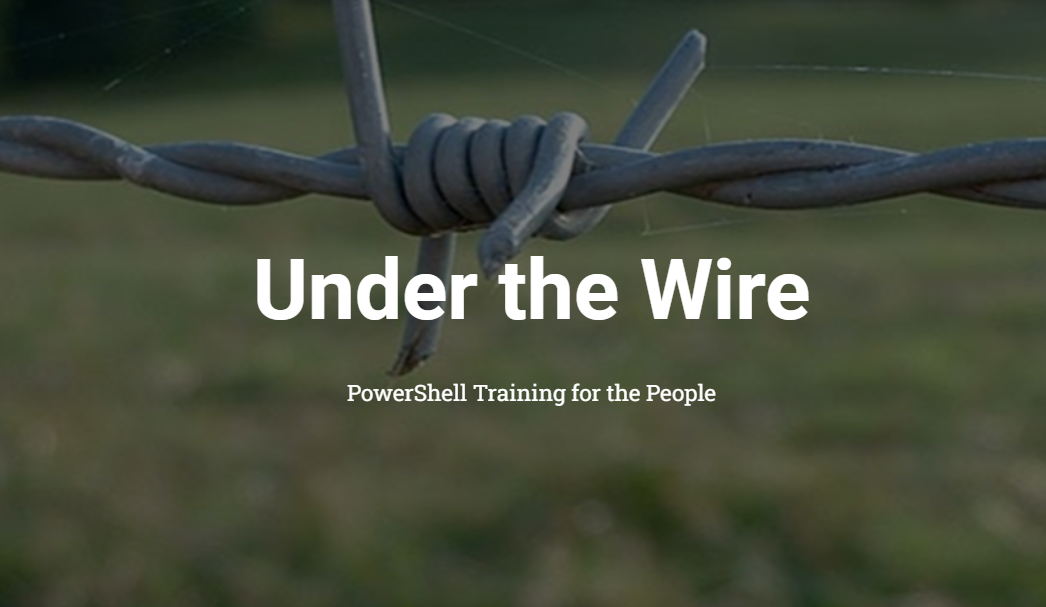
**October 16th’**

**(IS 401/501-01 MERGED) (FA23) CYBERSECURITY PRINCIPLES**

**Powershell / UnderTheWire - Century**

**Pod 6**

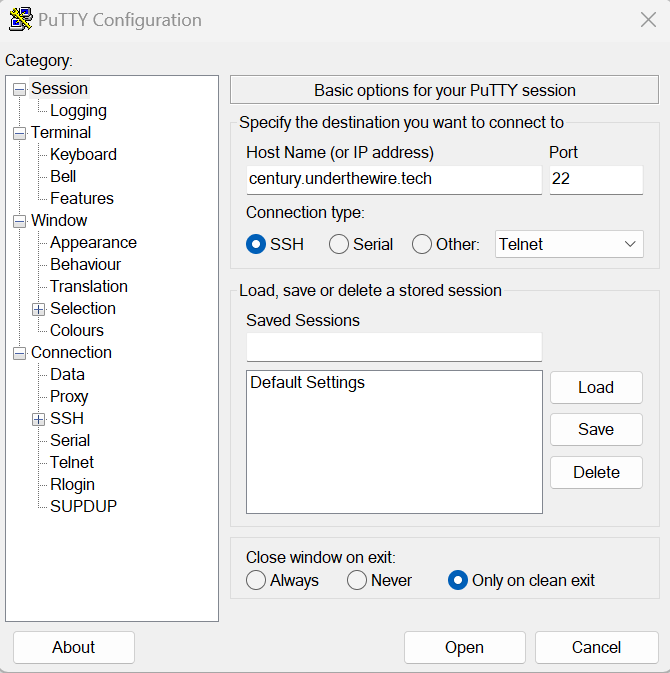
**Aayushi Bhatia, Frances Brandofino, Elsa Gilsdorf, Amogh Patel**



**Introduction:**

The host that you will be connecting to is **century.underthewire.tech**, on port **22**.

Since the provided slank link is no longer active we can use putty, an open source software which is a SSH and a telnet client.

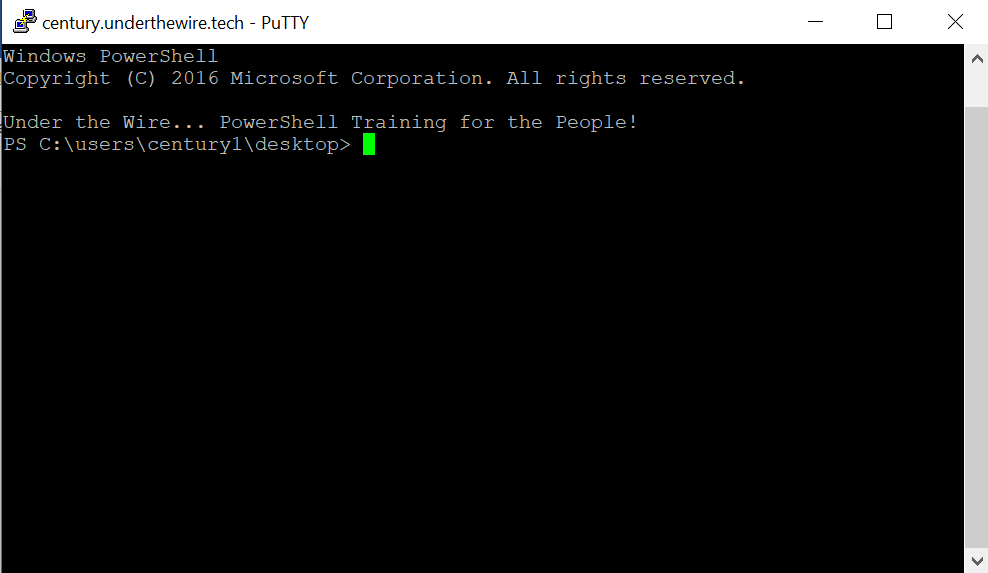


Note: (After every century level, close and reopen putty to start the next level)

