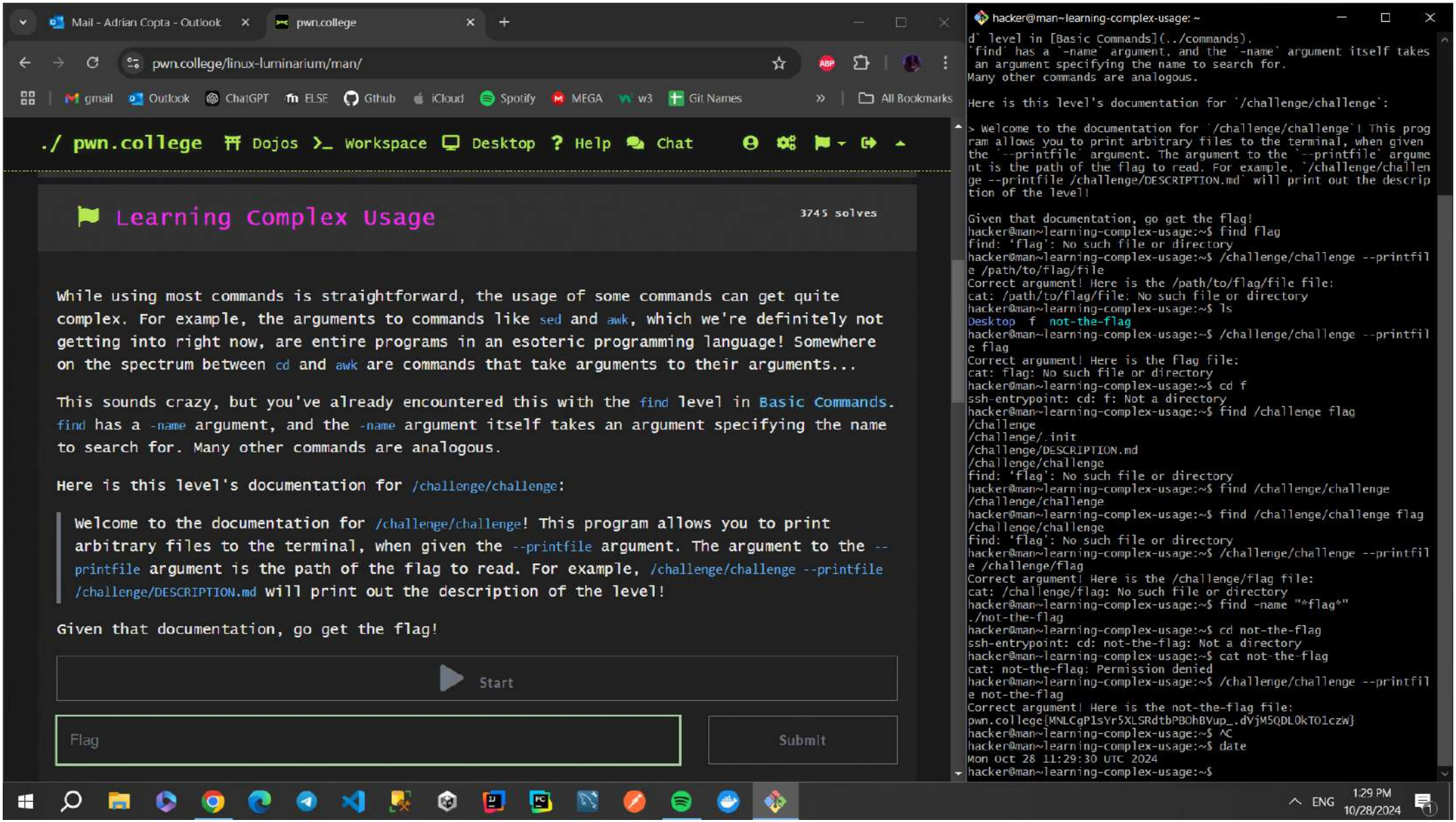
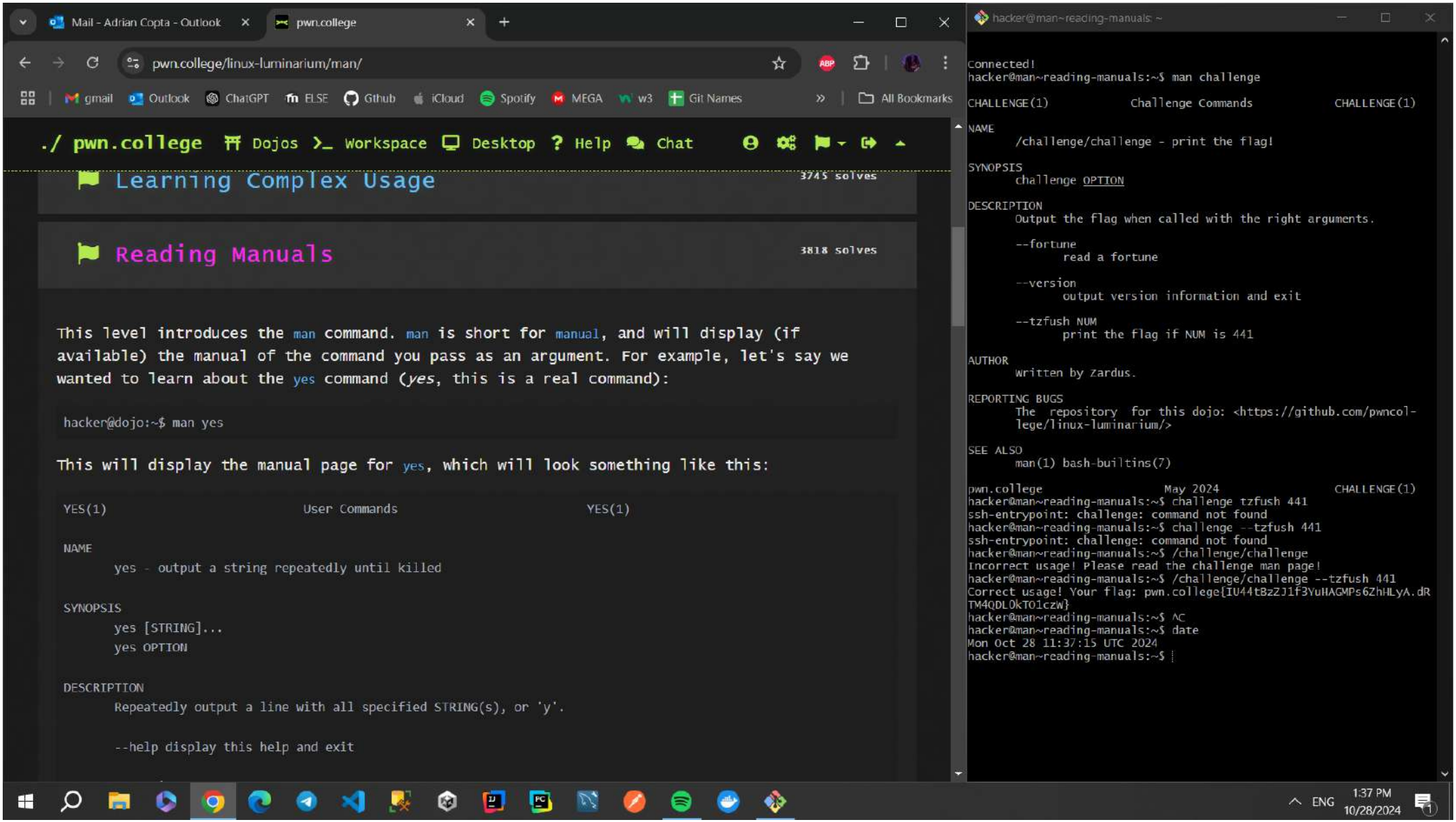
c8a

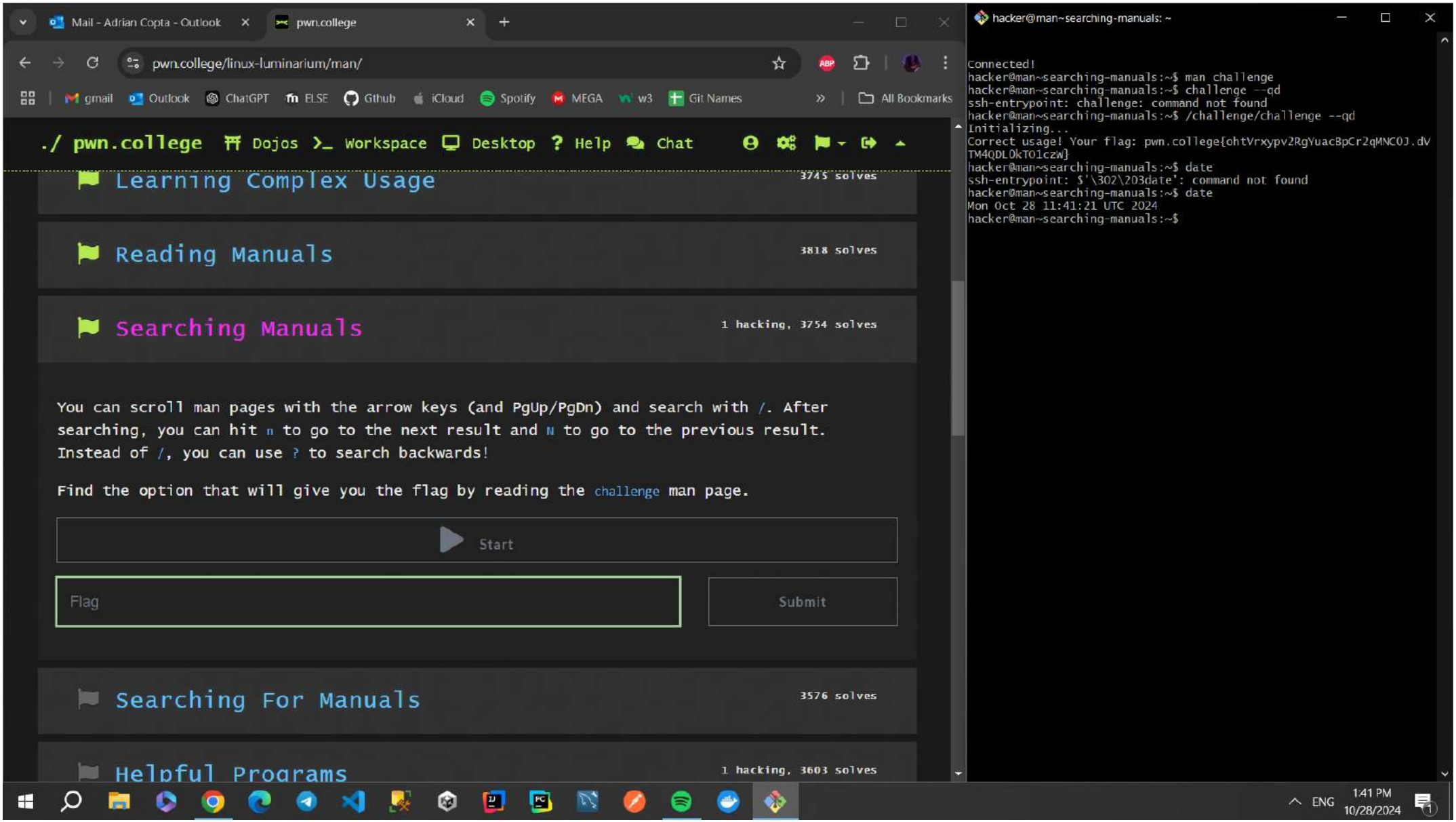
Linux Luminarium

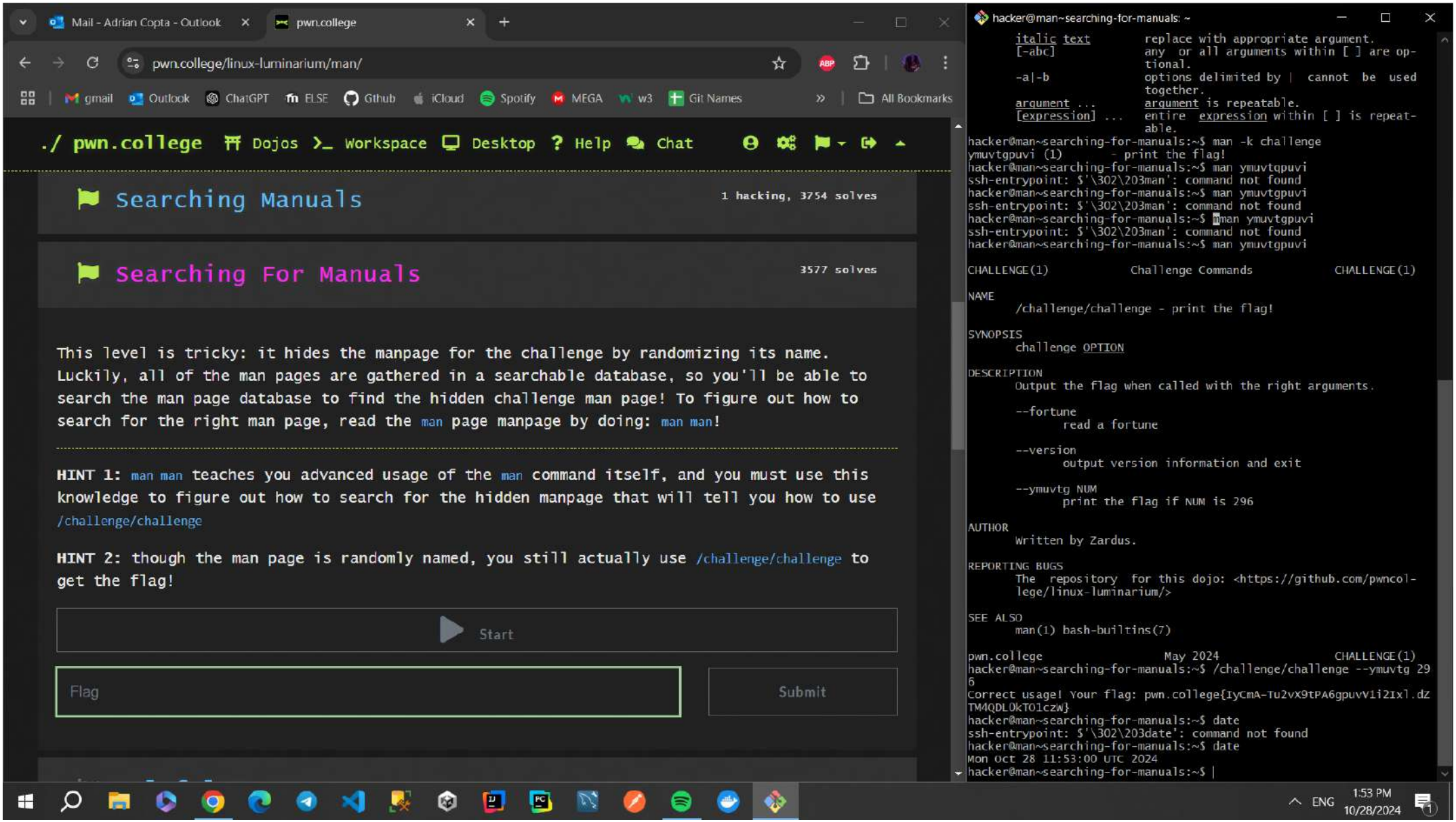
84 / 84 1758 / 8271

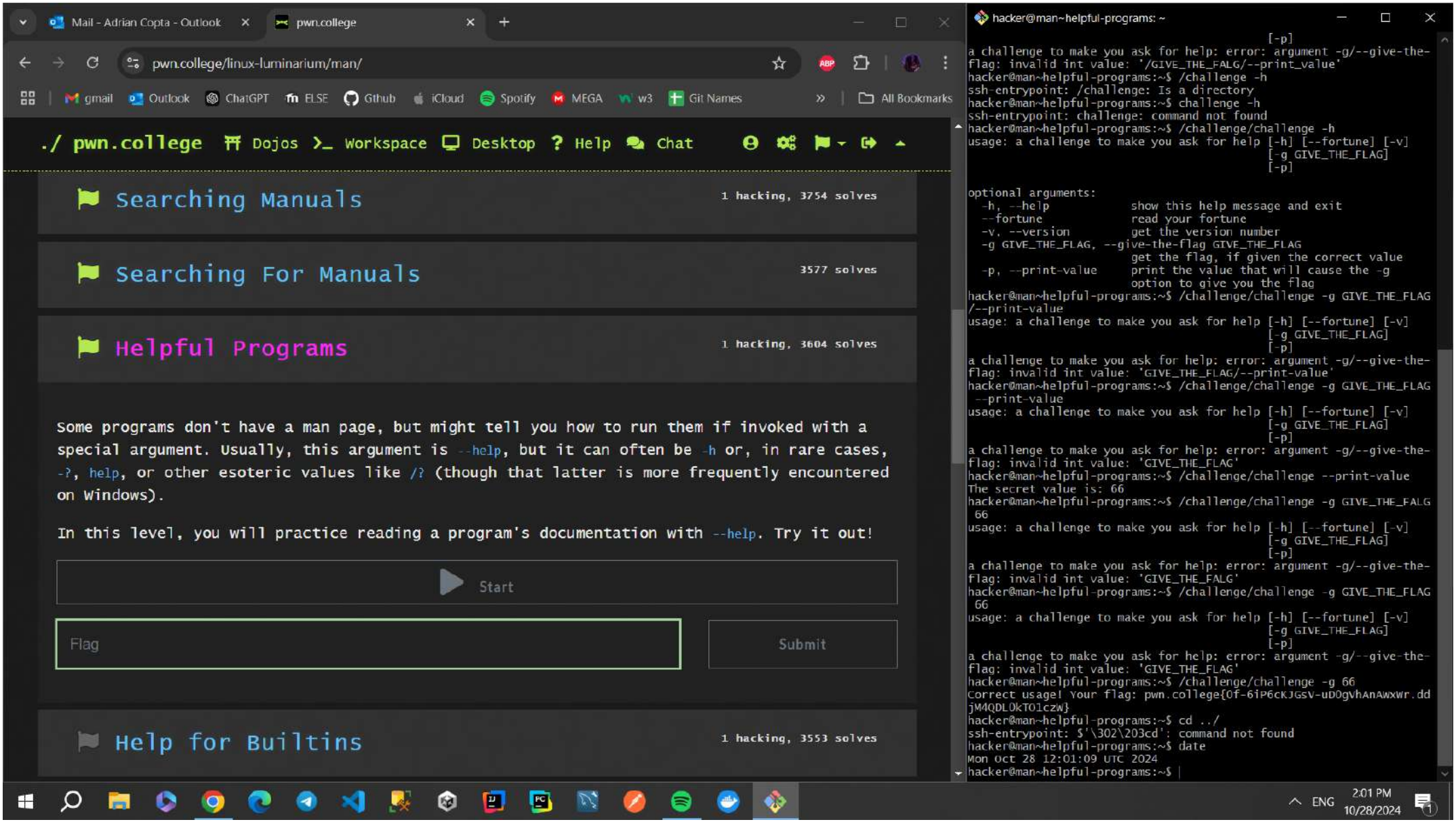
Hello Hackers	2 / 2	4588 / 7070
Pondering Paths	9 / 9	4132 / 7306
Comprehending Commands	12 / 12	2782 / 6035
Digesting Documentation	7 / 7	3463 / 5081
File Globbing	6 / 6	3276 / 4596
Practicing Piping	11 / 11	2143 / 4462
Shell Variables	8 / 8	2579 / 3984
Processes and Jobs	9 / 9	2317 / 3715
Perceiving Permissions	8 / 8	2448 / 3755
Untangling Users	4 / 4	2500 / 3161
Chaining Commands	4 / 4	2554 / 3565
Pondering PATH	4 / 4	2265 / 3361

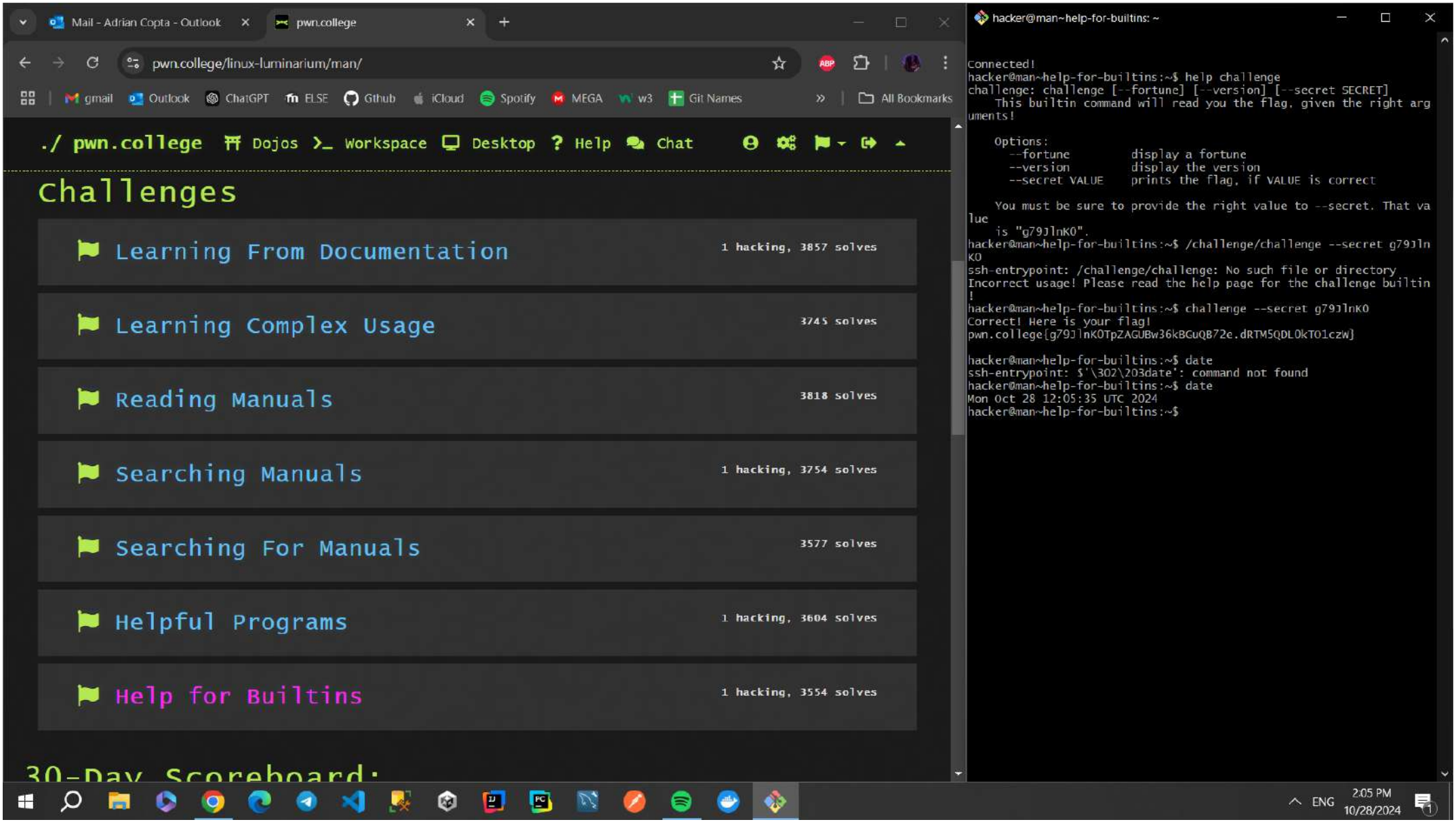

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
MEGA
w3
Git Names
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Learning Complex Usage
3745 solves
While using most commands is straightforward, the usage of some commands can get quite complex. For example, the arguments to commands like sed and awk, which we're definitely not getting into right now, are entire programs in an esoteric programming language! Somewhere on the spectrum between cd and awk are commands that take arguments to their arguments...
This sounds crazy, but you've already encountered this with the find level in Basic Commands. find has a -name argument, and the -name argument itself takes an argument specifying the name to search for. Many other commands are analogous.
Here is this level's documentation for /challenge/challenge:
Welcome to the documentation for /challenge/challenge! This program allows you to print arbitrary files to the terminal, when given the --printfile argument. The argument to the --printfile argument is the path of the flag to read. For example, /challenge/challenge --printfile /challenge/DESCRIPTION.md will print out the description of the level!
Given that documentation, go get the flag!
Start
Flag
Submit
hacker@man~learning-complex-usage: ~
d` level in [Basic Commands](../commands).
`find` has a `-name` argument, and the `-name` argument itself takes an argument specifying the name to search for.
Many other commands are analogous.
Here is this level's documentation for `/challenge/challenge`:
> Welcome to the documentation for `/challenge/challenge`! This program allows you to print arbitrary files to the terminal, when given the `--printfile` argument. The argument to the `--printfile` argument is the path of the flag to read. For example, `/challenge/challenge --printfile /challenge/DESCRIPTION.md` will print out the description of the level!
Given that documentation, go get the flag!
hacker@man~learning-complex-usage:~$ find flag
find: 'flag': No such file or directory
hacker@man~learning-complex-usage:~$ /challenge/challenge --printfile /path/to/flag/file
Correct argument! Here is the /path/to/flag/file file:
cat: /path/to/flag/file: No such file or directory
hacker@man~learning-complex-usage:~$ ls
Desktop f not-the-flag
hacker@man~learning-complex-usage:~$ /challenge/challenge --printfile flag
Correct argument! Here is the flag file:
cat: flag: No such file or directory
hacker@man~learning-complex-usage:~$ cd f
ssh-entrypoint: cd: f: Not a directory
hacker@man~learning-complex-usage:~$ find /challenge flag
/challenge
/challenge/.init
/challenge/DESCRIPTION.md
/challenge/challenge
find: 'flag': No such file or directory
hacker@man~learning-complex-usage:~$ find /challenge/challenge
/challenge/challenge
hacker@man~learning-complex-usage:~$ find /challenge/challenge flag
/challenge/challenge
find: 'flag': No such file or directory
hacker@man~learning-complex-usage:~$ /challenge/challenge --printfile /challenge/flag
Correct argument! Here is the /challenge/flag file:
cat: /challenge/flag: No such file or directory
hacker@man~learning-complex-usage:~$ find -name "*flag*"
./not-the-flag
hacker@man~learning-complex-usage:~$ cd not-the-flag
ssh-entrypoint: cd: not-the-flag: Not a directory
hacker@man~learning-complex-usage:~$ cat not-the-flag
cat: not-the-flag: Permission denied
hacker@man~learning-complex-usage:~$ /challenge/challenge --printfile not-the-flag
Correct argument! Here is the not-the-flag file:
pwn.college{MNLCgP1sYr5XLSRdtbPBOhBVup_.dVjM5QDL0kTO1czW}
hacker@man~learning-complex-usage:~$ ^C
hacker@man~learning-complex-usage:~$ date
Mon Oct 28 11:29:30 UTC 2024
hacker@man~learning-complex-usage:~$
ENG
1:29 PM
10/28/2024

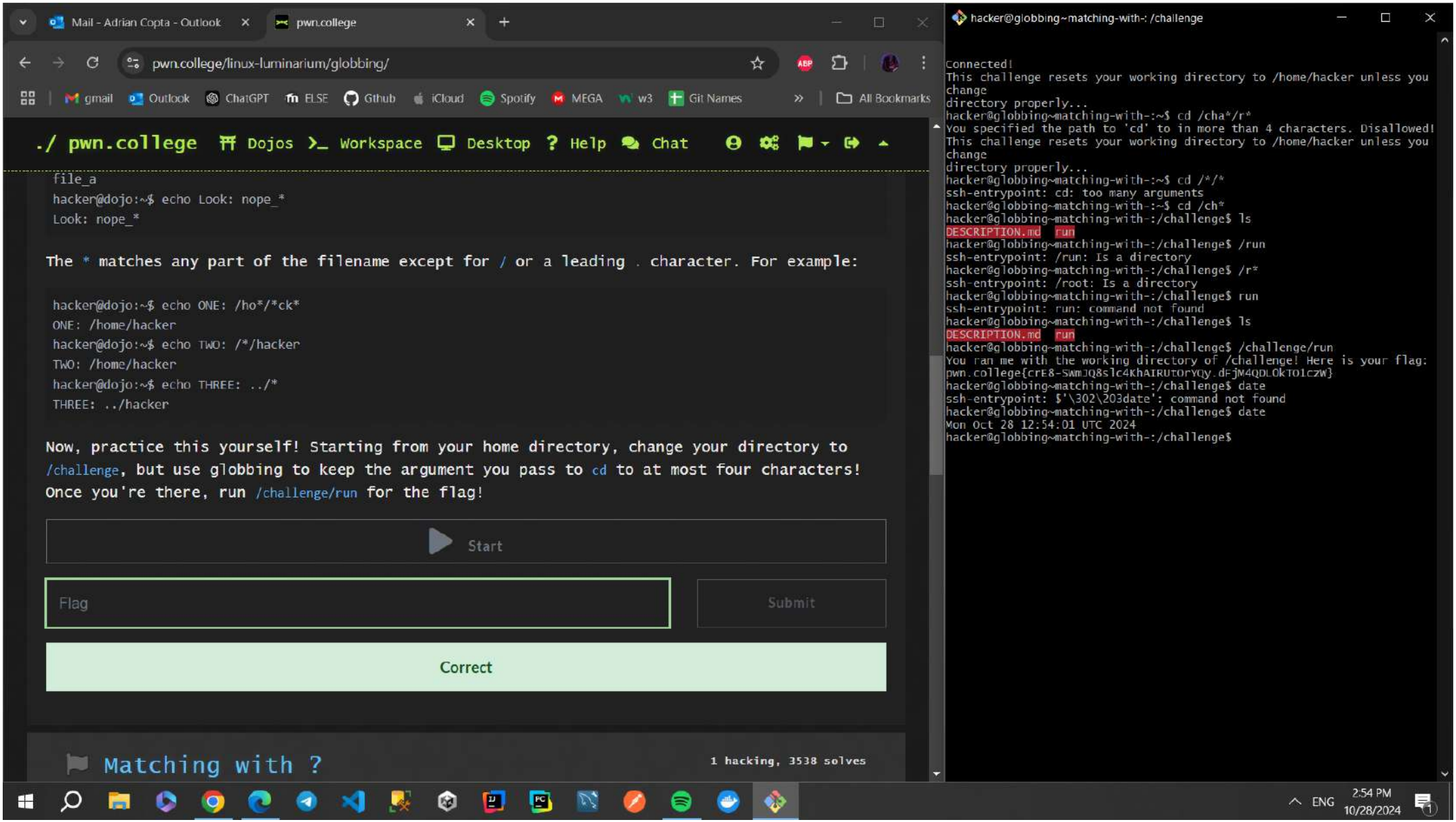

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Learning Complex Usage
3745 solves
Reading Manuals
3818 solves
This level introduces the man command. man is short for manual, and will display (if available) the manual of the command you pass as an argument. For example, let's say we wanted to learn about the yes command (yes, this is a real command):
hacker@dojo:~$ man yes
This will display the manual page for yes, which will look something like this:
YES(1) User Commands YES(1)
NAME
yes - output a string repeatedly until killed
SYNOPSIS
yes [STRING]...
yes OPTION
DESCRIPTION
Repeatedly output a line with all specified STRING(s), or 'y'.
--help display this help and exit
hacker@man~reading-manuals: ~
Connected!
hacker@man~reading-manuals:~$ man challenge
CHALLENGE(1) Challenge Commands CHALLENGE(1)
NAME
/challenge/challenge - print the flag!
SYNOPSIS
challenge OPTION
DESCRIPTION
Output the flag when called with the right arguments.
--fortune
read a fortune
--version
output version information and exit
--tzfush NUM
print the flag if NUM is 441
AUTHOR
Written by Zardus.
REPORTING BUGS
The repository for this dojo: <https://github.com/pwncollege/linux-luminarium/>
SEE ALSO
man(1) bash-builtins(7)
pwn.college May 2024 CHALLENGE(1)
hacker@man~reading-manuals:~$ challenge tzfush 441
ssh-entrypoint: challenge: command not found
hacker@man~reading-manuals:~$ challenge --tzfush 441
ssh-entrypoint: challenge: command not found
hacker@man~reading-manuals:~$ /challenge/challenge
Incorrect usage! Please read the challenge man page!
hacker@man~reading-manuals:~$ /challenge/challenge --tzfush 441
Correct usage! Your flag: pwn.college{IU44tBzZJ1f3YuHAGMPs6ZhHLyA.dRTM4QDL0kTO1czW}
hacker@man~reading-manuals:~$ ^C
hacker@man~reading-manuals:~$ date
Mon Oct 28 11:37:15 UTC 2024
hacker@man~reading-manuals:~$
ENG
1:37 PM
10/28/2024

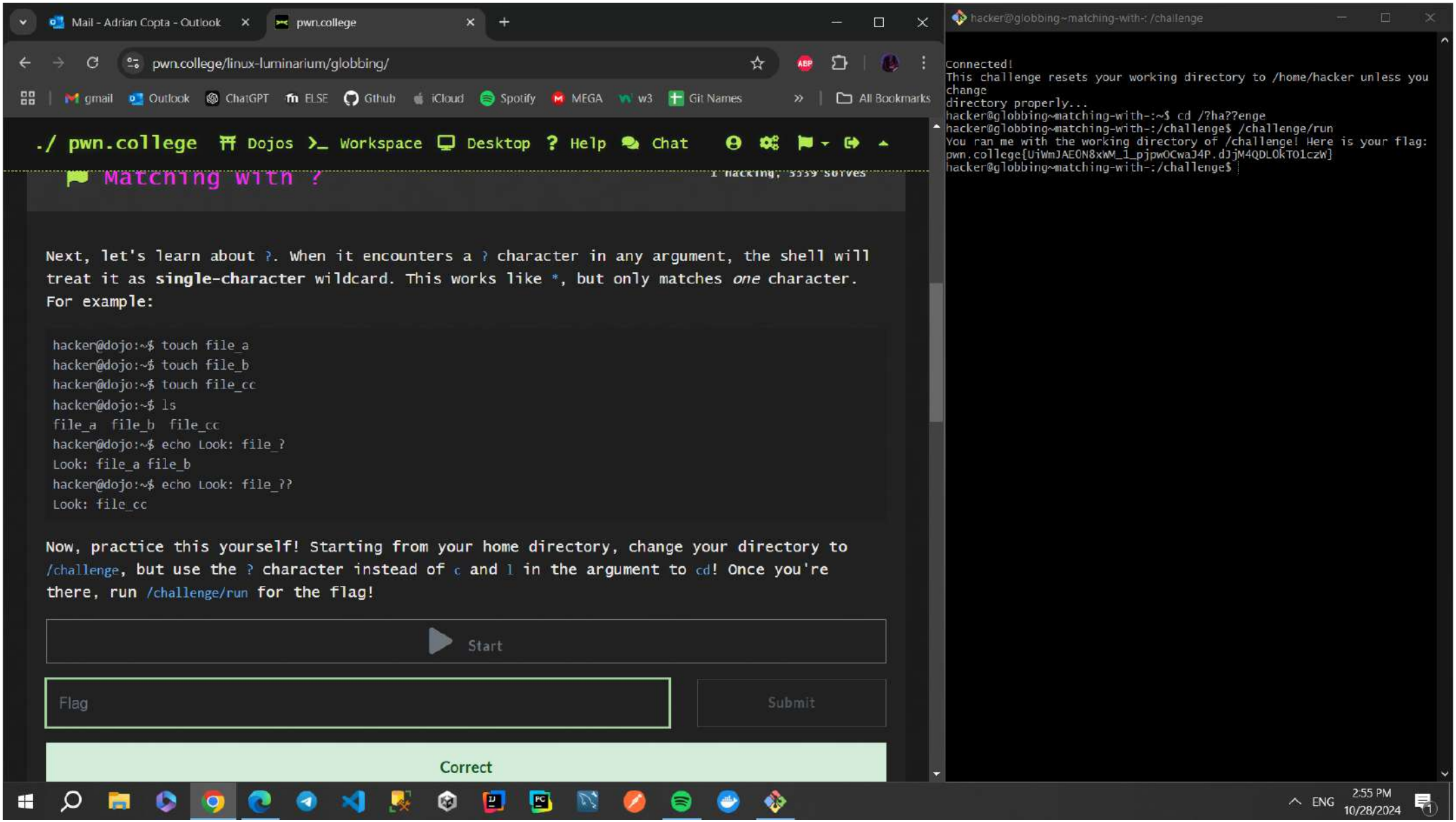

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
MEGA
w3
Git Names
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Learning Complex Usage
3745 solves
Reading Manuals
3818 solves
Searching Manuals
1 hacking, 3754 solves
You can scroll man pages with the arrow keys (and PgUp/PgDn) and search with /. After searching, you can hit n to go to the next result and N to go to the previous result. Instead of /, you can use ? to search backwards!
Find the option that will give you the flag by reading the challenge man page.
Start
Flag
Submit
Searching For Manuals
3576 solves
Helpful Programs
1 hacking, 3603 solves
hacker@man~searching-manuals: ~
Connected!
hacker@man~searching-manuals:~$ man challenge
hacker@man~searching-manuals:~$ challenge --qd
ssh-entrypoint: challenge: command not found
hacker@man~searching-manuals:~$ /challenge/challenge --qd
Initializing...
Correct usage! Your flag: pwn.college{ohtVrxypv2RgYuacBpCr2qMNC0J.dvTM4QDL0kTO1czW}
hacker@man~searching-manuals:~$ date
ssh-entrypoint: $'\302\203date': command not found
hacker@man~searching-manuals:~$ date
Mon Oct 28 11:41:21 UTC 2024
hacker@man~searching-manuals:~$
ENG
1:41 PM
10/28/2024

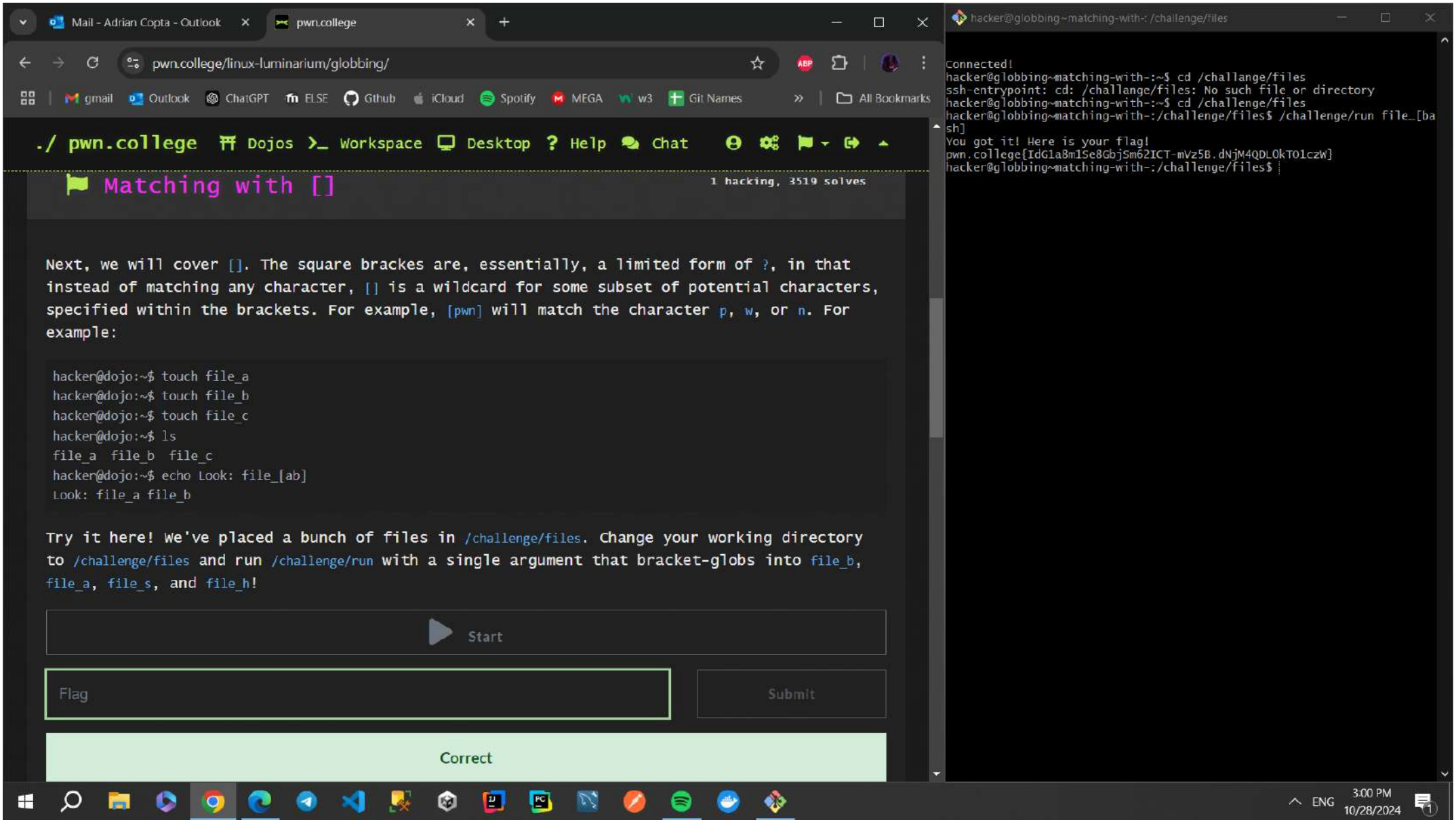

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Searching Manuals
1 hacking, 3754 solves
Searching For Manuals
3577 solves
This level is tricky: it hides the manpage for the challenge by randomizing its name. Luckily, all of the man pages are gathered in a searchable database, so you'll be able to search the man page database to find the hidden challenge man page! To figure out how to search for the right man page, read the man page manpage by doing: man man!
HINT 1: man man teaches you advanced usage of the man command itself, and you must use this knowledge to figure out how to search for the hidden manpage that will tell you how to use /challenge/challenge
HINT 2: though the man page is randomly named, you still actually use /challenge/challenge to get the flag!
Start
Flag
Submit
hacker@man~searching-for-manuals: ~
italic text replace with appropriate argument.
[-abc] any or all arguments within [ ] are optional.
-a|-b options delimited by | cannot be used together.
argument ... argument is repeatable.
[expression] ... entire expression within [ ] is repeatable.
hacker@man~searching-for-manuals:~$ man -k challenge
ymuvtgpuvi (1) - print the flag!
hacker@man~searching-for-manuals:~$ man ymuvtgpuvi
ssh-entrypoint: $'\302\203man': command not found
hacker@man~searching-for-manuals:~$ man ymuvtgpuvi
ssh-entrypoint: $'\302\203man': command not found
hacker@man~searching-for-manuals:~$ man ymuvtgpuvi
ssh-entrypoint: $'\302\203man': command not found
hacker@man~searching-for-manuals:~$ man ymuvtgpuvi
CHALLENGE(1) Challenge Commands CHALLENGE(1)
NAME
/challenge/challenge - print the flag!
SYNOPSIS
challenge OPTION
DESCRIPTION
Output the flag when called with the right arguments.
--fortune
read a fortune
--version
output version information and exit
--ymuvtg NUM
print the flag if NUM is 296
AUTHOR
Written by Zardus.
REPORTING BUGS
The repository for this dojo: <https://github.com/pwncollege/linux-luminarium/>
SEE ALSO
man(1) bash-builtins(7)
pwn.college May 2024 CHALLENGE(1)
hacker@man~searching-for-manuals:~$ /challenge/challenge --ymuvtg 296
Correct usage! Your flag: pwn.college{1yCmA-Tu2vx9tPA6gpuvV1i2Ixl.dzTM4QDLOkTO1czW}
hacker@man~searching-for-manuals:~$ date
ssh-entrypoint: $'\302\203date': command not found
hacker@man~searching-for-manuals:~$ date
Mon Oct 28 11:53:00 UTC 2024
hacker@man~searching-for-manuals:~$
ENG
1:53 PM
10/28/2024