[**Century 0 -> 1**](https://underthewire.tech/century)

**Username:** century1

**Password:** century1



[**Century 1 -> 2**](https://underthewire.tech/century-1)

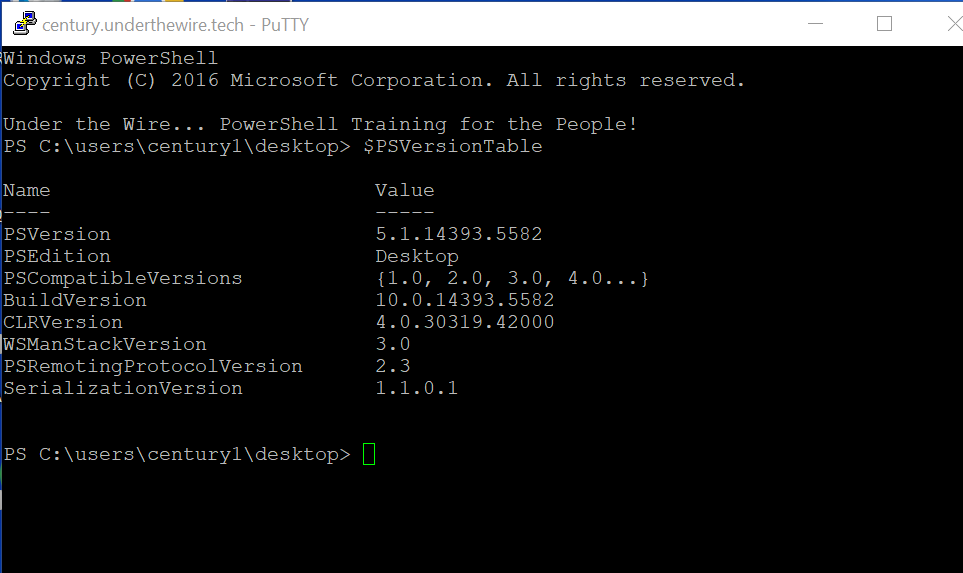
*The password for Century2 is the build version of the instance of PowerShell installed on this system.*

**Username:** century2

**Password:** 10.0.14393.5582

The password for this level is the Build Version.

Build Version can be found out using the command: **$PSVersionTable**





[**Century 2 -> 3**](https://underthewire.tech/century-2)

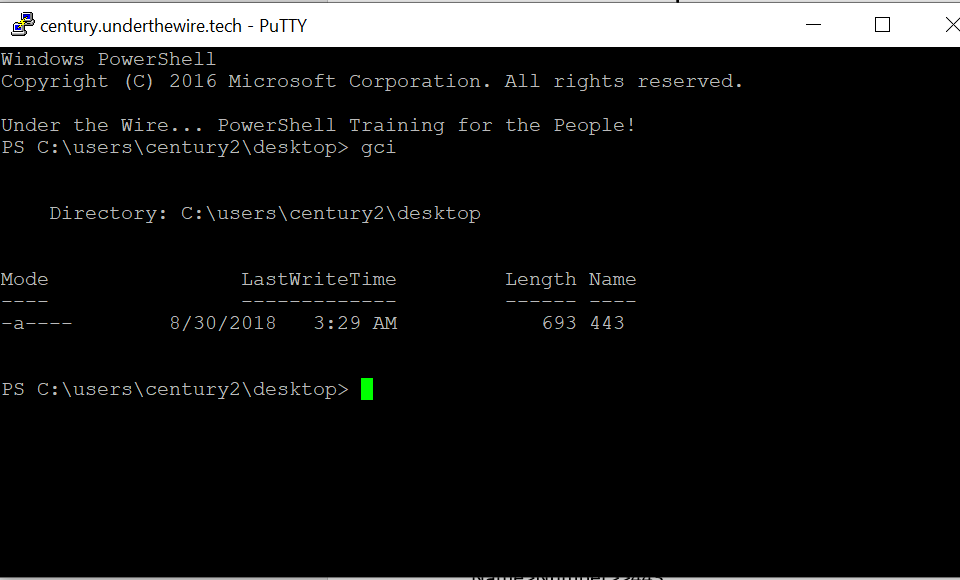
*The password for Century3 is the name of the built-in cmdlet that performs the wget like function within PowerShell PLUS the name of the file on the desktop.*

**Username:** century3

**Password:** Invoke-WebRequest443

The name of the file on the desktop is found by the following command - ‘**get-childitem**’ , ‘**gci**’. The objects are obtained in one or more specified locations using the Get-ChildItem cmdlet. If the item is a container, it retrieves the things, referred to as child items, that are contained within the container.

The ‘**wget**’ equivalent for powershell is ‘Invoke-WebRequest’. Therefore, the password is ‘Invoke-WebRequest443’





[**Century 3 -> 4**](https://underthewire.tech/century-3)

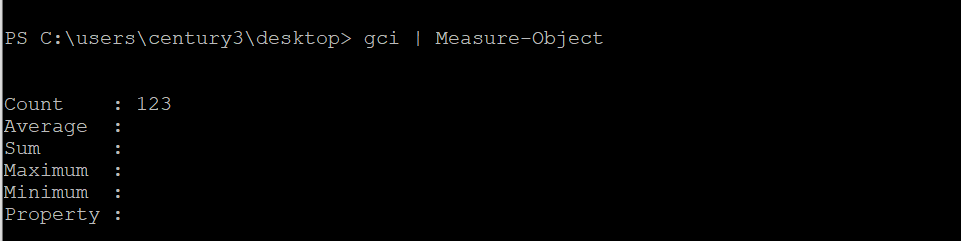
*The password for Century4 is the number of files on the desktop.*

**Username:** century4

**Password:** 123

‘**gci**’ retrieves the files from the specified location (in this case, desktop)

‘**Measure-Object**’ command counts the files and folders in the current directory.