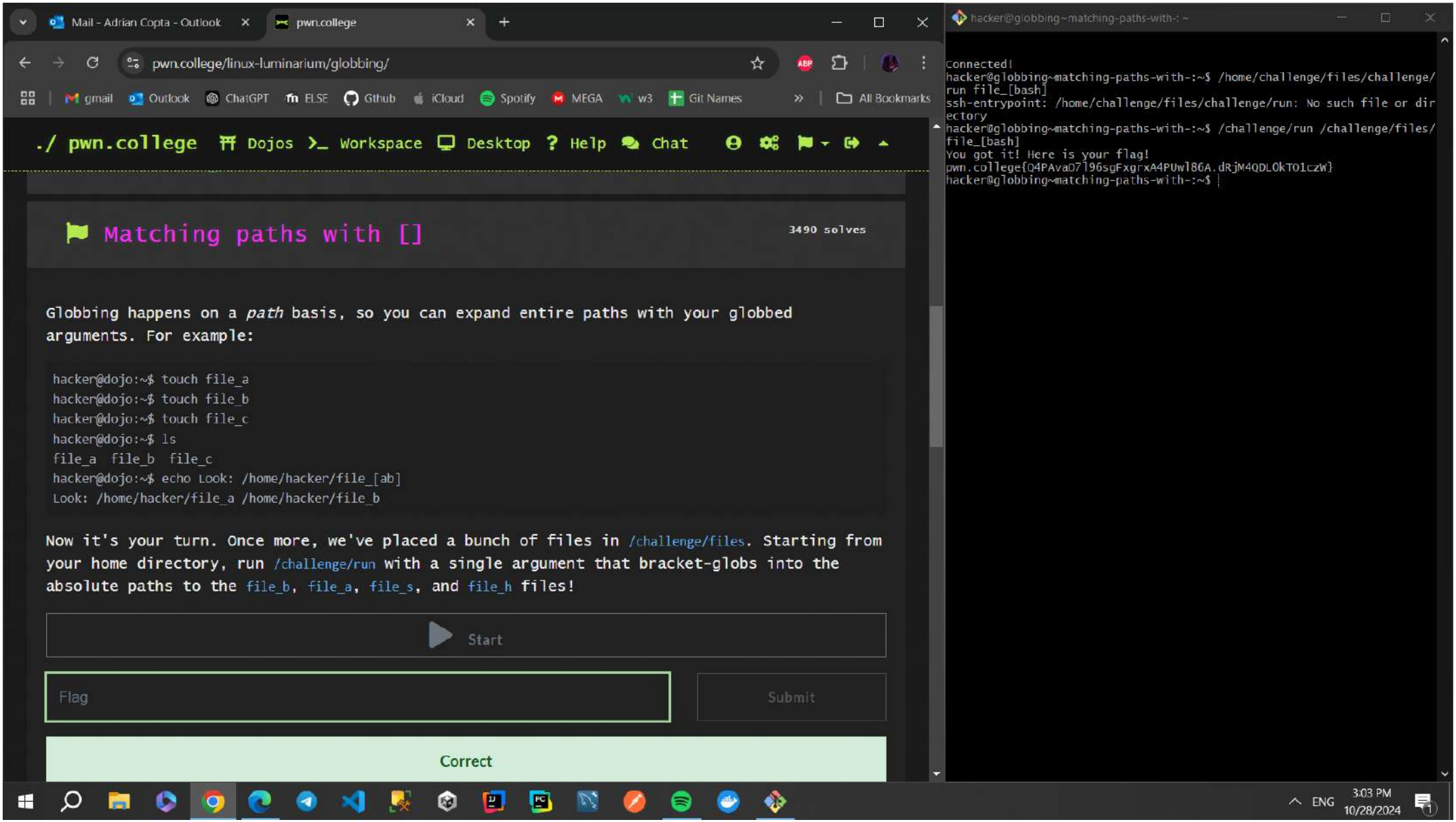

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
MEGA
w3
Git Names
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Searching Manuals
1 hacking, 3754 solves
Searching For Manuals
3577 solves
Helpful Programs
1 hacking, 3604 solves
Some programs don't have a man page, but might tell you how to run them if invoked with a special argument. Usually, this argument is --help, but it can often be -h or, in rare cases, -?, help, or other esoteric values like /? (though that latter is more frequently encountered on Windows).
In this level, you will practice reading a program's documentation with --help. Try it out!
Start
Flag
Submit
Help for Builtins
1 hacking, 3553 solves
hacker@man~helpful-programs: ~
[-p]
a challenge to make you ask for help: error: argument -g/--give-the-flag: invalid int value: '/GIVE_THE_FALG/--print_value'
hacker@man~helpful-programs:~$ /challenge -h
ssh-entrypoint: /challenge: Is a directory
hacker@man~helpful-programs:~$ challenge -h
ssh-entrypoint: challenge: command not found
hacker@man~helpful-programs:~$ /challenge/challenge -h
usage: a challenge to make you ask for help [-h] [--fortune] [-v]
[-g GIVE_THE_FLAG]
[-p]
optional arguments:
-h, --help show this help message and exit
--fortune read your fortune
-v, --version get the version number
-g GIVE_THE_FLAG, --give-the-flag GIVE_THE_FLAG
get the flag, if given the correct value
-p, --print-value print the value that will cause the -g option to give you the flag
hacker@man~helpful-programs:~$ /challenge/challenge -g GIVE_THE_FLAG /--print-value
usage: a challenge to make you ask for help [-h] [--fortune] [-v]
[-g GIVE_THE_FLAG]
[-p]
a challenge to make you ask for help: error: argument -g/--give-the-flag: invalid int value: 'GIVE_THE_FLAG/--print-value'
hacker@man~helpful-programs:~$ /challenge/challenge -g GIVE_THE_FLAG --print-value
usage: a challenge to make you ask for help [-h] [--fortune] [-v]
[-g GIVE_THE_FLAG]
[-p]
a challenge to make you ask for help: error: argument -g/--give-the-flag: invalid int value: 'GIVE_THE_FLAG'
hacker@man~helpful-programs:~$ /challenge/challenge --print-value
The secret value is: 66
hacker@man~helpful-programs:~$ /challenge/challenge -g GIVE_THE_FALG 66
usage: a challenge to make you ask for help [-h] [--fortune] [-v]
[-g GIVE_THE_FLAG]
[-p]
a challenge to make you ask for help: error: argument -g/--give-the-flag: invalid int value: 'GIVE_THE_FALG'
hacker@man~helpful-programs:~$ /challenge/challenge -g GIVE_THE_FLAG 66
usage: a challenge to make you ask for help [-h] [--fortune] [-v]
[-g GIVE_THE_FLAG]
[-p]
a challenge to make you ask for help: error: argument -g/--give-the-flag: invalid int value: 'GIVE_THE_FLAG'
hacker@man~helpful-programs:~$ /challenge/challenge -g 66
Correct usage! Your flag: pwn.college{Of-6iP6cKJGsV-uDOgVhAnAWxWr.dd jM4QDLOkTO1czW}
hacker@man~helpful-programs:~$ cd ../
ssh-entrypoint: $'\302\203cd': command not found
hacker@man~helpful-programs:~$ date
Mon Oct 28 12:01:09 UTC 2024
hacker@man~helpful-programs:~$
ENG
2:01 PM
10/28/2024

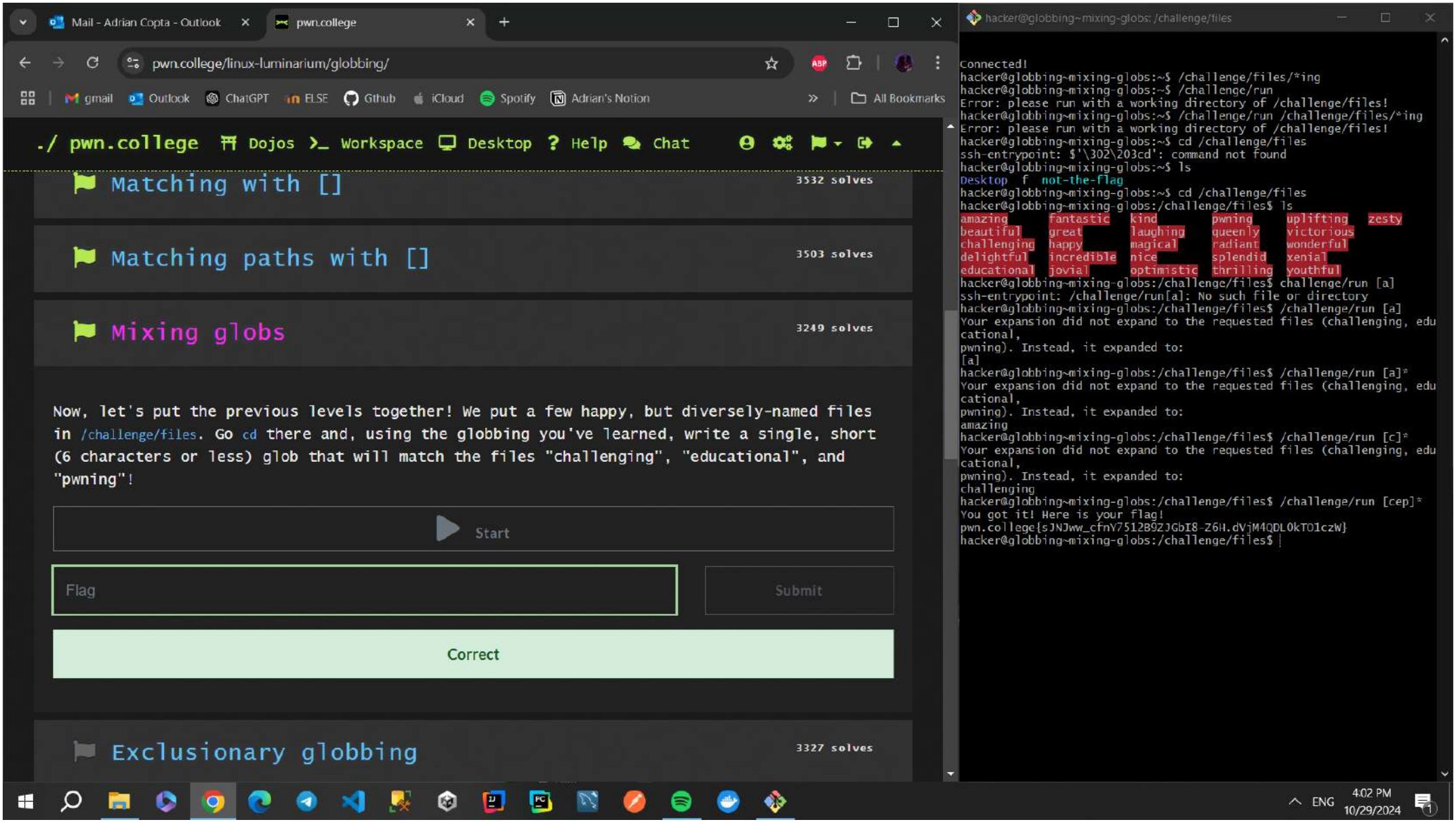

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/man/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
MEGA
w3
Git Names
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Challenges
Learning From Documentation
1 hacking, 3857 solves
Learning Complex Usage
3745 solves
Reading Manuals
3818 solves
Searching Manuals
1 hacking, 3754 solves
Searching For Manuals
3577 solves
Helpful Programs
1 hacking, 3604 solves
Help for Builtins
1 hacking, 3554 solves
30-Day Scoreboard:
hacker@man~help-for-builtins: ~
Connected!
hacker@man~help-for-builtins:~$ help challenge
challenge: challenge [--fortune] [--version] [--secret SECRET]
This builtin command will read you the flag, given the right arguments!
Options:
--fortune display a fortune
--version display the version
--secret VALUE prints the flag, if VALUE is correct
You must be sure to provide the right value to --secret. That value
is "g79JlnKO".
hacker@man~help-for-builtins:~$ /challenge/challenge --secret g79JlnKO
ssh-entrypoint: /challenge/challenge: No such file or directory
Incorrect usage! Please read the help page for the challenge builtin!
hacker@man~help-for-builtins:~$ challenge --secret g79JlnKO
Correct! Here is your flag!
pwn.college{g79JlnKOTpZAGUBw36kBGuQB72e.dRTM5QDLOkTO1czWJ
hacker@man~help-for-builtins:~$ date
ssh-entrypoint: $'\302\203date': command not found
hacker@man~help-for-builtins:~$ date
Mon Oct 28 12:05:35 UTC 2024
hacker@man~help-for-builtins:~$
ENG
2:05 PM
10/28/2024

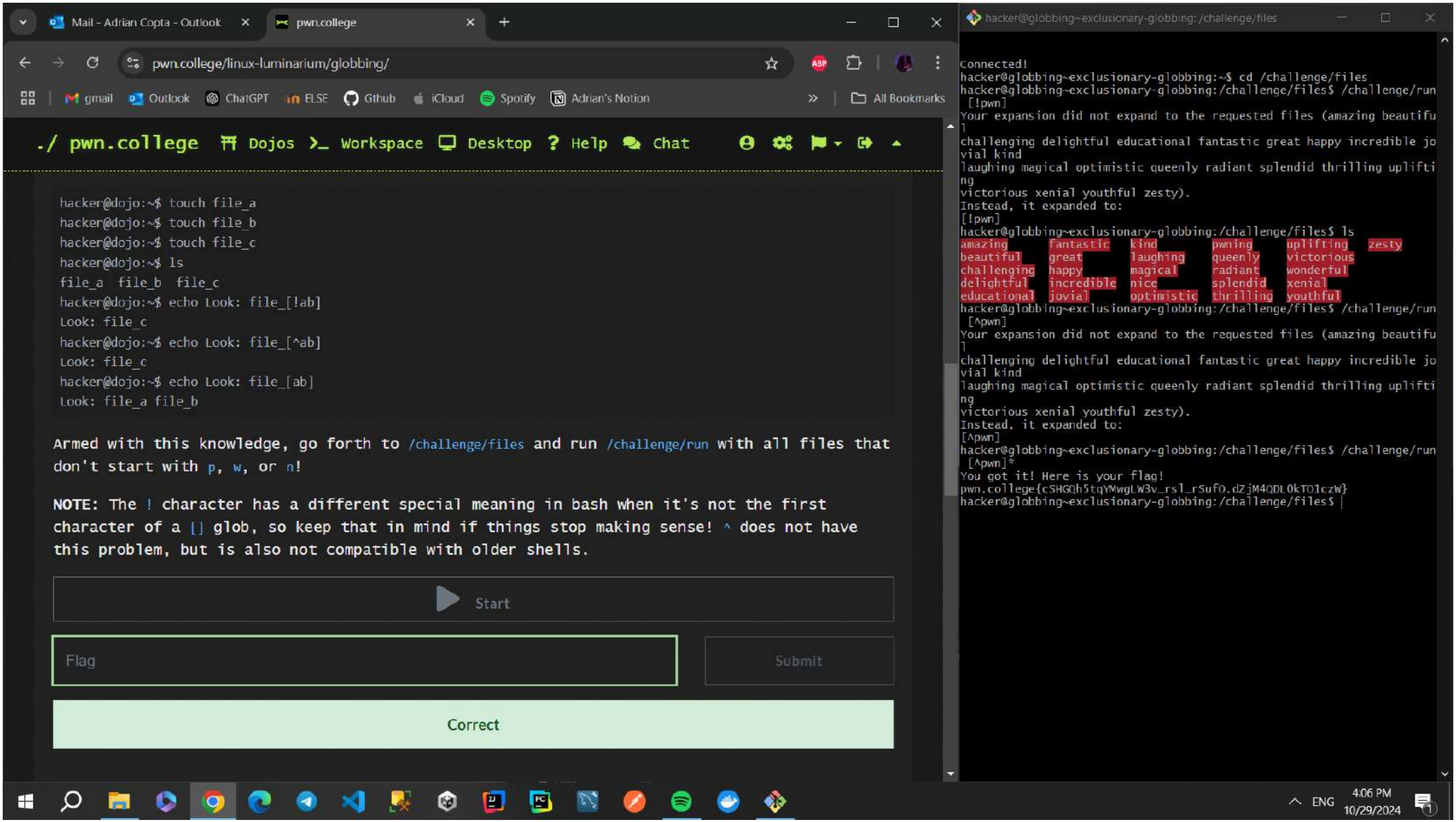

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
./ pwn.college Dojos Workspace Desktop Help Chat
file_a
hacker@dojo:~$ echo Look: nope_*
Look: nope_*
The * matches any part of the filename except for / or a leading . character. For example:
hacker@dojo:~$ echo ONE: /ho*/*ck*
ONE: /home/hacker
hacker@dojo:~$ echo TWO: /*/hacker
TWO: /home/hacker
hacker@dojo:~$ echo THREE: ../*
THREE: ../hacker
Now, practice this yourself! Starting from your home directory, change your directory to /challenge, but use globbing to keep the argument you pass to cd to at most four characters! Once you're there, run /challenge/run for the flag!
Start
Flag
Submit
Correct
Matching with ?
1 hacking, 3538 solves
hacker@globbing~matching-with-: /challenge
Connected!
This challenge resets your working directory to /home/hacker unless you change
directory properly...
hacker@globbing~matching-with-:~$ cd /cha*/r*
You specified the path to 'cd' to in more than 4 characters. Disallowed!
This challenge resets your working directory to /home/hacker unless you change
directory properly...
hacker@globbing~matching-with-:~$ cd /*/*
ssh-entrypoint: cd: too many arguments
hacker@globbing~matching-with-:~$ cd /ch*
hacker@globbing~matching-with-:/challenge$ ls
DESCRIPTION.md run
hacker@globbing~matching-with-:/challenge$ /run
ssh-entrypoint: /run: Is a directory
hacker@globbing~matching-with-:/challenge$ /r*
ssh-entrypoint: /root: Is a directory
hacker@globbing~matching-with-:/challenge$ run
ssh-entrypoint: run: command not found
hacker@globbing~matching-with-:/challenge$ ls
DESCRIPTION.md run
hacker@globbing~matching-with-:/challenge$ /challenge/run
You ran me with the working directory of /challenge! Here is your flag:
pwn.college{crE8-SWmJQ8slc4KhAIRUtOrYQy.dFjM4QDLOkTO1czW}
hacker@globbing~matching-with-:/challenge$ date
ssh-entrypoint: $'\302\203date': command not found
hacker@globbing~matching-with-:/challenge$ date
Mon Oct 28 12:54:01 UTC 2024
hacker@globbing~matching-with-:/challenge$
ENG
2:54 PM
10/28/2024