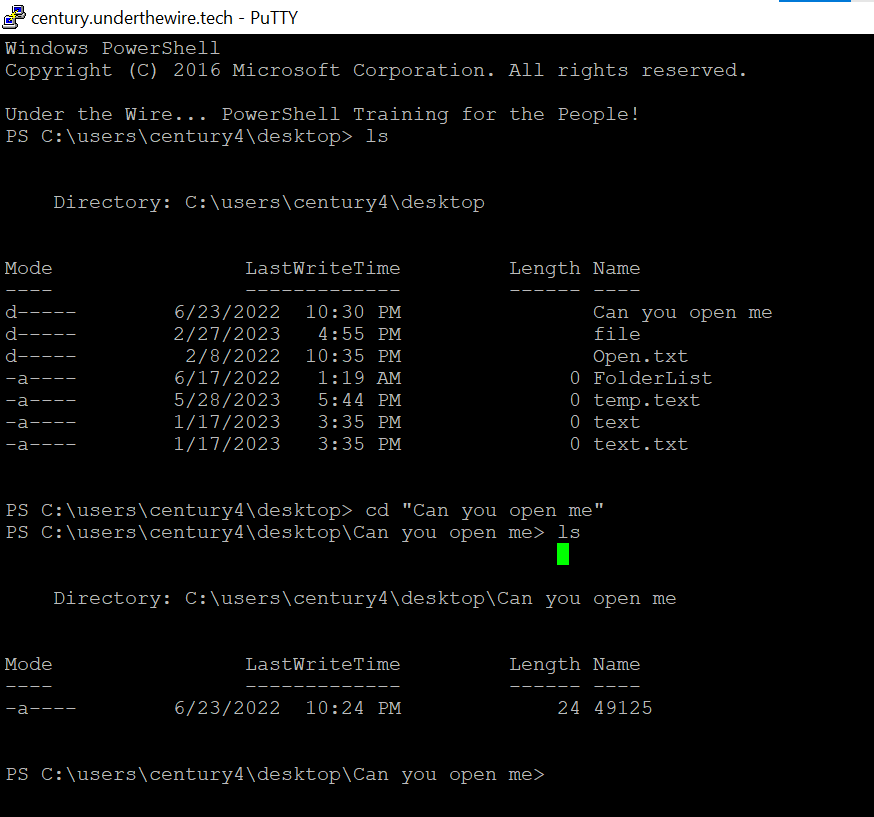
[**Century 4 -> 5**](https://underthewire.tech/century-4)

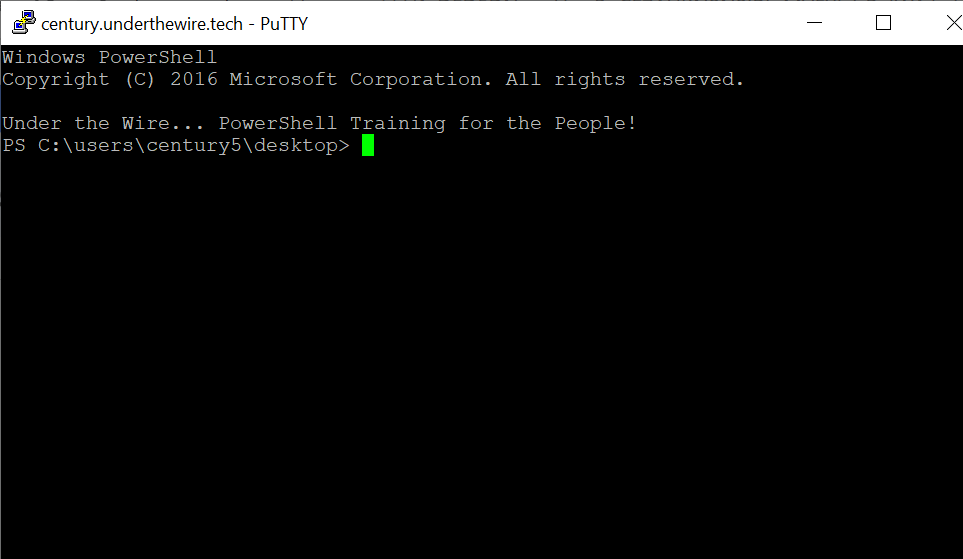
*The password for Century5 is the name of the file within a directory on the desktop that has spaces in its name.*

**Username:** century5

**Password:** 49125

The ‘**ls**’ command lists all the folders/files that are present on the desktop. The directory with spaces in its name is ‘Can you open me’. We change the directory from desktop to ‘Can you open me’ by using the command ‘**cd**’. In the command line we put the directory name in quotes to factor in the space in the name. We use the ‘**ls**’ command to list the files in the new directory and the filename is the password for century 5.