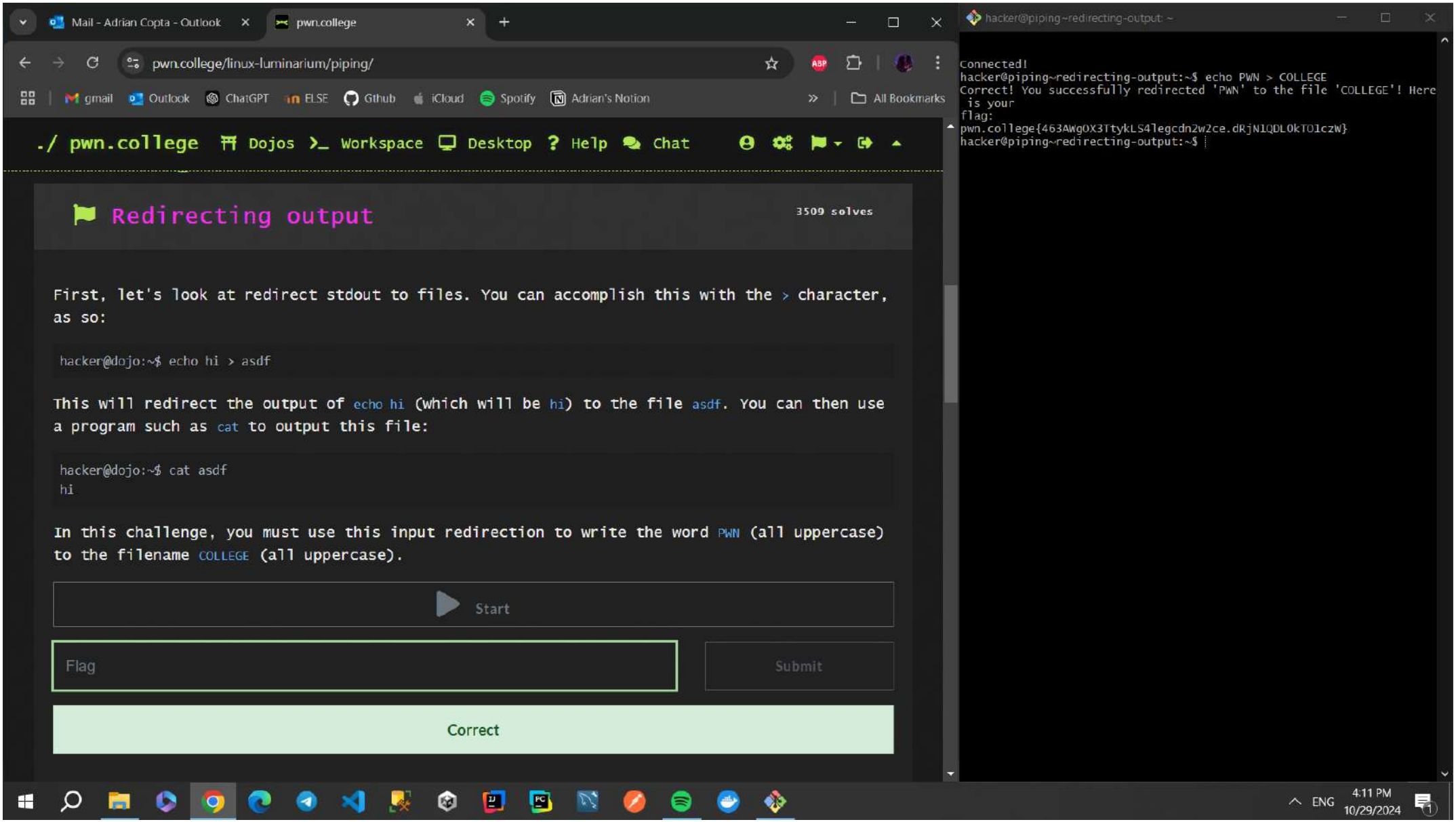

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
MEGA
w3
Git Names
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Matching With ?
1 hacking, 3339 solves
Next, let's learn about ?. When it encounters a ? character in any argument, the shell will treat it as single-character wildcard. This works like *, but only matches one character. For example:
hacker@dojo:~$ touch file_a
hacker@dojo:~$ touch file_b
hacker@dojo:~$ touch file_cc
hacker@dojo:~$ ls
file_a file_b file_cc
hacker@dojo:~$ echo Look: file_?
Look: file_a file_b
hacker@dojo:~$ echo Look: file_??
Look: file_cc
Now, practice this yourself! Starting from your home directory, change your directory to /challenge, but use the ? character instead of c and l in the argument to cd! Once you're there, run /challenge/run for the flag!
Start
Flag
Submit
Correct
hacker@globbing~matching-with~: /challenge
Connected!
This challenge resets your working directory to /home/hacker unless you change
directory properly...
hacker@globbing~matching-with~:~$ cd /?ha??enge
hacker@globbing~matching-with~:/challenge$ /challenge/run
You ran me with the working directory of /challenge! Here is your flag:
pwn.college{UiWmJAEON8xWM_1_pjpwOCwaJ4P.dJjM4QDLOkTO1czW}
hacker@globbing~matching-with~:/challenge$
ENG
2:55 PM
10/28/2024

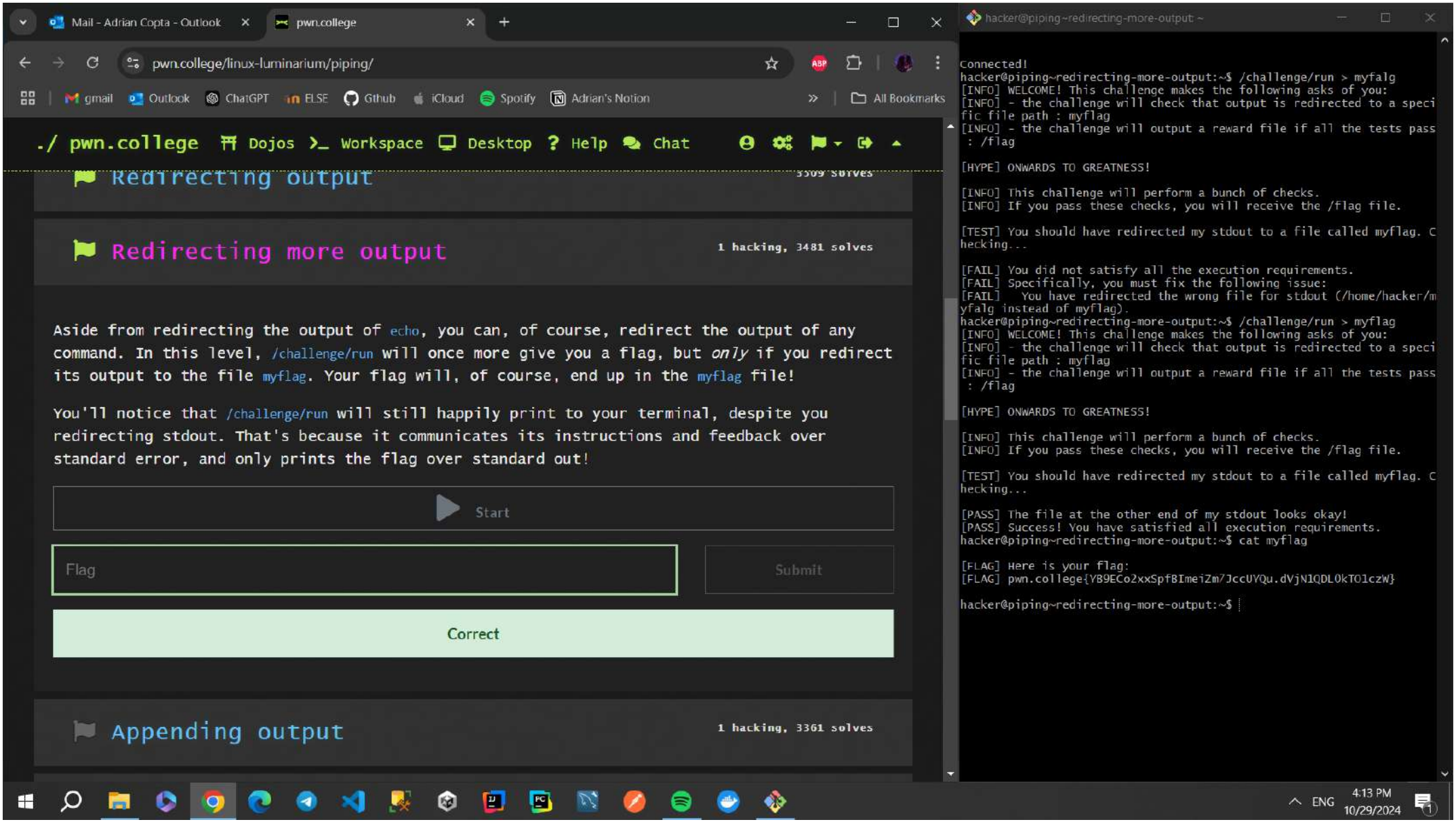

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Matching with []
1 hacking, 3519 solves
Next, we will cover []. The square brackes are, essentially, a limited form of ?, in that instead of matching any character, [] is a wildcard for some subset of potential characters, specified within the brackets. For example, [pwn] will match the character p, w, or n. For example:
hacker@dojo:~$ touch file_a
hacker@dojo:~$ touch file_b
hacker@dojo:~$ touch file_c
hacker@dojo:~$ ls
file_a file_b file_c
hacker@dojo:~$ echo Look: file_[ab]
Look: file_a file_b
Try it here! We've placed a bunch of files in /challenge/files. Change your working directory to /challenge/files and run /challenge/run with a single argument that bracket-globs into file_b, file_a, file_s, and file_h!
Start
Flag
Submit
Correct
hacker@globbing~matching-with-: /challenge/files
Connected!
hacker@globbing~matching-with-:~$ cd /challange/files
ssh-entrypoint: cd: /challange/files: No such file or directory
hacker@globbing~matching-with-:~$ cd /challenge/files
hacker@globbing~matching-with-:/challenge/files$ /challenge/run file_[bash]
You got it! Here is your flag!
pwn.college{IdG1a8m1Se8GbjSm62ICT-mVz5B.dNjM4QDL0kTO1czW}
hacker@globbing~matching-with-:/challenge/files$
ENG
3:00 PM
10/28/2024

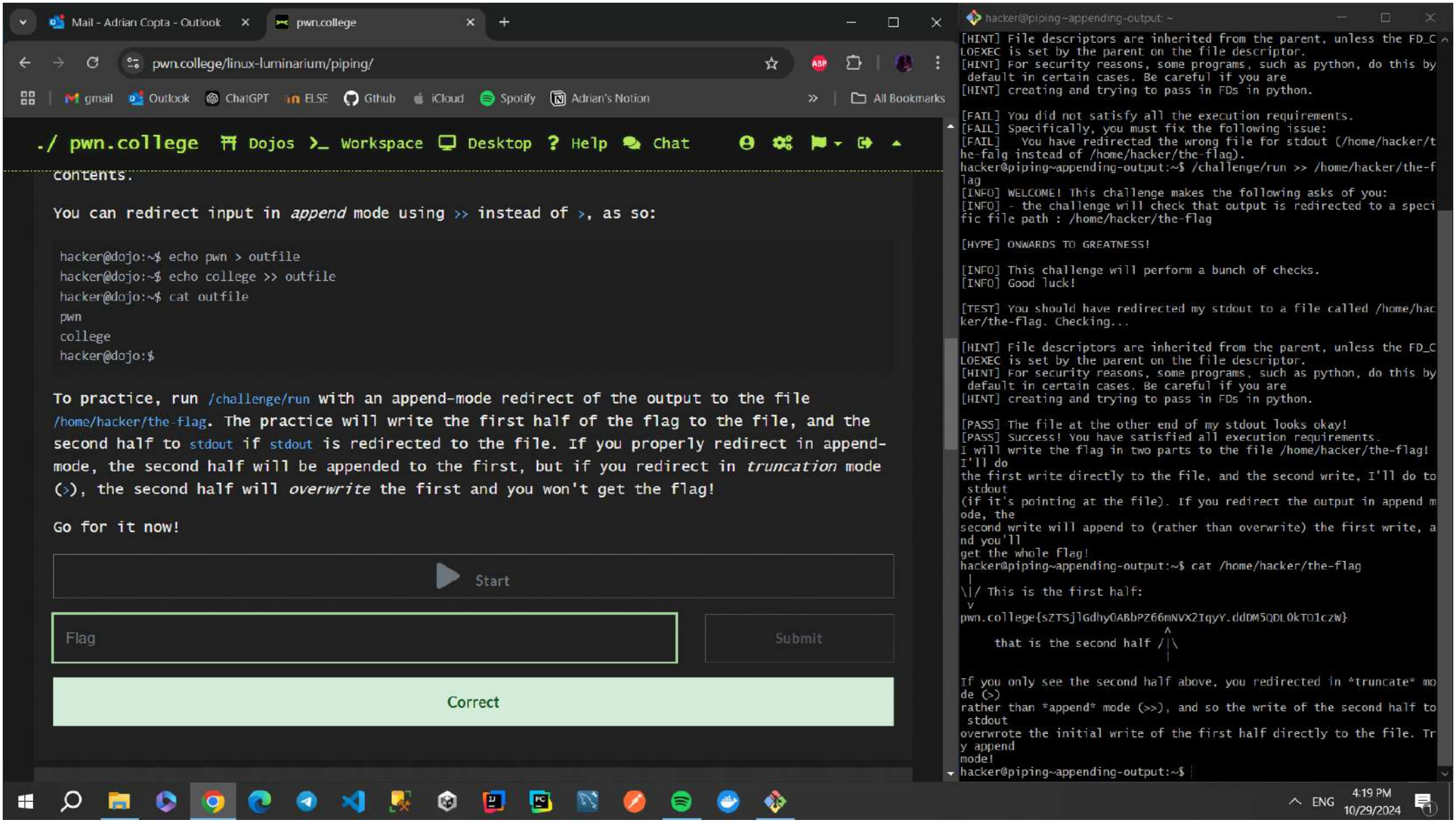

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Matching paths with []
3490 solves
Globbing happens on a path basis, so you can expand entire paths with your globbed arguments. For example:
hacker@dojo:~$ touch file_a
hacker@dojo:~$ touch file_b
hacker@dojo:~$ touch file_c
hacker@dojo:~$ ls
file_a file_b file_c
hacker@dojo:~$ echo Look: /home/hacker/file_[ab]
Look: /home/hacker/file_a /home/hacker/file_b
Now it's your turn. Once more, we've placed a bunch of files in /challenge/files. Starting from your home directory, run /challenge/run with a single argument that bracket-globs into the absolute paths to the file_b, file_a, file_s, and file_h files!
Start
Flag
Submit
Correct
hacker@globbing~matching-paths-with-:~
Connected!
hacker@globbing~matching-paths-with-:~$ /home/challenge/files/challenge/run file_[bash]
ssh-entrypoint: /home/challenge/files/challenge/run: No such file or directory
hacker@globbing~matching-paths-with-:~$ /challenge/run /challenge/files/file_[bash]
You got it! Here is your flag!
pwn.college{Q4PAvaO7196sgFxgrxA4PUwl86A.dRjM4QDLOkTO1czW}
hacker@globbing~matching-paths-with-:~$
ENG
3:03 PM
10/28/2024

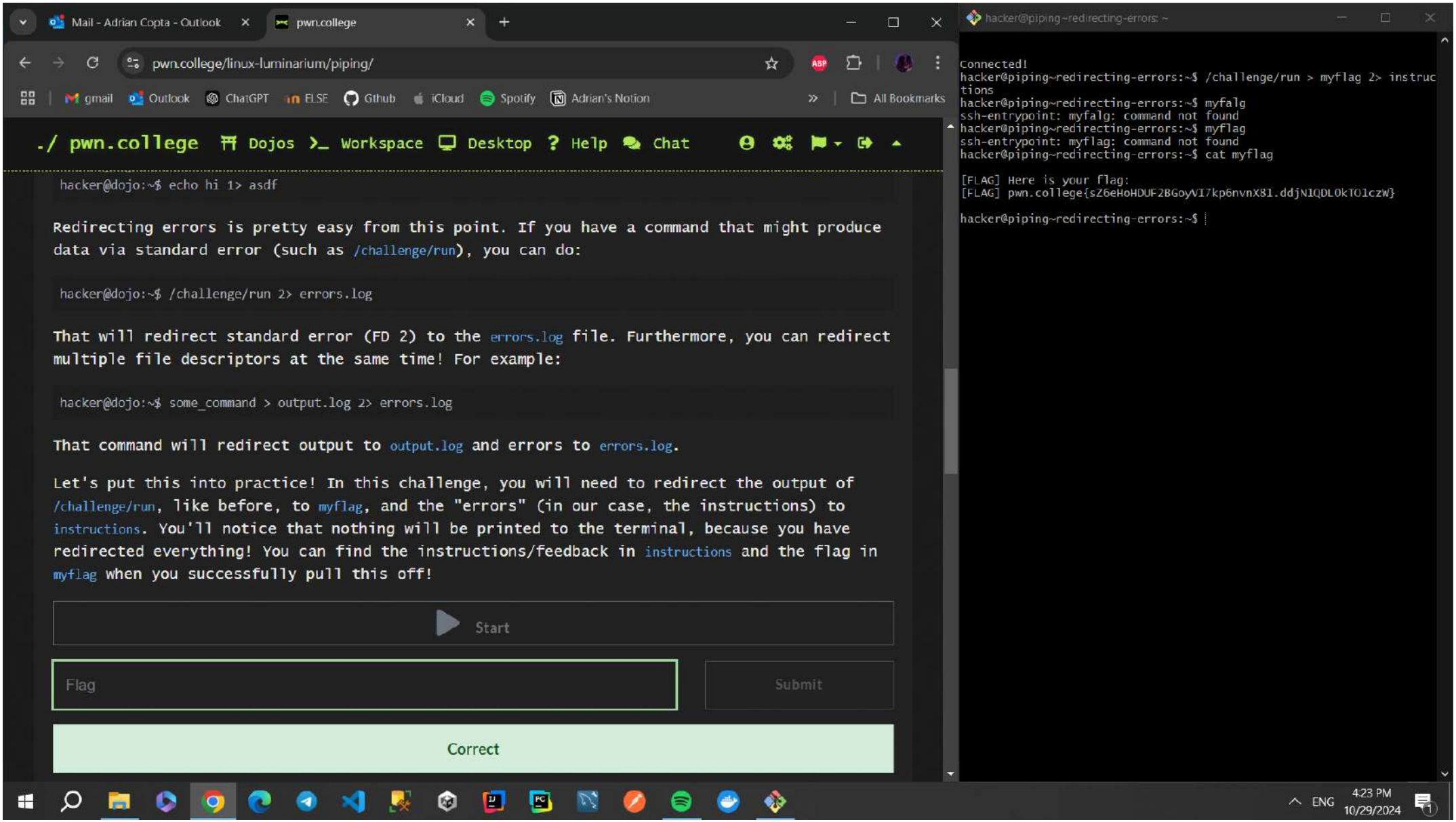

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Matching with []
3532 solves
Matching paths with []
3503 solves
Mixing globs
3249 solves
Now, let's put the previous levels together! We put a few happy, but diversely-named files in /challenge/files. Go cd there and, using the globbing you've learned, write a single, short (6 characters or less) glob that will match the files "challenging", "educational", and "pwning"!
Start
Flag
Submit
Correct
Exclusionary globbing
3327 solves
hacker@globbing~mixing-globs: /challenge/files
Connected!
hacker@globbing~mixing-globs:~$ /challenge/files/*ing
hacker@globbing~mixing-globs:~$ /challenge/run
Error: please run with a working directory of /challenge/files!
hacker@globbing~mixing-globs:~$ /challenge/run /challenge/files/*ing
Error: please run with a working directory of /challenge/files!
hacker@globbing~mixing-globs:~$ cd /challenge/files
ssh-entrypoint: $'\302\203cd': command not found
hacker@globbing~mixing-globs:~$ ls
Desktop f not-the-flag
hacker@globbing~mixing-globs:~$ cd /challenge/files
hacker@globbing~mixing-globs:/challenge/files$ ls
amazing fantastic kind pwning uplifting zesty
beautiful great laughing queenly victorious
challenging happy magical radiant wonderful
delightful incredible nice splendid xenial
educational jovial optimistic thrilling youthful
hacker@globbing~mixing-globs:/challenge/files$ challenge/run [a]
ssh-entrypoint: /challenge/run[a]: No such file or directory
hacker@globbing~mixing-globs:/challenge/files$ /challenge/run [a]
Your expansion did not expand to the requested files (challenging, educational,
pwning). Instead, it expanded to:
[a]
hacker@globbing~mixing-globs:/challenge/files$ /challenge/run [a]*
Your expansion did not expand to the requested files (challenging, educational,
pwning). Instead, it expanded to:
amazing
hacker@globbing~mixing-globs:/challenge/files$ /challenge/run [c]*
Your expansion did not expand to the requested files (challenging, educational,
pwning). Instead, it expanded to:
challenging
hacker@globbing~mixing-globs:/challenge/files$ /challenge/run [cep]*
You got it! Here is your flag!
pwn.college{sJNJww_cfnY7512B9ZJGbI8-Z6H.dVjM4QDL0kTO1czW}
hacker@globbing~mixing-globs:/challenge/files$
ENG
4:02 PM
10/29/2024

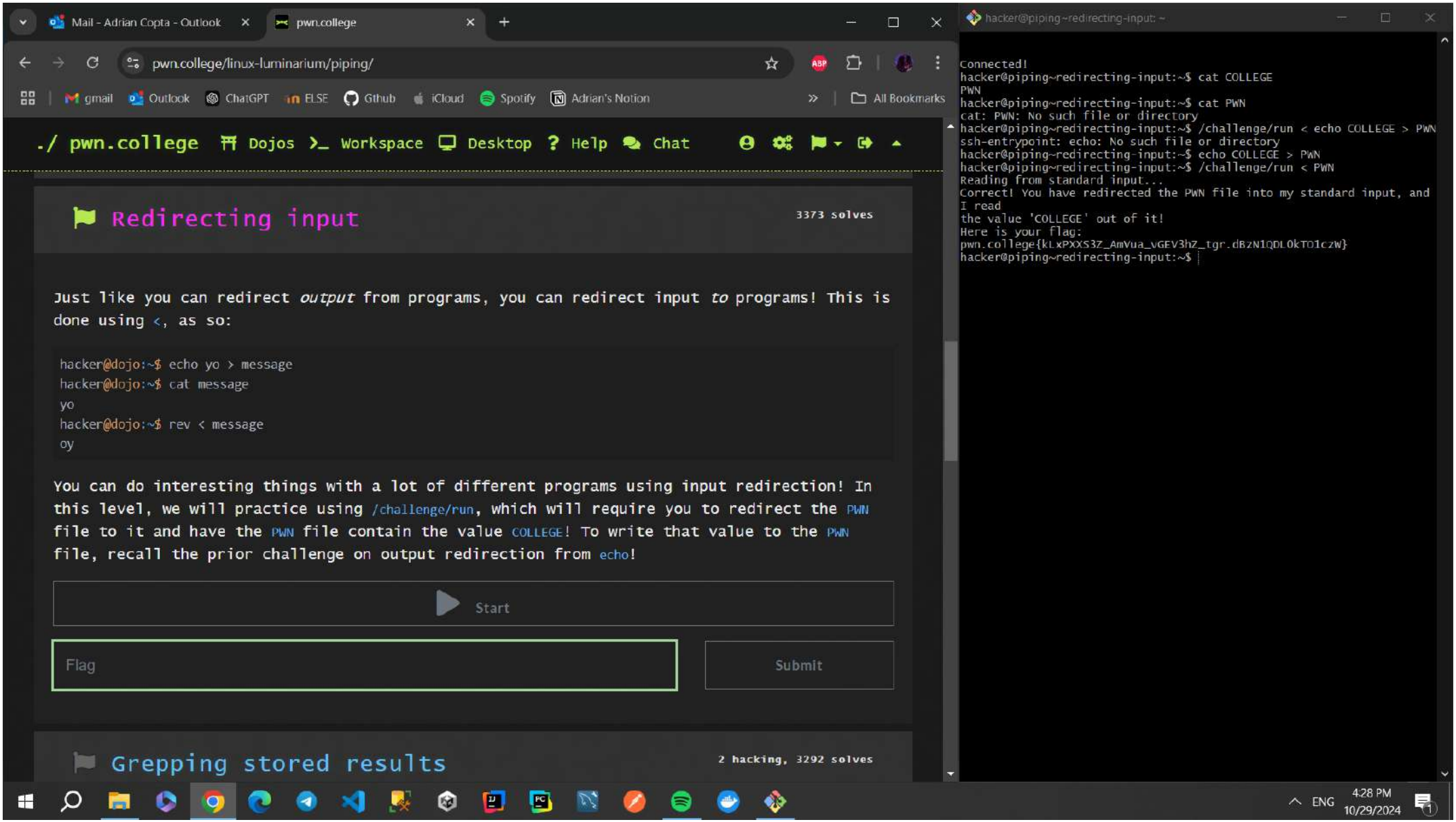

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/globbing/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ touch file_a
hacker@dojo:~$ touch file_b
hacker@dojo:~$ touch file_c
hacker@dojo:~$ ls
file_a file_b file_c
hacker@dojo:~$ echo Look: file_[!ab]
Look: file_c
hacker@dojo:~$ echo Look: file_[^ab]
Look: file_c
hacker@dojo:~$ echo Look: file_[ab]
Look: file_a file_b
Armed with this knowledge, go forth to /challenge/files and run /challenge/run with all files that don't start with p, w, or n!
NOTE: The ! character has a different special meaning in bash when it's not the first character of a [] glob, so keep that in mind if things stop making sense! ^ does not have this problem, but is also not compatible with older shells.
Start
Flag
Submit
Correct
hacker@globbing-exclusionary-globbing: /challenge/files
Connected!
hacker@globbing-exclusionary-globbing:~$ cd /challenge/files
hacker@globbing-exclusionary-globbing:/challenge/files$ /challenge/run
[!pwn]
Your expansion did not expand to the requested files (amazing beautiful challenging delightful educational fantastic great happy incredible jovial kind laughing magical optimistic queenly radiant splendid thrilling uplifting victorious xenial youthful zesty).
Instead, it expanded to:
[!pwn]
hacker@globbing-exclusionary-globbing:/challenge/files$ ls
amazing fantastic kind pwning uplifting zesty
beautiful great laughing queenly victorious
challenging happy magical radiant wonderful
delightful incredible nice splendid xenial
educational jovial optimistic thrilling youthful
hacker@globbing-exclusionary-globbing:/challenge/files$ /challenge/run
[^pwn]
Your expansion did not expand to the requested files (amazing beautiful challenging delightful educational fantastic great happy incredible jovial kind laughing magical optimistic queenly radiant splendid thrilling uplifting victorious xenial youthful zesty).
Instead, it expanded to:
[^pwn]
hacker@globbing-exclusionary-globbing:/challenge/files$ /challenge/run
[^pwn]*
You got it! Here is your flag!
pwn.college{cSHGQh5tqYWwgLW3v_rsl_rSufO.dZjM4QDL0kTO1czW}
hacker@globbing-exclusionary-globbing:/challenge/files$
ENG
4:06 PM
10/29/2024

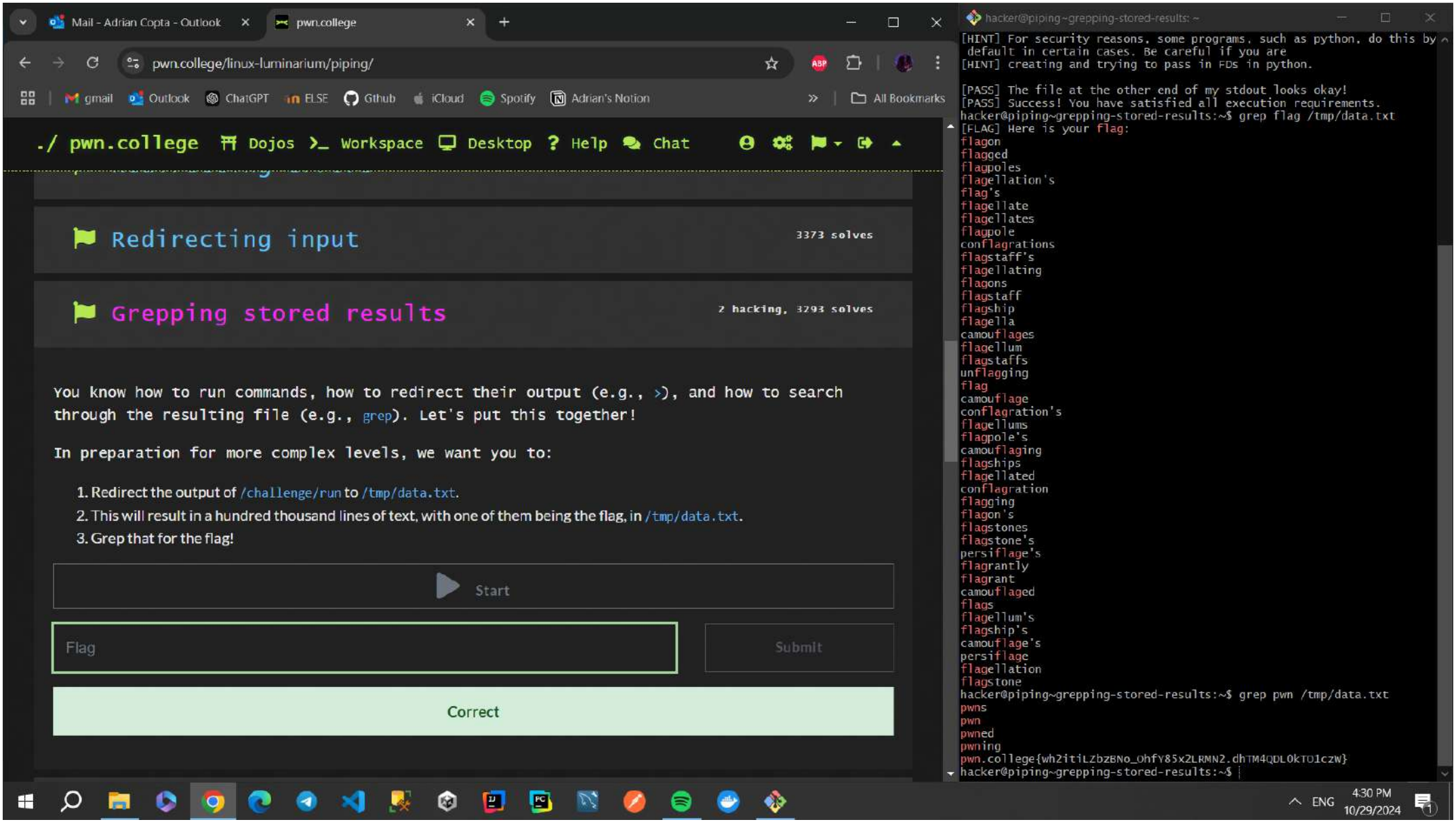

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Redirecting output
3509 solves
First, let's look at redirect stdout to files. You can accomplish this with the > character, as so:
hacker@dojo:~$ echo hi > asdf
This will redirect the output of echo hi (which will be hi) to the file asdf. You can then use a program such as cat to output this file:
hacker@dojo:~$ cat asdf
hi
In this challenge, you must use this input redirection to write the word PWN (all uppercase) to the filename COLLEGE (all uppercase).
Start
Flag
Submit
Correct
hacker@piping~redirecting-output: ~
Connected!
hacker@piping~redirecting-output:~$ echo PWN > COLLEGE
Correct! You successfully redirected 'PWN' to the file 'COLLEGE'! Here is your
flag:
pwn.college{463AWgOX3TtykLS4legcdn2w2ce.dRjN1QDL0kTO1czW}
hacker@piping~redirecting-output:~$
ENG
4:11 PM
10/29/2024

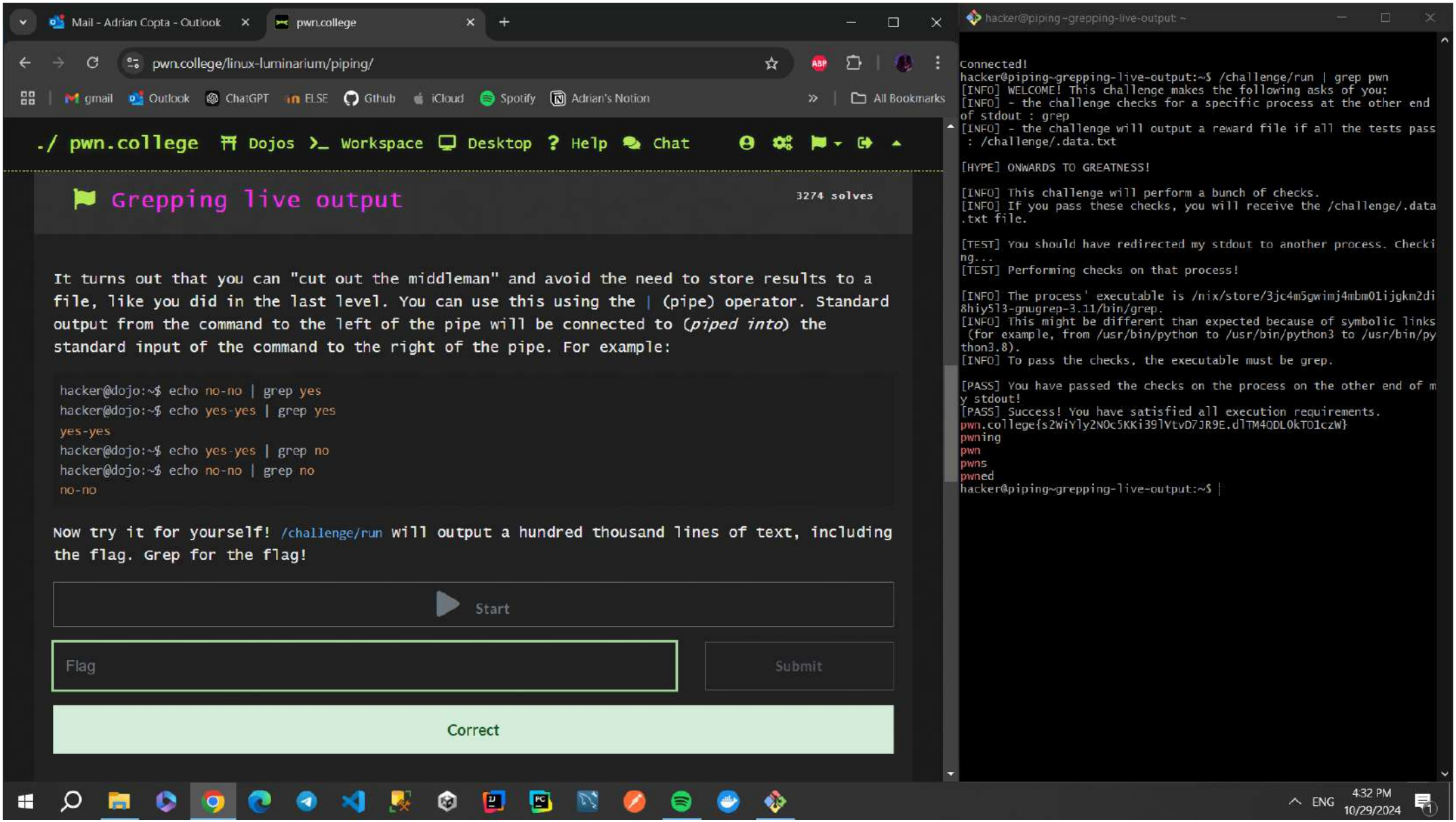

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Redirecting output
3509 solves
Redirecting more output
1 hacking, 3481 solves
Aside from redirecting the output of echo, you can, of course, redirect the output of any command. In this level, /challenge/run will once more give you a flag, but only if you redirect its output to the file myflag. Your flag will, of course, end up in the myflag file!
You'll notice that /challenge/run will still happily print to your terminal, despite you redirecting stdout. That's because it communicates its instructions and feedback over standard error, and only prints the flag over standard out!
Start
Flag
Submit
Correct
Appending output
1 hacking, 3361 solves
hacker@piping~redirecting-more-output ~
Connected!
hacker@piping~redirecting-more-output:~$ /challenge/run > myfalg
[INFO] WELCOME! This challenge makes the following asks of you:
[INFO] - the challenge will check that output is redirected to a specific file path : myflag
[INFO] - the challenge will output a reward file if all the tests pass : /flag
[HYPE] ONWARDS TO GREATNESS!
[INFO] This challenge will perform a bunch of checks.
[INFO] If you pass these checks, you will receive the /flag file.
[TEST] You should have redirected my stdout to a file called myflag. Checking...
[FAIL] You did not satisfy all the execution requirements.
[FAIL] Specifically, you must fix the following issue:
[FAIL] You have redirected the wrong file for stdout (/home/hacker/myfalg instead of myflag).
hacker@piping~redirecting-more-output:~$ /challenge/run > myflag
[INFO] WELCOME! This challenge makes the following asks of you:
[INFO] - the challenge will check that output is redirected to a specific file path : myflag
[INFO] - the challenge will output a reward file if all the tests pass : /flag
[HYPE] ONWARDS TO GREATNESS!
[INFO] This challenge will perform a bunch of checks.
[INFO] If you pass these checks, you will receive the /flag file.
[TEST] You should have redirected my stdout to a file called myflag. Checking...
[PASS] The file at the other end of my stdout looks okay!
[PASS] Success! You have satisfied all execution requirements.
hacker@piping~redirecting-more-output:~$ cat myflag
[FLAG] Here is your flag:
[FLAG] pwn.college{YB9ECo2xxSpfBImeiZm7JccUYQu.dVjN1QDL0kTO1czW}
hacker@piping~redirecting-more-output:~$
ENG
4:13 PM
10/29/2024