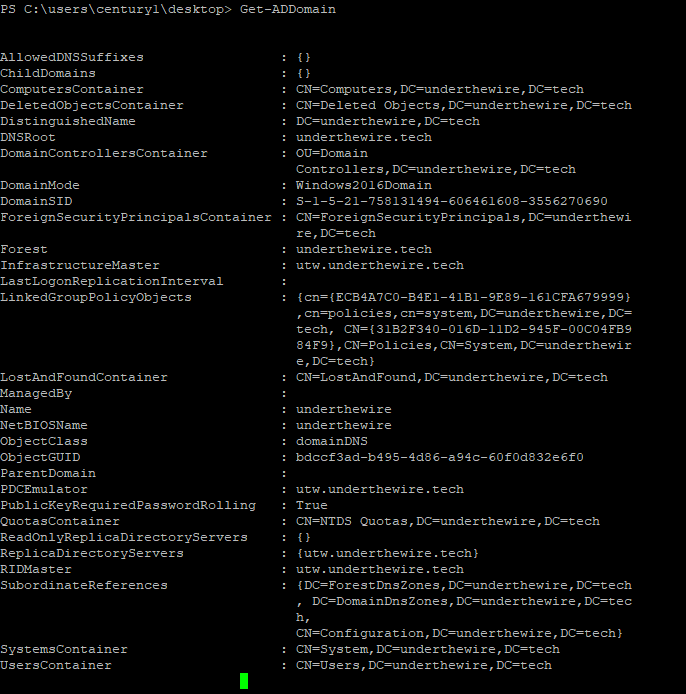
[**Century 5 -> 6**](https://underthewire.tech/century-5)

*The password for Century6 is the short name of the domain in which this system resides in* ***PLUS*** *the name of the file on the desktop.*

**Username:** century6

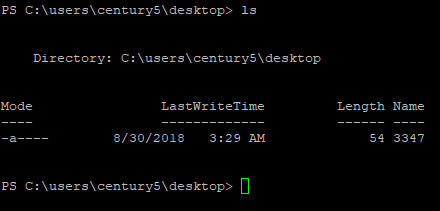
**Password:** underthewire3347

The ‘**Get-ADDomain**’ command outputs the Active Directory Domain.



The name for this domain is ‘underthewire’.

Then, using the command ‘**ls**’, we can find the file in the desktop is named ‘3347’.



Thus, the password is ‘underthewire3347’.

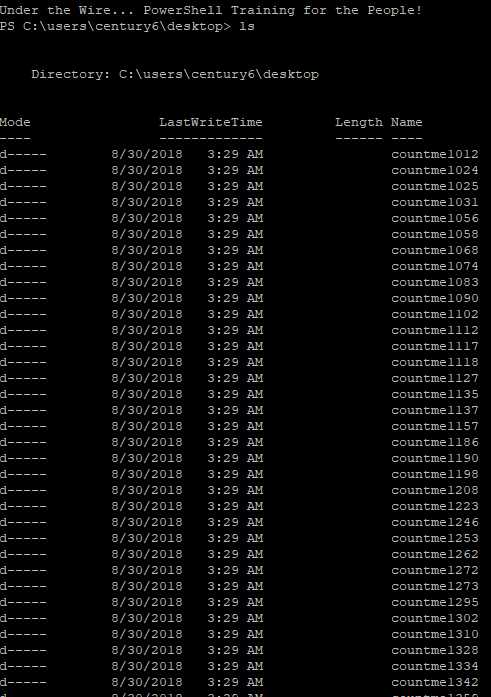
[**Century 6 -> 7**](https://underthewire.tech/century-6)

*The password for Century7 is the number of folders on the desktop.*

**Username:** century7

**Password:** 197

Once again, we use ‘**ls**’ to look at the files on the desktop.



However, it would be extremely inefficient to count each and every folder. Instead, we use the command, ‘**(Get-ChildItem | Measure-Object).Count**’. Get-ChildItem will grab whatever is in the desktop, then Measure-Object will check whatever Get-ChildItem finds to see if it can be counted, thus the .Count addition.



Therefore, the solution is 197.

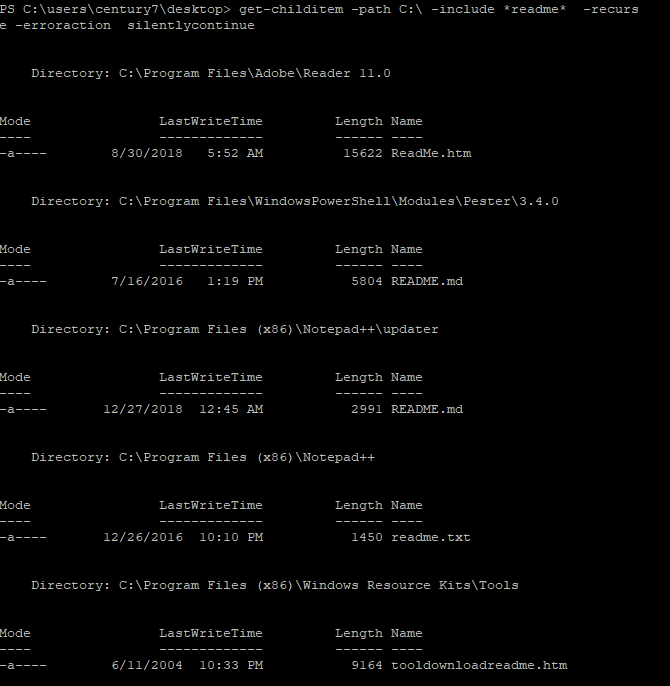
[**Century 7 -> 8**](https://underthewire.tech/century-7)

*The password for Century8 is in a readme file somewhere within the contacts, desktop, documents, downloads, favorites, music, or videos folder in the user’s profile.*

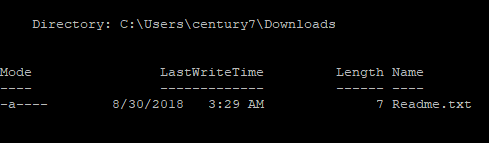
**Username:** century8

**Password:** 7points

To find the file, the command ‘**get-childitem -path C:\ -include \*readme\* -recurse -erroraction silently continue**’ must be called. See Get-childitem explanation above in “**Century 2 -> 3”**. **-path** will specify which path get-childitem can access, in this case being C:\. -include with specify to show only files with a certain name included, with the name readme (specified in \*). **-recurse** will ensure that get-childitem will not just check inside the first folder it accesses, but all folders inside that folder as well. Finally, **-erroraction** specifies what the system must do in case there is a permissions issue, where in this case silentlycontinue means to ignore the error and not display it.