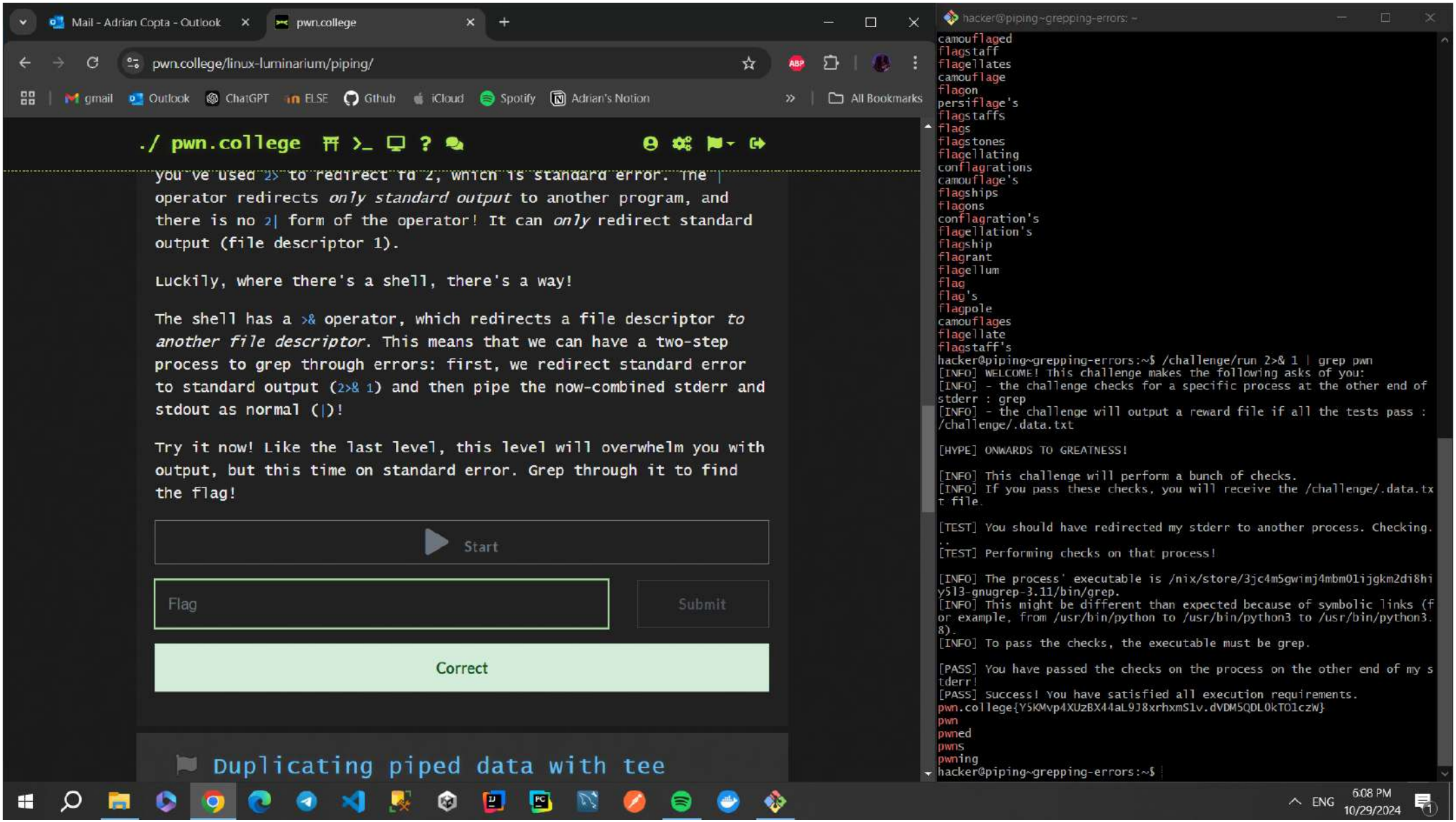

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
contents.
You can redirect input in append mode using >> instead of >, as so:
hacker@dojo:~$ echo pwn > outfile
hacker@dojo:~$ echo college >> outfile
hacker@dojo:~$ cat outfile
pwn
college
hacker@dojo:$
To practice, run /challenge/run with an append-mode redirect of the output to the file /home/hacker/the-flag. The practice will write the first half of the flag to the file, and the second half to stdout if stdout is redirected to the file. If you properly redirect in append-mode, the second half will be appended to the first, but if you redirect in truncation mode (>), the second half will overwrite the first and you won't get the flag!
Go for it now!
Start
Flag
Submit
Correct
hacker@piping~appending-output: ~
[HINT] File descriptors are inherited from the parent, unless the FD_CLOEXEC is set by the parent on the file descriptor.
[HINT] For security reasons, some programs, such as python, do this by default in certain cases. Be careful if you are
[HINT] creating and trying to pass in FDs in python.
[FAIL] You did not satisfy all the execution requirements.
[FAIL] Specifically, you must fix the following issue:
[FAIL] You have redirected the wrong file for stdout (/home/hacker/the-falg instead of /home/hacker/the-flag).
hacker@piping~appending-output:~$ /challenge/run >> /home/hacker/the-flag
[INFO] WELCOME! This challenge makes the following asks of you:
[INFO] - the challenge will check that output is redirected to a specific file path : /home/hacker/the-flag
[HYPE] ONWARDS TO GREATNESS!
[INFO] This challenge will perform a bunch of checks.
[INFO] Good luck!
[TEST] You should have redirected my stdout to a file called /home/hacker/the-flag. Checking...
[HINT] File descriptors are inherited from the parent, unless the FD_CLOEXEC is set by the parent on the file descriptor.
[HINT] For security reasons, some programs, such as python, do this by default in certain cases. Be careful if you are
[HINT] creating and trying to pass in FDs in python.
[PASS] The file at the other end of my stdout looks okay!
[PASS] Success! You have satisfied all execution requirements.
I will write the flag in two parts to the file /home/hacker/the-flag! I'll do
the first write directly to the file, and the second write, I'll do to stdout
(if it's pointing at the file). If you redirect the output in append mode, the
second write will append to (rather than overwrite) the first write, and you'll
get the whole flag!
hacker@piping~appending-output:~$ cat /home/hacker/the-flag
\|/ This is the first half:
pwn.college{sZTSjlGdhyOABbPZ66mNVx2IqyY.ddDM5QDL0kTO1czW}
that is the second half /|\
If you only see the second half above, you redirected in *truncate* mode (>)
rather than *append* mode (>>), and so the write of the second half to stdout
overwrote the initial write of the first half directly to the file. Try append
mode!
hacker@piping~appending-output:~$
ENG
4:19 PM
10/29/2024

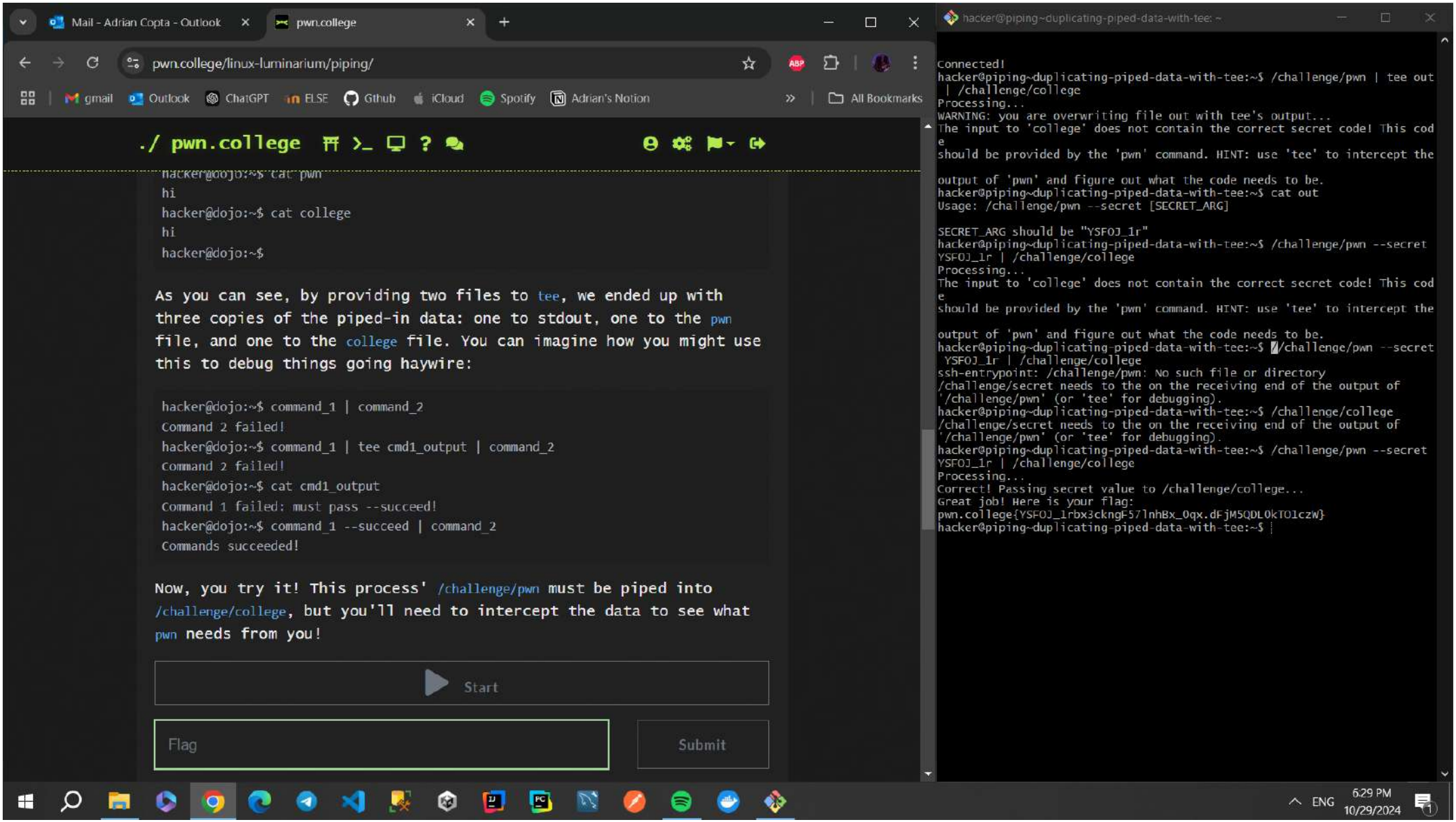

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ echo hi 1> asdf
Redirecting errors is pretty easy from this point. If you have a command that might produce data via standard error (such as /challenge/run), you can do:
hacker@dojo:~$ /challenge/run 2> errors.log
That will redirect standard error (FD 2) to the errors.log file. Furthermore, you can redirect multiple file descriptors at the same time! For example:
hacker@dojo:~$ some_command > output.log 2> errors.log
That command will redirect output to output.log and errors to errors.log.
Let's put this into practice! In this challenge, you will need to redirect the output of /challenge/run, like before, to myflag, and the "errors" (in our case, the instructions) to instructions. You'll notice that nothing will be printed to the terminal, because you have redirected everything! You can find the instructions/feedback in instructions and the flag in myflag when you successfully pull this off!
Start
Flag
Submit
Correct
hacker@piping~redirecting-errors: ~
Connected!
hacker@piping~redirecting-errors:~$ /challenge/run > myflag 2> instructions
hacker@piping~redirecting-errors:~$ myfalg
ssh-entrypoint: myfalg: command not found
hacker@piping~redirecting-errors:~$ myflag
ssh-entrypoint: myflag: command not found
hacker@piping~redirecting-errors:~$ cat myflag
[FLAG] Here is your flag:
[FLAG] pwn.college{sZ6eHoHDUF2BGoyVI7kp6nvnX81.ddjN1QDL0kTO1czW}
hacker@piping~redirecting-errors:~$
ENG
4:23 PM
10/29/2024

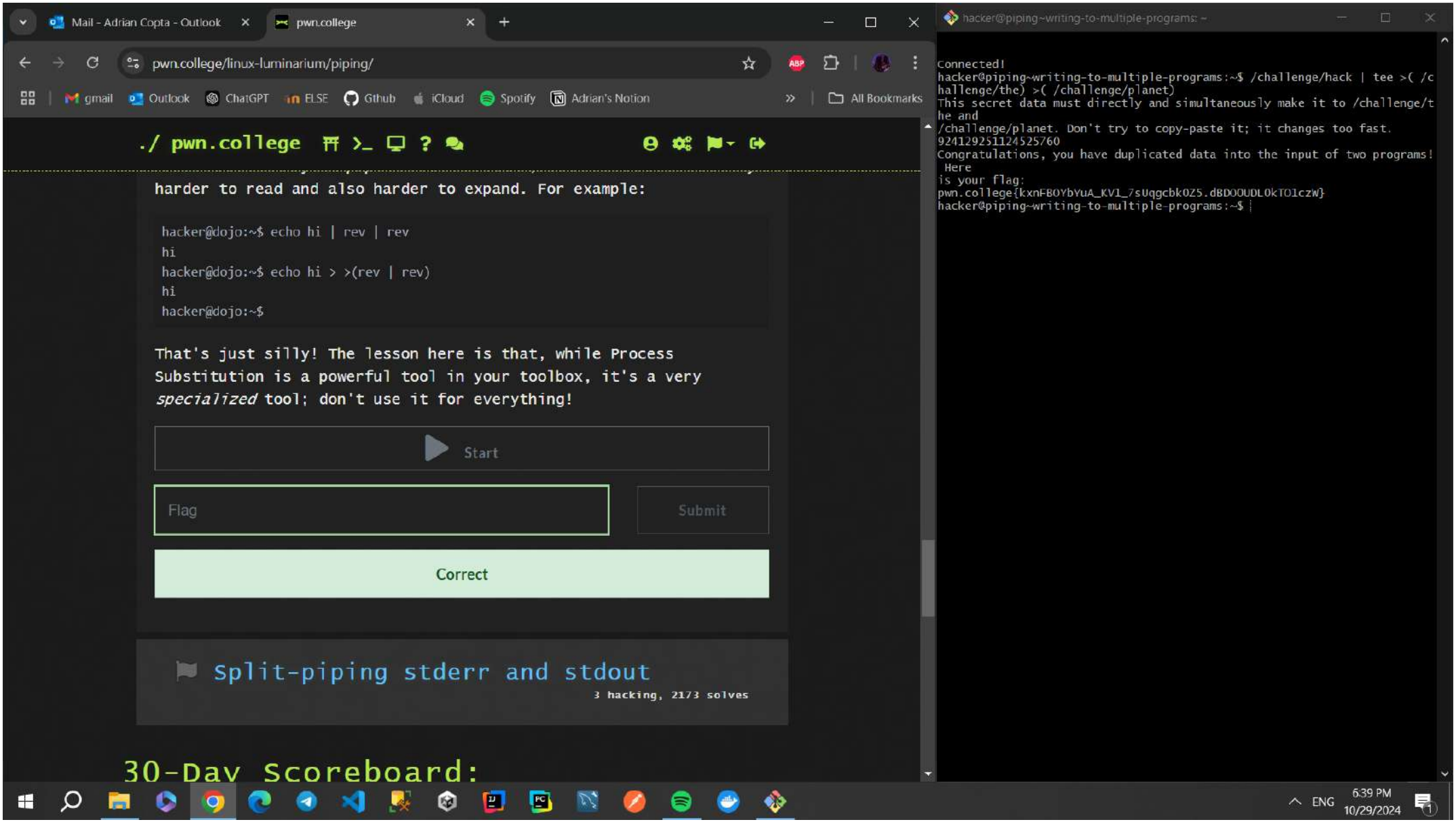

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Redirecting input
3373 solves
Just like you can redirect output from programs, you can redirect input to programs! This is done using <, as so:
hacker@dojo:~$ echo yo > message
hacker@dojo:~$ cat message
yo
hacker@dojo:~$ rev < message
oy
You can do interesting things with a lot of different programs using input redirection! In this level, we will practice using /challenge/run, which will require you to redirect the PWN file to it and have the PWN file contain the value COLLEGE! To write that value to the PWN file, recall the prior challenge on output redirection from echo!
Start
Flag
Submit
Grepping stored results
2 hacking, 3292 solves
hacker@piping~redirecting-input: ~
Connected!
hacker@piping~redirecting-input:~$ cat COLLEGE
PWN
hacker@piping~redirecting-input:~$ cat PWN
cat: PWN: No such file or directory
hacker@piping~redirecting-input:~$ /challenge/run < echo COLLEGE > PWN
ssh-entrypoint: echo: No such file or directory
hacker@piping~redirecting-input:~$ echo COLLEGE > PWN
hacker@piping~redirecting-input:~$ /challenge/run < PWN
Reading from standard input...
Correct! You have redirected the PWN file into my standard input, and I read
the value 'COLLEGE' out of it!
Here is your flag:
pwn.college{kLxPXXS3Z_AmVua_vGEV3hZ_tgr.dBzN1QDL0kTO1czW}
hacker@piping~redirecting-input:~$
ENG
4:28 PM
10/29/2024

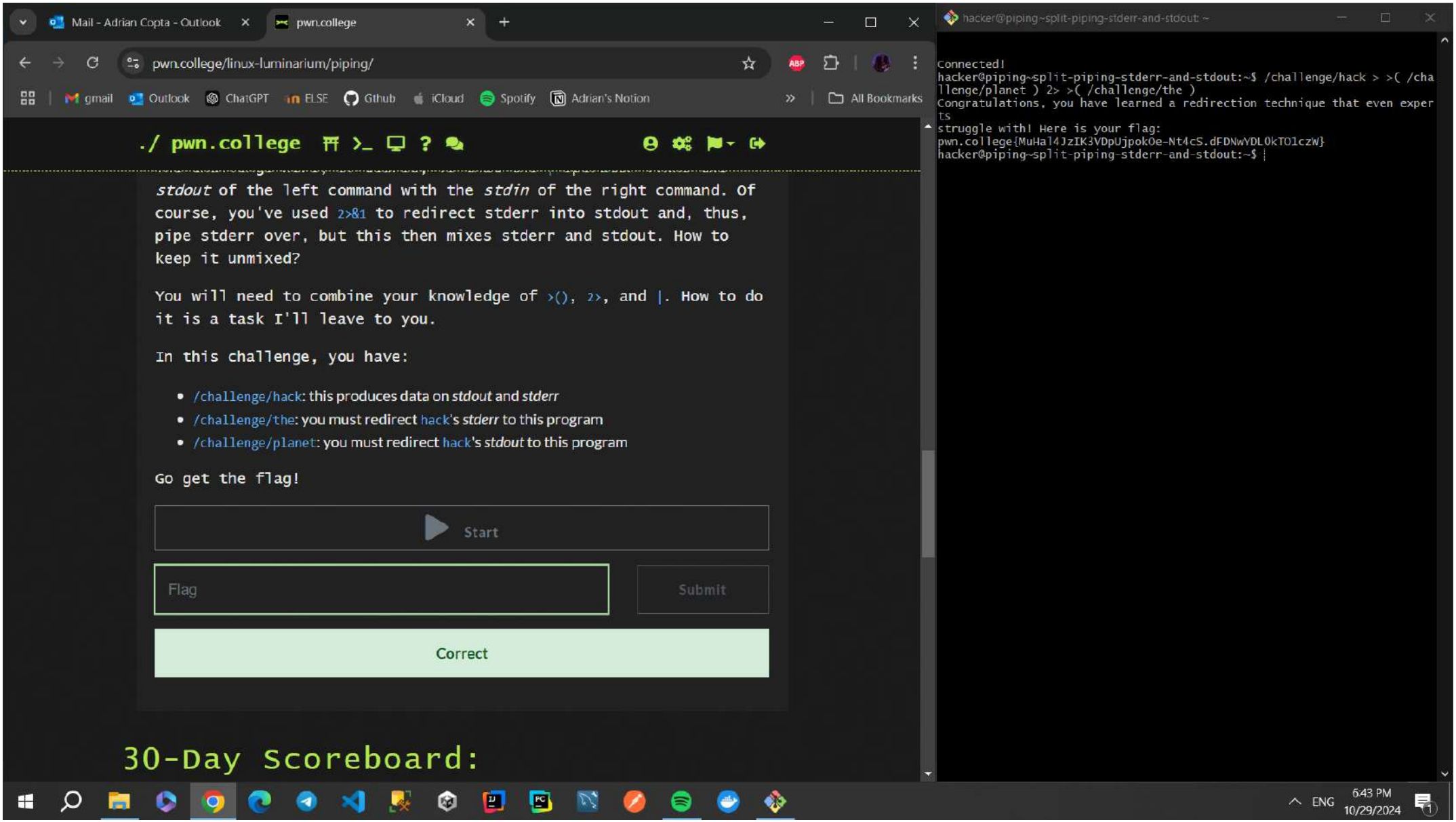

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Redirecting input
3373 solves
Grepping stored results
2 hacking, 3293 solves
You know how to run commands, how to redirect their output (e.g., >), and how to search through the resulting file (e.g., grep). Let's put this together!
In preparation for more complex levels, we want you to:
1. Redirect the output of /challenge/run to /tmp/data.txt.
2. This will result in a hundred thousand lines of text, with one of them being the flag, in /tmp/data.txt.
3. Grep that for the flag!
Start
Flag
Submit
Correct
hacker@piping~grepping-stored-results: ~
[HINT] For security reasons, some programs, such as python, do this by default in certain cases. Be careful if you are
[HINT] creating and trying to pass in FDs in python.
[PASS] The file at the other end of my stdout looks okay!
[PASS] Success! You have satisfied all execution requirements.
hacker@piping~grepping-stored-results:~$ grep flag /tmp/data.txt
[FLAG] Here is your flag:
flagon
flagged
flagpoles
flagellation's
flag's
flagellate
flagellates
flagpole
conflagrations
flagstaff's
flagellating
flagons
flagstaff
flagship
flagella
camouflages
flagellum
flagstaffs
unflagging
flag
camouflage
conflagration's
flagellums
flagpole's
camouflaging
flagships
flagellated
conflagration
flagging
flagon's
flagstones
flagstone's
persiflage's
flagrantly
flagrant
camouflaged
flags
flagellum's
flagship's
camouflage's
persiflage
flagellation
flagstone
hacker@piping~grepping-stored-results:~$ grep pwn /tmp/data.txt
pwns
pwn
pwned
pwning
pwn.college{wh2itiLZbzBNo_OhfY85x2LRMN2.dhTM4QDL0kTD1czW}
hacker@piping~grepping-stored-results:~$
ENG
4:30 PM
10/29/2024