The question specifies that this readme file is inside the user’s profile, the list that pops up must have a directory inside the century7 user folder. The only file that follows this guideline is as follows:



Now, to access this folder use the command ‘**set-location -path C:\users\century7\downloads**’.



Then, use ‘**get-content readme.txt**’.



The answer is 7points.

[**Century 8 -> 9**](https://underthewire.tech/century-8)

*The password for Century9 is the number of unique entries within the file on the desktop.*

**Username:** century9

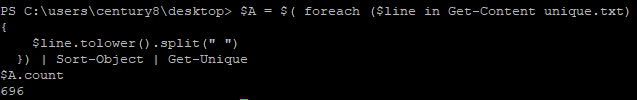
**Password:** 696

Using the command ‘**$A = $( foreach ($line in Get-Content C:\Test1\File1.txt) {**

**$line.tolower().split(" ")**

**}) | Sort-Object | Get-Unique**

**$A.count**’, we can find how many unique entries there are in the file, unique.txt. To explain, the command $A, is a variable, where it equals every line that get-content collects from unique.txt that is unique, thanks to | sort-object | get-unique. Then, the resulting list is counted with **$A.count**.



Thus, the password is 696.

[**Century 9 -> 10**](https://underthewire.tech/century-9)

*The password for Century10 is the* ***161st*** *word within the file on the desktop.*

**Username:** century10

**Password:** pierid

To complete this puzzle, the command ‘**((get-content Word\_File.txt).split(‘ ‘))[160]**’ must be used. See Get-content explanation in **Century 2 ->3**. Then the command **.split** will split each line by any spaces in between each word. After it has done this 160 times, it outputs the word needed for the puzzle.



Thus, the word is pierid.

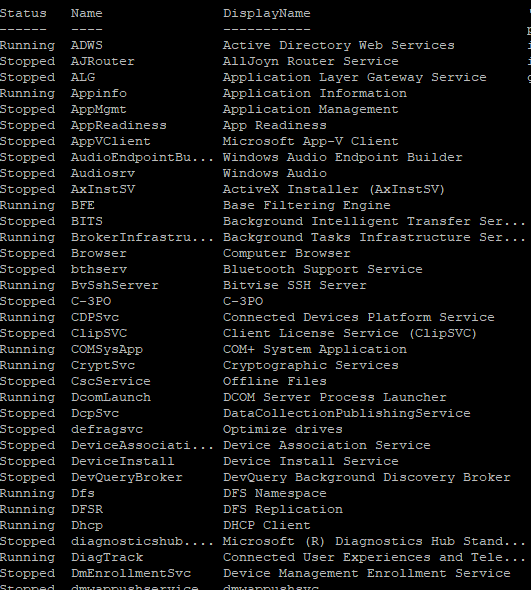
[**Century 10 -> 11**](https://underthewire.tech/century-10)

*The password for Century11 is the* ***10th*** *and* ***8th*** *word of the Windows Update service description combined* ***PLUS*** *the name of the file on the desktop.*

**Username:** century11

**Password:** windowsupdates110

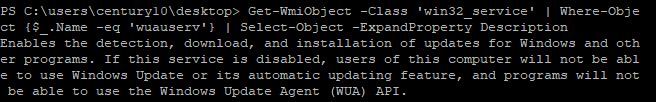
First, use ‘**get-service**’ to see a list of windows services and their names.



One service is titled Windows Update.



This puzzle is completed with the ‘**Get-WmiObject -Class 'win32\_service' | Where-Object {$\_.Name -eq 'wuauserv'} | Select-Object -ExpandProperty Description**’ command. The Get-wmiobject looks for a specific windows object, where -class ‘win32Oservice’ specifies it. Where-object then specifies what the object should be, in this case it should be named ‘wuauserv’. Then, select-object shows the object, with the added -expand property description showing only the description of the object.



By counting the words manually, the 10th word is Windows and the 8th is updates. Further, by using **ls**, we find that the name of the file on the desktop is 110. Therefore, the answer is windowsupdates110.

[**Century 11 -> 12**](https://underthewire.tech/century-11)

*The password for Century12 is the name of the hidden file within the contacts, desktop, documents, downloads, favorites, music, or videos folder in the user’s profile.*

**Username:** century12

**Password:** secret\_sauce

**Help used:** ChatGPT

Command used: **$(ls ../ -Recurse -Hidden -File | ? {$\_.name -ne "desktop.ini"}).name**

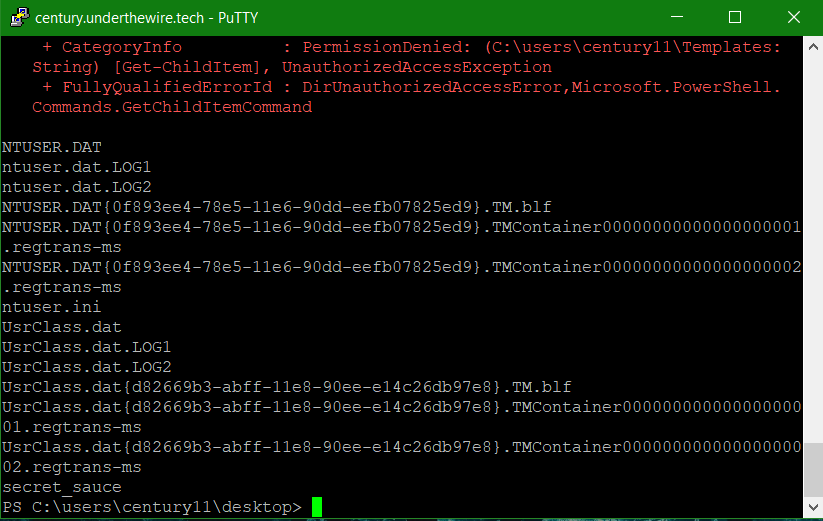
This command retrieves the names of all hidden files in the parent directory and its subdirectories, excluding files with the name "desktop.ini."

1. **`ls ../ -Recurse -Hidden -File`:** This part of the command uses `**ls`** (an alias for **`Get-ChildItem`**) to list files (not directories) that are hidden within the specified directory and its subdirectories. It combines several parameters:

* **`../`**: This specifies the directory to start the search. In this case, it's going up one level from the current directory.
* **`-Recurse`**: This parameter tells PowerShell to search for files recursively in all subdirectories.
* **`-Hidden`**: This parameter includes hidden files in the search.
* **`-File`**: This parameter filters the results to only include files, not directories.

2. **`| ? {$\_.name -ne "desktop.ini"}`**: After listing the files, the `**|**` (pipe) character is used to send the output (file objects) to the next part of the command, which is a `Where-Object` filter (`?` is an alias for `Where-Object`). This filter checks each file object's `Name` property and only lets files pass through where the name is not equal to "desktop.ini."

3. **`.name`**: Finally, after filtering the files, the `.name` property is used to extract the names of the remaining files.

****

[**Century 12 -> 13**](https://underthewire.tech/century-12)

*The password for Century13 is the description of the computer designated as a Domain Controller within this domain* ***PLUS*** *the name of the file on the desktop.*

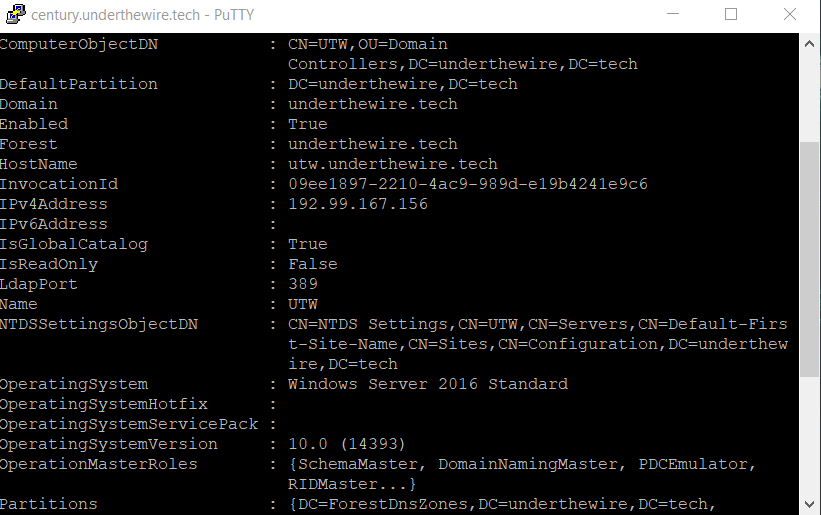
**Username:** century13

**Password:** i\_authenticate\_things

**Help used:** https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-adcomputer?view=windowsserver2022-ps&source=post\_page-----6fe886134d06--------------------------------

We use ‘Get-ADDomainController’ to retrieve information about domain controllers in an Active Directory domain.

We get the computer name which is ‘UTW’ but no description.



To get the description we use the command: **‘get-adcomputer -Identity “UTW” -Properties Description’**

The `**Get-ADComputer**` cmdlet with the `-Identity` parameter is used to retrieve detailed information about a specific computer object in Active Directory. With the command, we are attempting to retrieve information about a computer with the name "UTW" and specifically requesting the "Description" property.

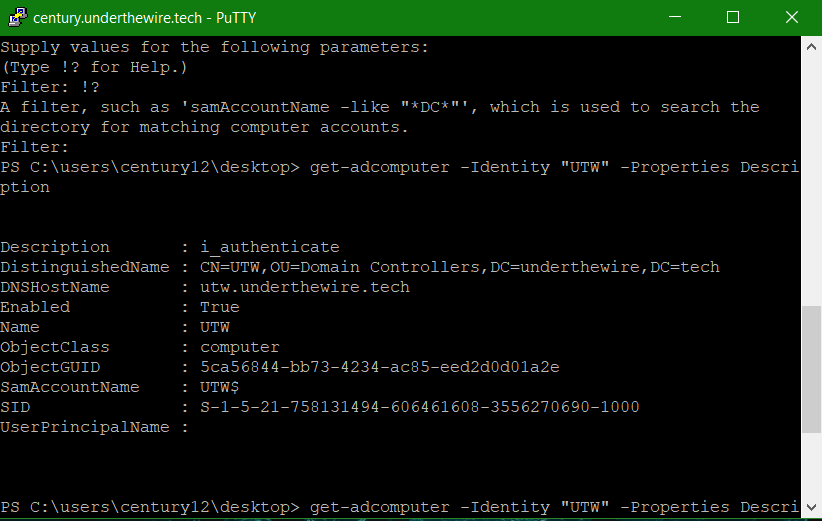
Here's what each part of your command does:

1. **`Get-ADComputer`:** This is the cmdlet used to retrieve computer object information from Active Directory.