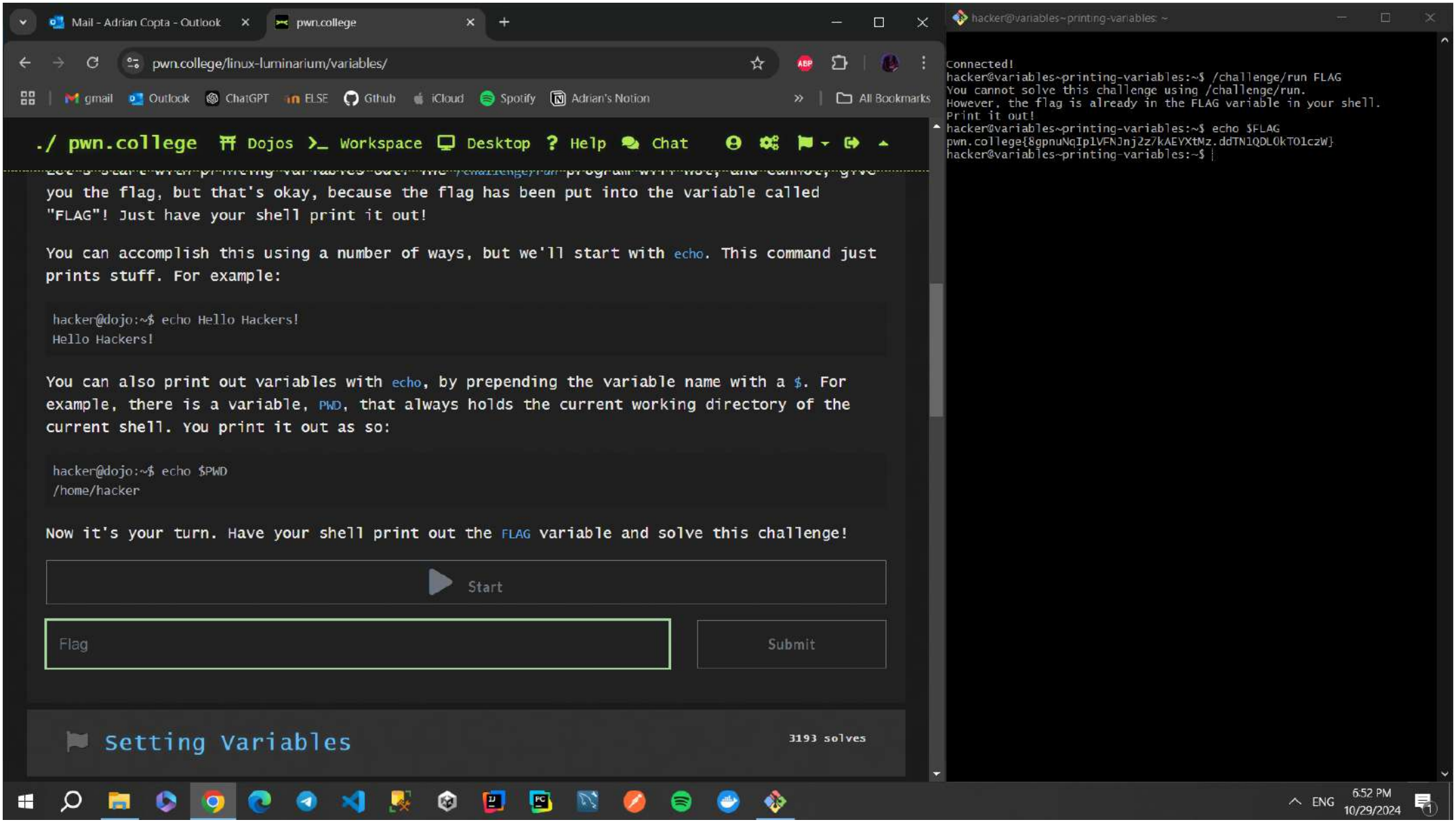

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Grepping live output
3274 solves
It turns out that you can "cut out the middleman" and avoid the need to store results to a file, like you did in the last level. You can use this using the | (pipe) operator. Standard output from the command to the left of the pipe will be connected to (piped into) the standard input of the command to the right of the pipe. For example:
hacker@dojo:~$ echo no-no | grep yes
hacker@dojo:~$ echo yes-yes | grep yes
yes-yes
hacker@dojo:~$ echo yes-yes | grep no
hacker@dojo:~$ echo no-no | grep no
no-no
Now try it for yourself! /challenge/run will output a hundred thousand lines of text, including the flag. Grep for the flag!
Start
Flag
Submit
Correct
hacker@piping~grepping-live-output: ~
Connected!
hacker@piping~grepping-live-output:~$ /challenge/run | grep pwn
[INFO] WELCOME! This challenge makes the following asks of you:
[INFO] - the challenge checks for a specific process at the other end of stdout : grep
[INFO] - the challenge will output a reward file if all the tests pass : /challenge/.data.txt
[HYPE] ONWARDS TO GREATNESS!
[INFO] This challenge will perform a bunch of checks.
[INFO] If you pass these checks, you will receive the /challenge/.data.txt file.
[TEST] You should have redirected my stdout to another process. Checking...
[TEST] Performing checks on that process!
[INFO] The process' executable is /nix/store/3jc4m5gwimj4mbm01ijgkm2di8hiy5l3-gnugrep-3.11/bin/grep.
[INFO] This might be different than expected because of symbolic links (for example, from /usr/bin/python to /usr/bin/python3 to /usr/bin/python3.8).
[INFO] To pass the checks, the executable must be grep.
[PASS] You have passed the checks on the process on the other end of my stdout!
[PASS] Success! You have satisfied all execution requirements.
pwn.college{s2WiYly2NOc5KKi39lYtvD7JR9E.dlTM4QDL0kTO1czW}
pwning
pwn
pwns
pwned
hacker@piping~grepping-live-output:~$
ENG
4:32 PM
10/29/2024

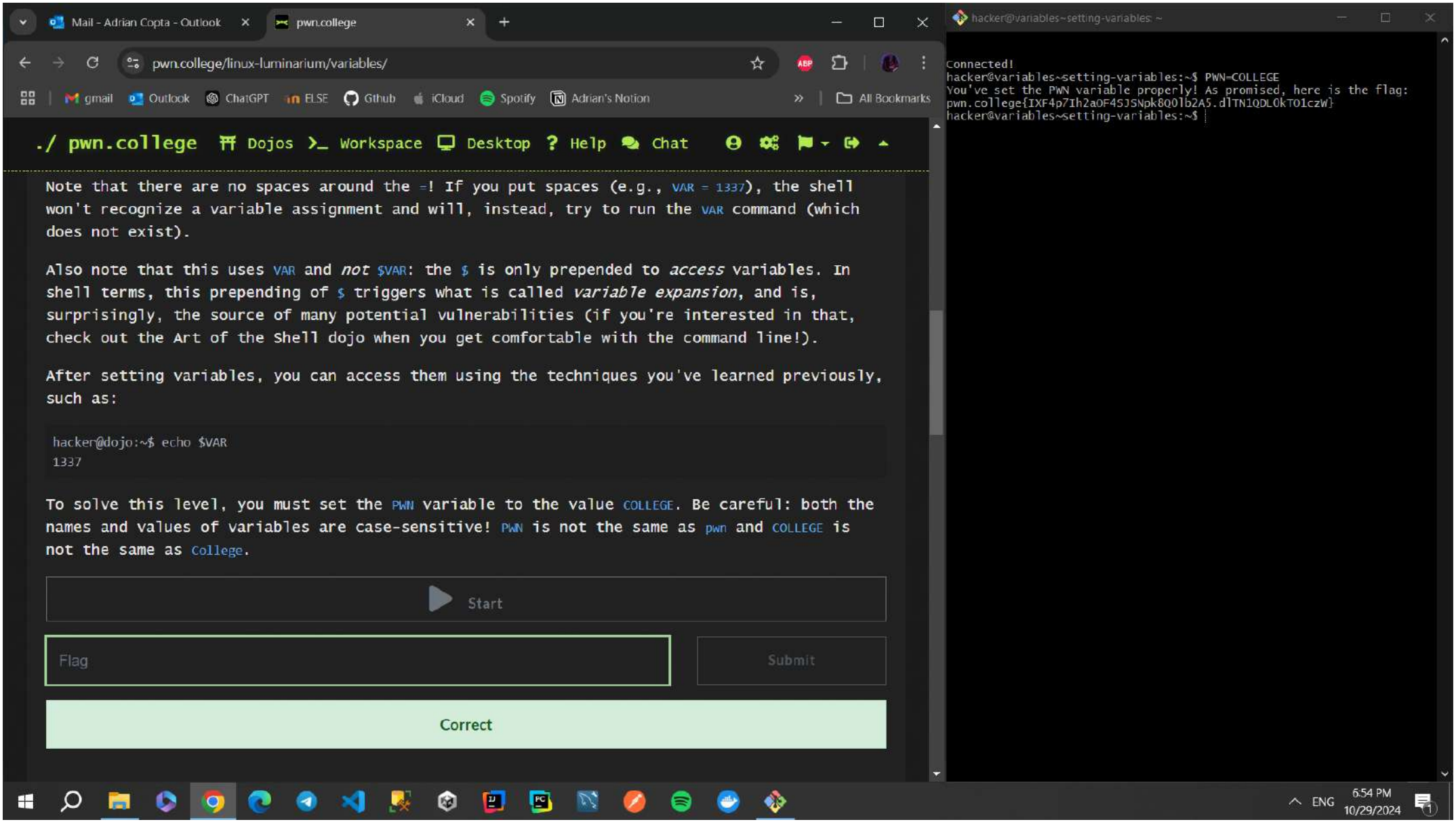

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
you've used 2> to redirect to 2, which is standard error. The | operator redirects only standard output to another program, and there is no 2| form of the operator! It can only redirect standard output (file descriptor 1).
Luckily, where there's a shell, there's a way!
The shell has a >& operator, which redirects a file descriptor to another file descriptor. This means that we can have a two-step process to grep through errors: first, we redirect standard error to standard output (2>& 1) and then pipe the now-combined stderr and stdout as normal (|)!
Try it now! Like the last level, this level will overwhelm you with output, but this time on standard error. Grep through it to find the flag!
Start
Flag
Submit
Correct
Duplicating piped data with tee
hacker@piping~grepping-errors: ~
camouflaged
flagstaff
flagellates
camouflage
flagon
persiflage's
flagstaffs
flags
flagstones
flagellating
conflagrations
camouflage's
flagships
flagons
conflagration's
flagellation's
flagship
flagrant
flagellum
flag
flag's
flagpole
camouflages
flagellate
flagstaff's
hacker@piping~grepping-errors:~$ /challenge/run 2>& 1 | grep pwn
[INFO] WELCOME! This challenge makes the following asks of you:
[INFO] - the challenge checks for a specific process at the other end of stderr : grep
[INFO] - the challenge will output a reward file if all the tests pass : /challenge/.data.txt
[HYPE] ONWARDS TO GREATNESS!
[INFO] This challenge will perform a bunch of checks.
[INFO] If you pass these checks, you will receive the /challenge/.data.txt file.
[TEST] You should have redirected my stderr to another process. Checking...
[TEST] Performing checks on that process!
[INFO] The process' executable is /nix/store/3jc4m5gwimj4mbm01ijgkm2di8hiy5l3-gnugrep-3.11/bin/grep.
[INFO] This might be different than expected because of symbolic links (for example, from /usr/bin/python to /usr/bin/python3 to /usr/bin/python3.8).
[INFO] To pass the checks, the executable must be grep.
[PASS] You have passed the checks on the process on the other end of my stderr!
[PASS] Success! You have satisfied all execution requirements.
pwn.college{Y5KMvp4XUzBX44aL9J8xrhxmS1v.dVDM5QDL0kTO1czW}
pwn
pwned
pwns
pwning
hacker@piping~grepping-errors:~$
ENG
6:08 PM
10/29/2024

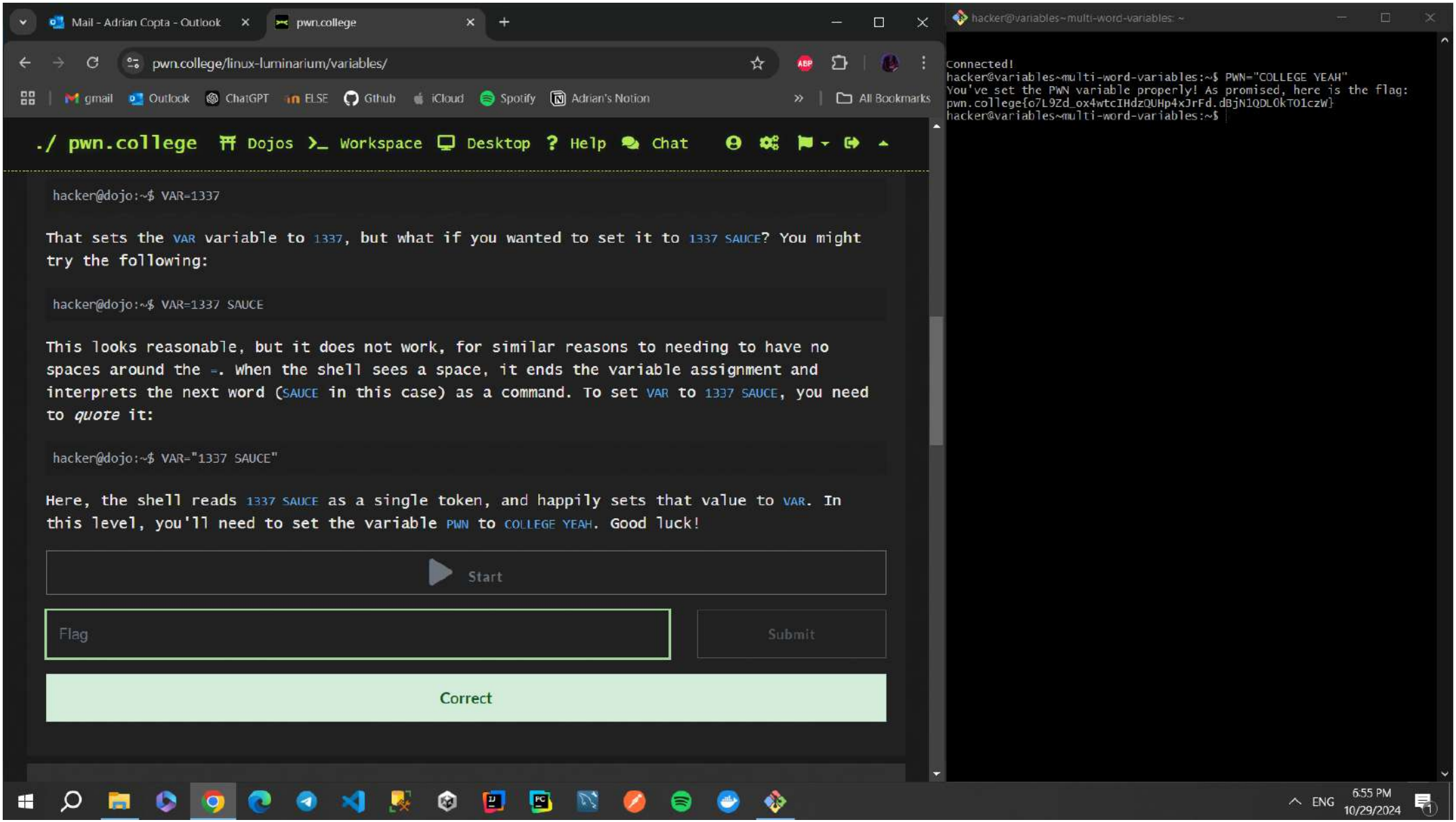

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
hacker@dojo:~$ cat pwn
hi
hacker@dojo:~$ cat college
hi
hacker@dojo:~$
As you can see, by providing two files to tee, we ended up with three copies of the piped-in data: one to stdout, one to the pwn file, and one to the college file. You can imagine how you might use this to debug things going haywire:
hacker@dojo:~$ command_1 | command_2
Command 2 failed!
hacker@dojo:~$ command_1 | tee cmd1_output | command_2
Command 2 failed!
hacker@dojo:~$ cat cmd1_output
Command 1 failed: must pass --succeed!
hacker@dojo:~$ command_1 --succeed | command_2
Commands succeeded!
Now, you try it! This process' /challenge/pwn must be piped into /challenge/college, but you'll need to intercept the data to see what pwn needs from you!
Start
Flag
Submit
hacker@piping~duplicating-piped-data-with-tee: ~
Connected!
hacker@piping~duplicating-piped-data-with-tee:~$ /challenge/pwn | tee out | /challenge/college
Processing...
WARNING: you are overwriting file out with tee's output...
The input to 'college' does not contain the correct secret code! This code
should be provided by the 'pwn' command. HINT: use 'tee' to intercept the
output of 'pwn' and figure out what the code needs to be.
hacker@piping~duplicating-piped-data-with-tee:~$ cat out
Usage: /challenge/pwn --secret [SECRET_ARG]
SECRET_ARG should be "YSFOJ_1r"
hacker@piping~duplicating-piped-data-with-tee:~$ /challenge/pwn --secret YSFOJ_1r | /challenge/college
Processing...
The input to 'college' does not contain the correct secret code! This code
should be provided by the 'pwn' command. HINT: use 'tee' to intercept the
output of 'pwn' and figure out what the code needs to be.
hacker@piping~duplicating-piped-data-with-tee:~$ /challenge/pwn --secret YSFOJ_1r | /challenge/college
ssh-entrypoint: /challenge/pwn: No such file or directory
/challenge/secret needs to the on the receiving end of the output of '/challenge/pwn' (or 'tee' for debugging).
hacker@piping~duplicating-piped-data-with-tee:~$ /challenge/college
/challenge/secret needs to the on the receiving end of the output of '/challenge/pwn' (or 'tee' for debugging).
hacker@piping~duplicating-piped-data-with-tee:~$ /challenge/pwn --secret YSFOJ_1r | /challenge/college
Processing...
Correct! Passing secret value to /challenge/college...
Great job! Here is your flag:
pwn.college{YSFOJ_1rbx3ckngF57lnhBx_0qx.dFjM5QDL0kTO1czW}
hacker@piping~duplicating-piped-data-with-tee:~$
ENG
6:29 PM
10/29/2024

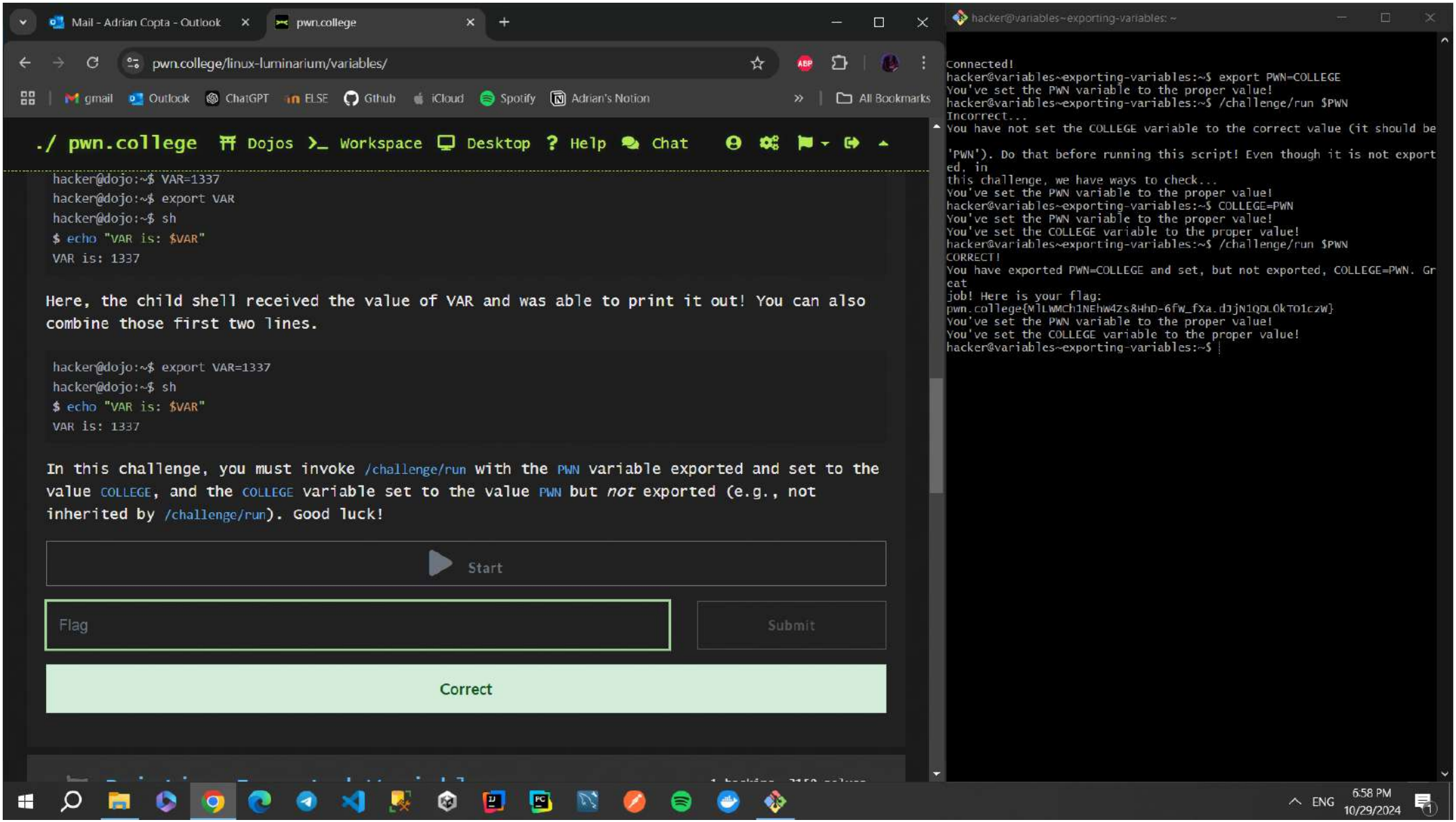

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
harder to read and also harder to expand. For example:
hacker@dojo:~$ echo hi | rev | rev
hi
hacker@dojo:~$ echo hi > >(rev | rev)
hi
hacker@dojo:~$
That's just silly! The lesson here is that, while Process Substitution is a powerful tool in your toolbox, it's a very specialized tool; don't use it for everything!
Start
Flag
Submit
Correct
Split-piping stderr and stdout
3 hacking, 2173 solves
30-Day Scoreboard:
hacker@piping-writing-to-multiple-programs: ~
Connected!
hacker@piping-writing-to-multiple-programs:~$ /challenge/hack | tee >(/challenge/the) >(/challenge/planet)
This secret data must directly and simultaneously make it to /challenge/the and
/challenge/planet. Don't try to copy-paste it; it changes too fast.
92412925112452576O
Congratulations, you have duplicated data into the input of two programs! Here
is your flag:
pwn.college{kxnFBOYbYuA_KV1_7sUqgcbk0Z5.dBDOOUDL0kTO1czW}
hacker@piping-writing-to-multiple-programs:~$
ENG
6:39 PM
10/29/2024