2. **`-Identity "UTW"`:** This part specifies the computer object we want to retrieve. We are looking for a computer object with the name "UTW."

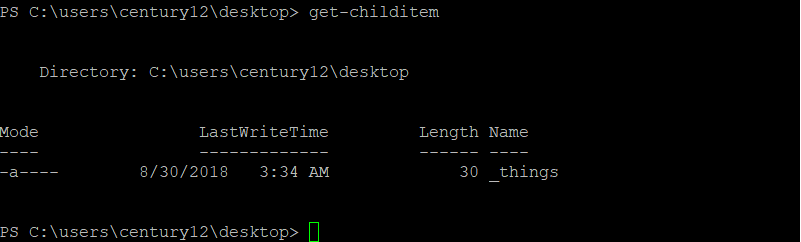
3. **`-Properties Description`:** By using the `-Properties` parameter, we are requesting additional properties of the computer object. In this case, we are specifying that we want to retrieve the "Description" property of the computer object.

So, when we run this command, it will return information about the computer object with the name "UTW," including its "Description" property if it is set. The "Description" property can be used to store additional information about the computer object.



From output the description is ‘i\_authenticate’

To check the file name on the desktop we can use the **‘get-childitem’**. The only file name that comes up is **‘\_things’**.



By adding the description and the file name on the desktop we get **‘i\_authenticate\_things’**.

[**Century 13 -> 14**](https://underthewire.tech/century-13)

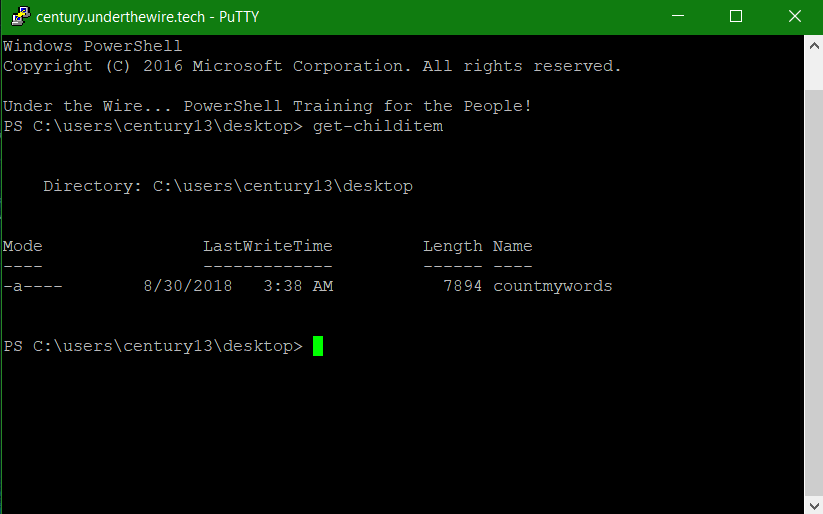
*The password for Century14 is the number of words within the file on the desktop.*

**Username:** century14

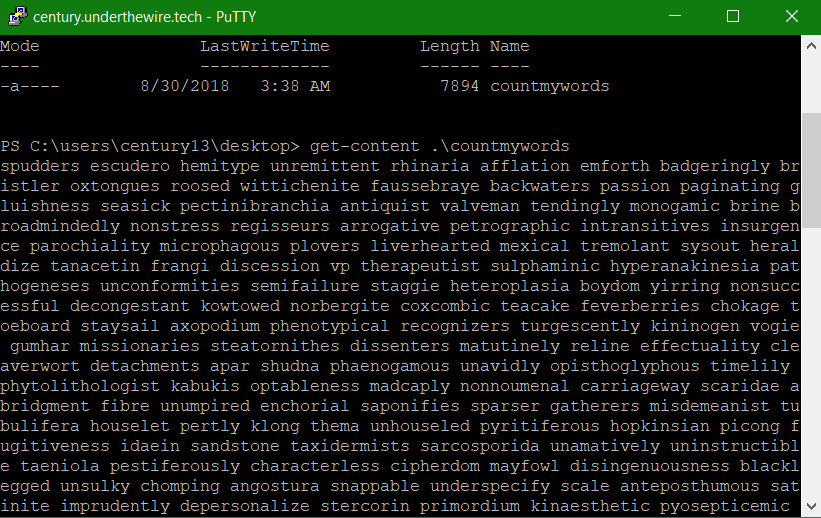
**Password:** 755

**Help used:** ChatGPT

We can start with get-childitem to see the file on the desktop.



To see what is in the file we can use get-content (‘**get-content .\countmywords’**)



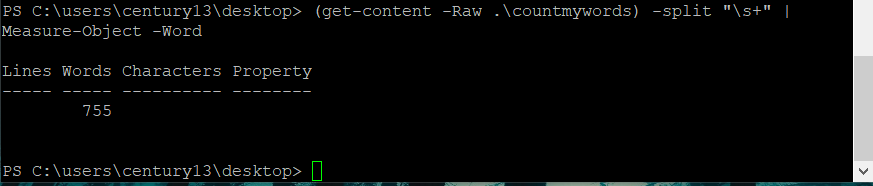
There are a lot of words. To count the number of words in the file we use **‘(get-content -Raw .\countmywords) -split “\s+” | Measure-Object -Word’.**

Here's the breakdown of the command:

1. **`(Get-Content -Raw .\countmywords)`**: This part of the command reads the content of the specified file as a single string. The `-Raw` parameter is used to read the entire file as one string.

2. **`-split "\s+"`**: This part of the command splits the string into words using a regular expression that matches one or more whitespace characters (`\s+`). This effectively separates the text into individual words.

3. **`| Measure-Object -Word`**: This part of the command uses the `Measure-Object` cmdlet with the `-Word` parameter to count the number of words in the array of words produced by the `-split` operation.



[**Century 14 -> 15**](https://underthewire.tech/century-14)

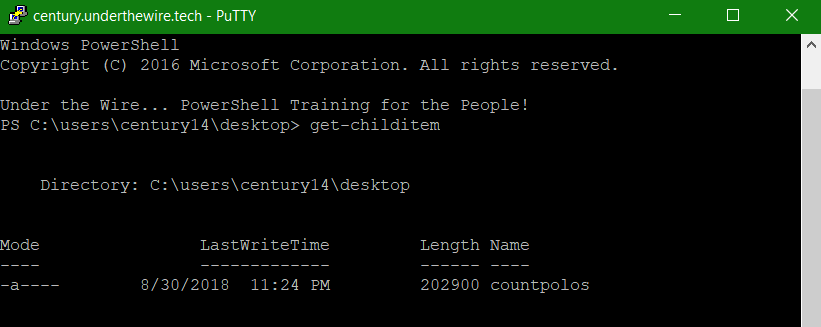
*The password for Century15 is the number of times the word “polo” appears within the file on the desktop.*

**Username:** century15

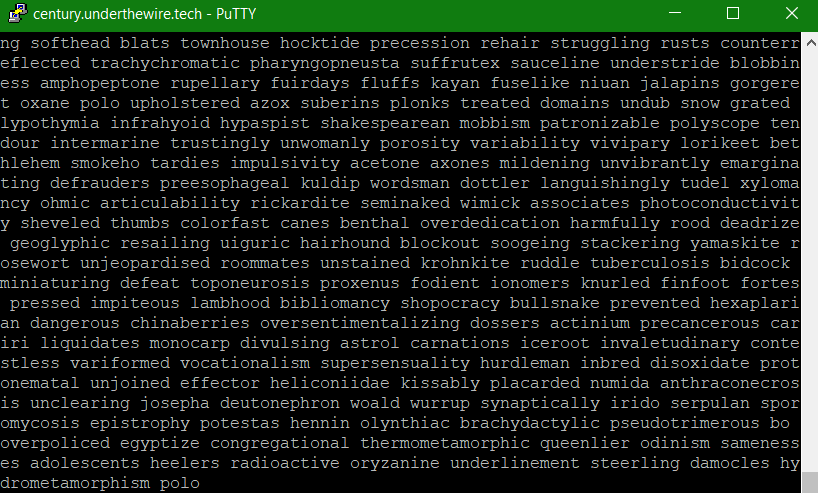
**Password:** 153

**Help used:** ChatGPT

Once again we can use get-childitem to see the file on the desktop.



When we open the file using get-content, we can see there are words that contain the word polo in them and the word polo. We need to count only the word polo.



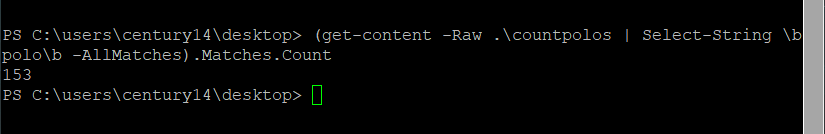
To count the number of times the word ‘polo’ appears in the file we use **‘(get-content -Raw .\countpolos | Select-String \bpolo\b -AllMatches).Matches.Count’**

Here's what the command does:

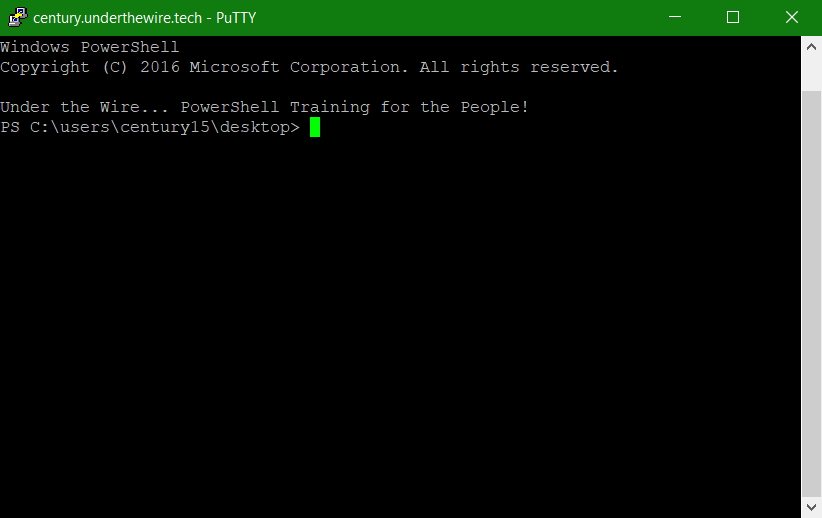
1. **`(Get-Content -Raw .\countpolos)`:** This part of the command reads the content of the specified text file as a single string.

2. **`Select-String -Pattern \bpolo\b -AllMatches`:** This part of the command uses the `Select-String` cmdlet to search for all occurrences of the word "**polo**" in the string obtained from the file. The `-Pattern` parameter specifies the word you want to search for, and the `-AllMatches` parameter ensures that all occurrences are found. ‘\b’ is used to look for that specific word only.

3. **`.Matches.Count`:** This part of the command counts the number of matches found and returns the count.



[**Century 15**](https://underthewire.tech/century-15)



**Division of Labor Report:**

**Write Up -** Aayushi Bhatia, Frances Brandofino, Amogh Patel

**Formatter -** Aayushi Bhatia, Amogh Patel, Elsa Gilsdorf

**Wargames Operator -** Aayushi Bhatia, Frances Brandofino, Amogh Patel