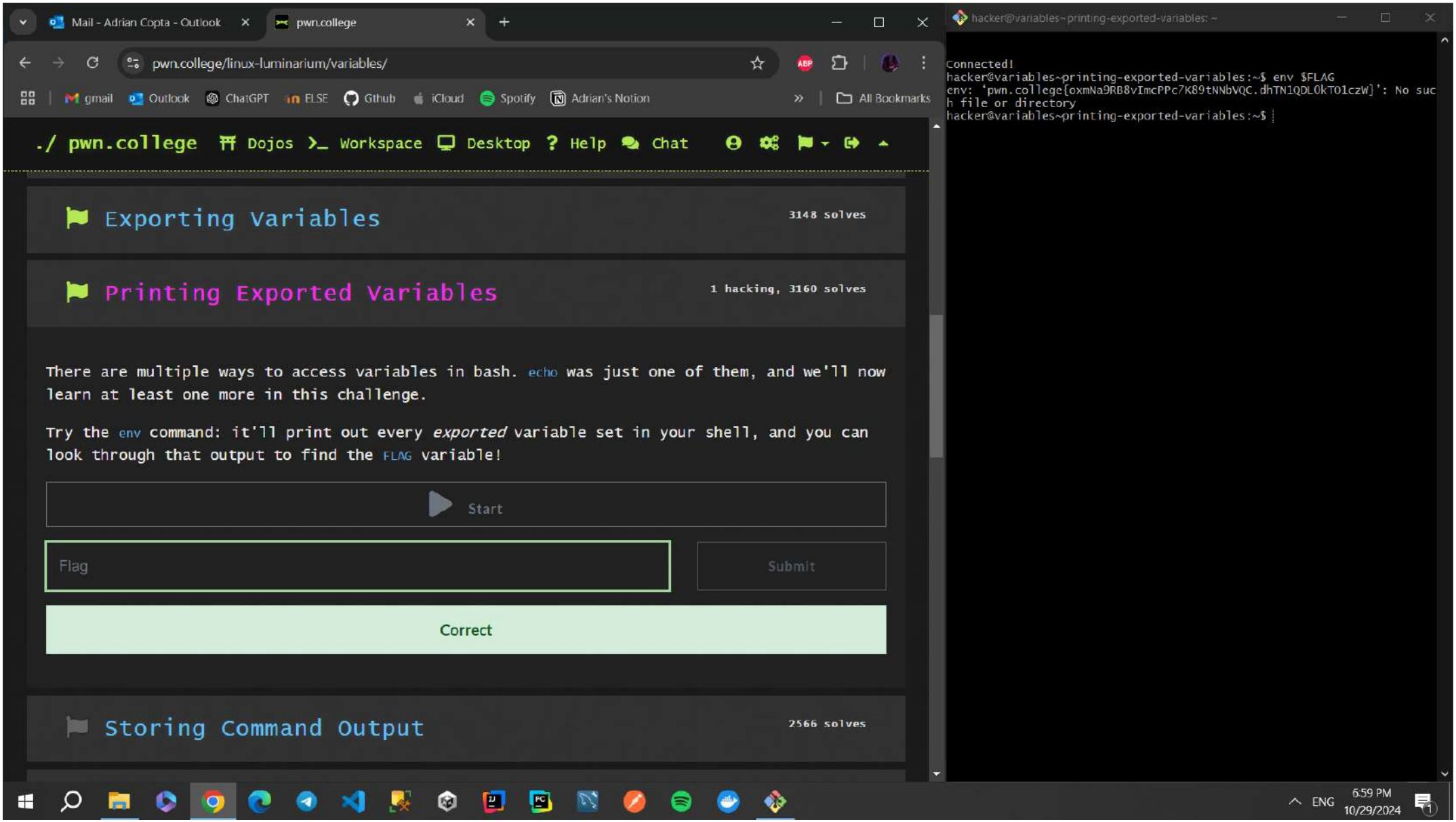

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/piping/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
stdout of the left command with the stdin of the right command. Of course, you've used 2>&1 to redirect stderr into stdout and, thus, pipe stderr over, but this then mixes stderr and stdout. How to keep it unmixed?
You will need to combine your knowledge of >(), 2>, and |. How to do it is a task I'll leave to you.
In this challenge, you have:
/challenge/hack: this produces data on stdout and stderr
/challenge/the: you must redirect hack's stderr to this program
/challenge/planet: you must redirect hack's stdout to this program
Go get the flag!
Start
Flag
Submit
Correct
30-Day Scoreboard:
hacker@piping~split-piping-stderr-and-stdout: ~
Connected!
hacker@piping~split-piping-stderr-and-stdout:~$ /challenge/hack > >(/challenge/planet) 2> >(/challenge/the)
Congratulations, you have learned a redirection technique that even experts
struggle with! Here is your flag:
pwn.college{MuHal4JzIK3VDpUjpokOe-Nt4cS.dFDNwYDL0kTO1czW}
hacker@piping~split-piping-stderr-and-stdout:~$
ENG
6:43 PM
10/29/2024

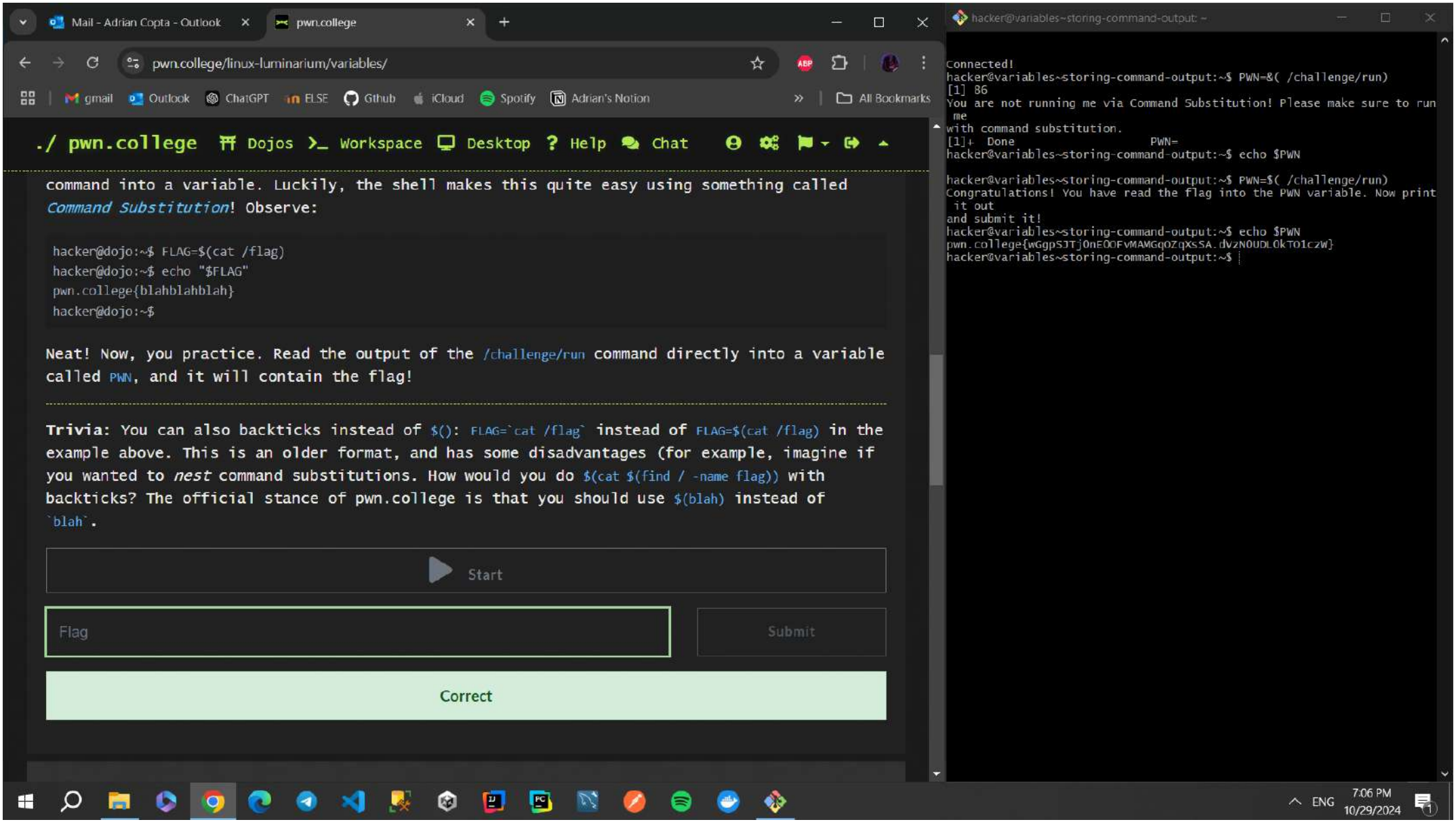

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
you the flag, but that's okay, because the flag has been put into the variable called "FLAG"! Just have your shell print it out!
You can accomplish this using a number of ways, but we'll start with echo. This command just prints stuff. For example:
hacker@dojo:~$ echo Hello Hackers!
Hello Hackers!
You can also print out variables with echo, by prepending the variable name with a $. For example, there is a variable, PWD, that always holds the current working directory of the current shell. You print it out as so:
hacker@dojo:~$ echo $PWD
/home/hacker
Now it's your turn. Have your shell print out the FLAG variable and solve this challenge!
Start
Flag
Submit
Setting Variables
3193 solves
hacker@variables~printing-variables: ~
Connected!
hacker@variables~printing-variables:~$ /challenge/run FLAG
You cannot solve this challenge using /challenge/run.
However, the flag is already in the FLAG variable in your shell.
Print it out!
hacker@variables~printing-variables:~$ echo $FLAG
pwn.college{8gpnuNqIp1VFNJnj2z/kAEYXtMz.ddTN1QDL0kTO1czW}
hacker@variables~printing-variables:~$
ENG
6:52 PM
10/29/2024

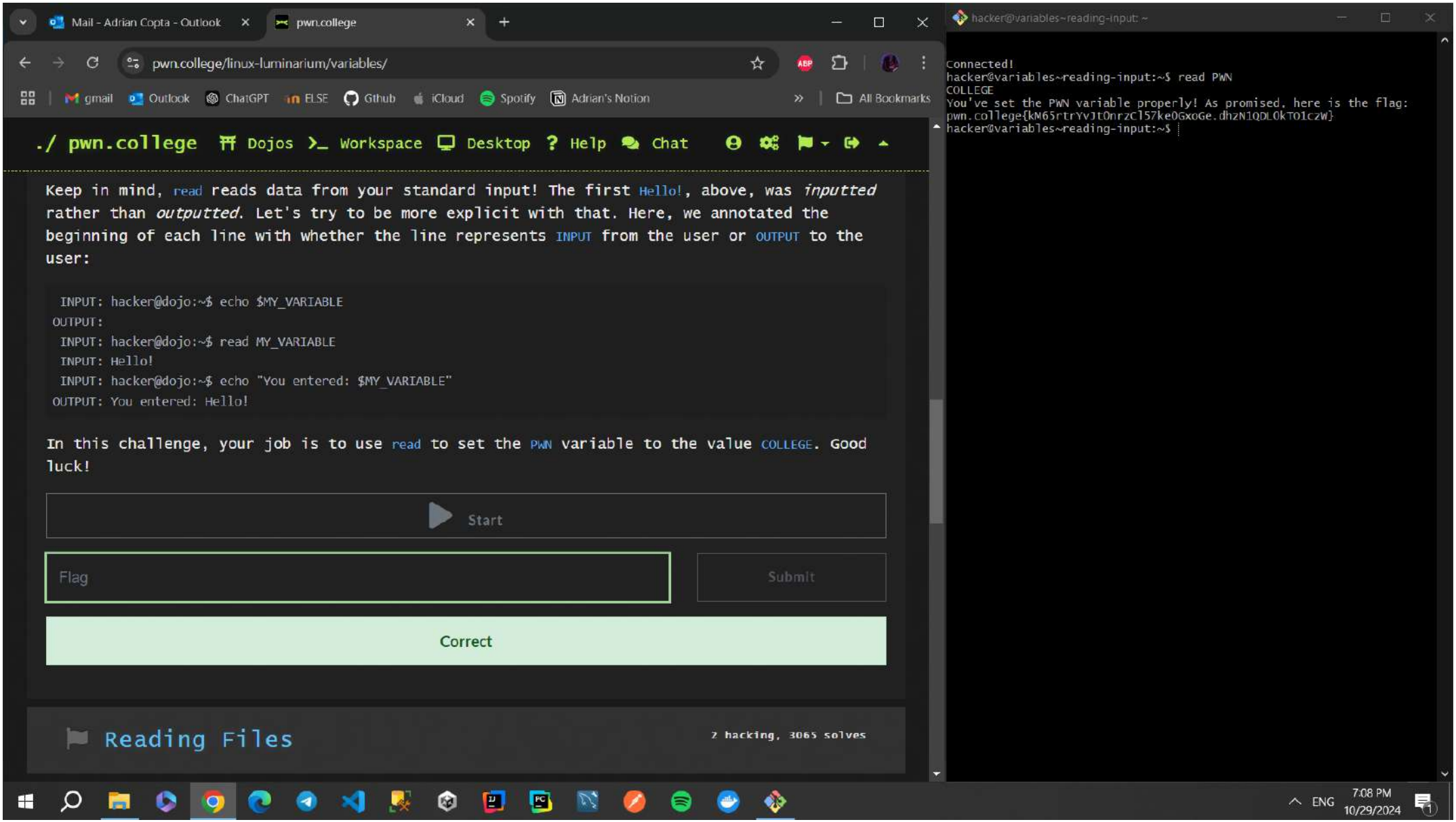

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Note that there are no spaces around the =! If you put spaces (e.g., VAR = 1337), the shell won't recognize a variable assignment and will, instead, try to run the VAR command (which does not exist).
Also note that this uses VAR and not $VAR: the $ is only prepended to access variables. In shell terms, this prepending of $ triggers what is called variable expansion, and is, surprisingly, the source of many potential vulnerabilities (if you're interested in that, check out the Art of the Shell dojo when you get comfortable with the command line!).
After setting variables, you can access them using the techniques you've learned previously, such as:
hacker@dojo:~$ echo $VAR
1337
To solve this level, you must set the PWN variable to the value COLLEGE. Be careful: both the names and values of variables are case-sensitive! PWN is not the same as pwn and COLLEGE is not the same as College.
Start
Flag
Submit
Correct
hacker@variables~setting-variables: ~
Connected!
hacker@variables~setting-variables:~$ PWN=COLLEGE
You've set the PWN variable properly! As promised, here is the flag:
pwn.college{IXF4p7Ih2aOF45J5Npk8Q0lb2A5.dlTN1QDL0kTO1czW}
hacker@variables~setting-variables:~$
ENG
6:54 PM
10/29/2024

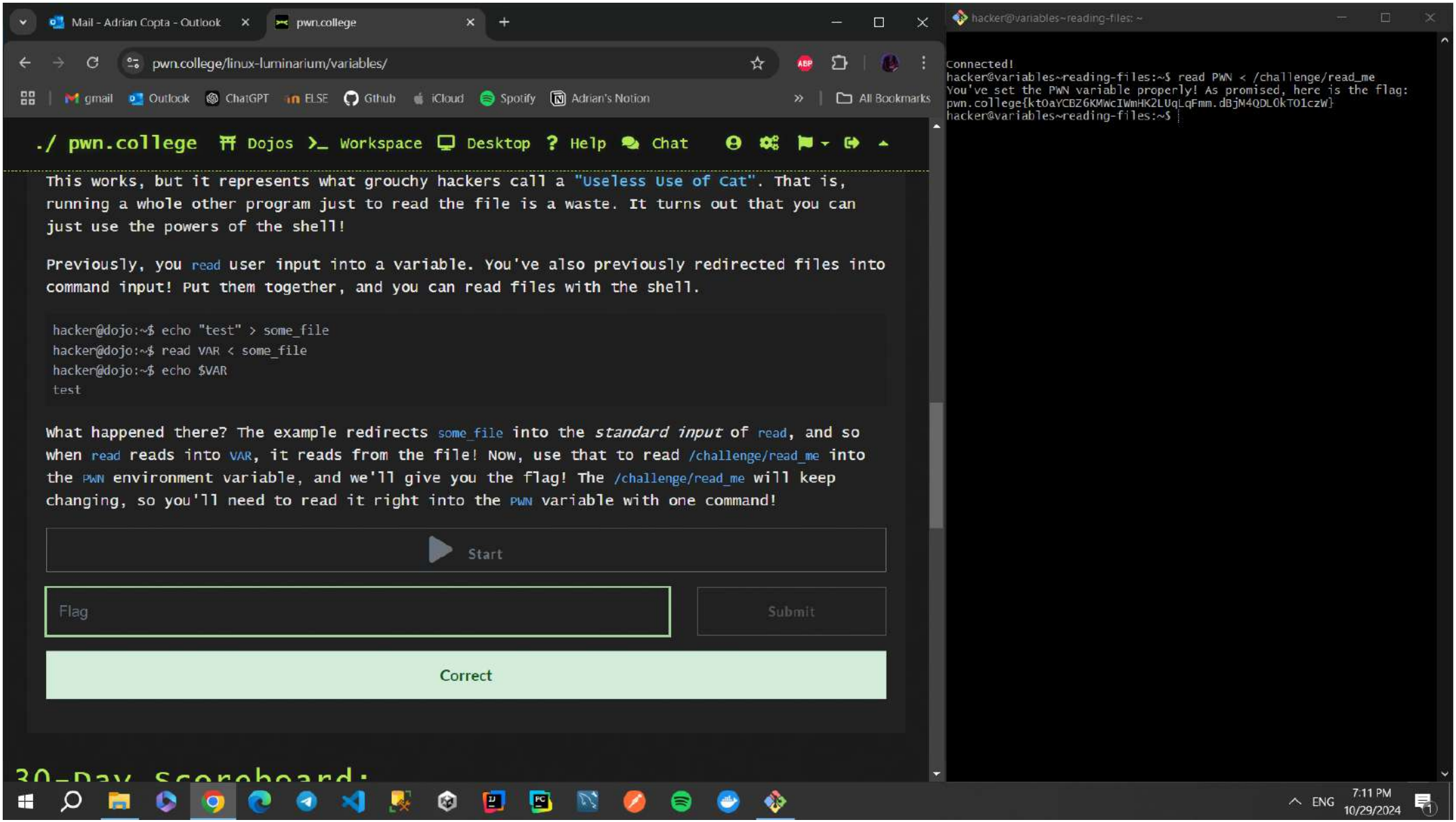

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ VAR=1337
That sets the VAR variable to 1337, but what if you wanted to set it to 1337 SAUCE? You might try the following:
hacker@dojo:~$ VAR=1337 SAUCE
This looks reasonable, but it does not work, for similar reasons to needing to have no spaces around the =. When the shell sees a space, it ends the variable assignment and interprets the next word (SAUCE in this case) as a command. To set VAR to 1337 SAUCE, you need to quote it:
hacker@dojo:~$ VAR="1337 SAUCE"
Here, the shell reads 1337 SAUCE as a single token, and happily sets that value to VAR. In this level, you'll need to set the variable PWN to COLLEGE YEAH. Good luck!
Start
Flag
Submit
Correct
hacker@variables~multi-word-variables: ~
Connected!
hacker@variables~multi-word-variables:~$ PWN="COLLEGE YEAH"
You've set the PWN variable properly! As promised, here is the flag:
pwn.college{o7L9Zd_ox4wtcIHdzQUHp4xJrFd.dBjN1QDLOkTO1czW}
hacker@variables~multi-word-variables:~$
ENG
6:55 PM
10/29/2024

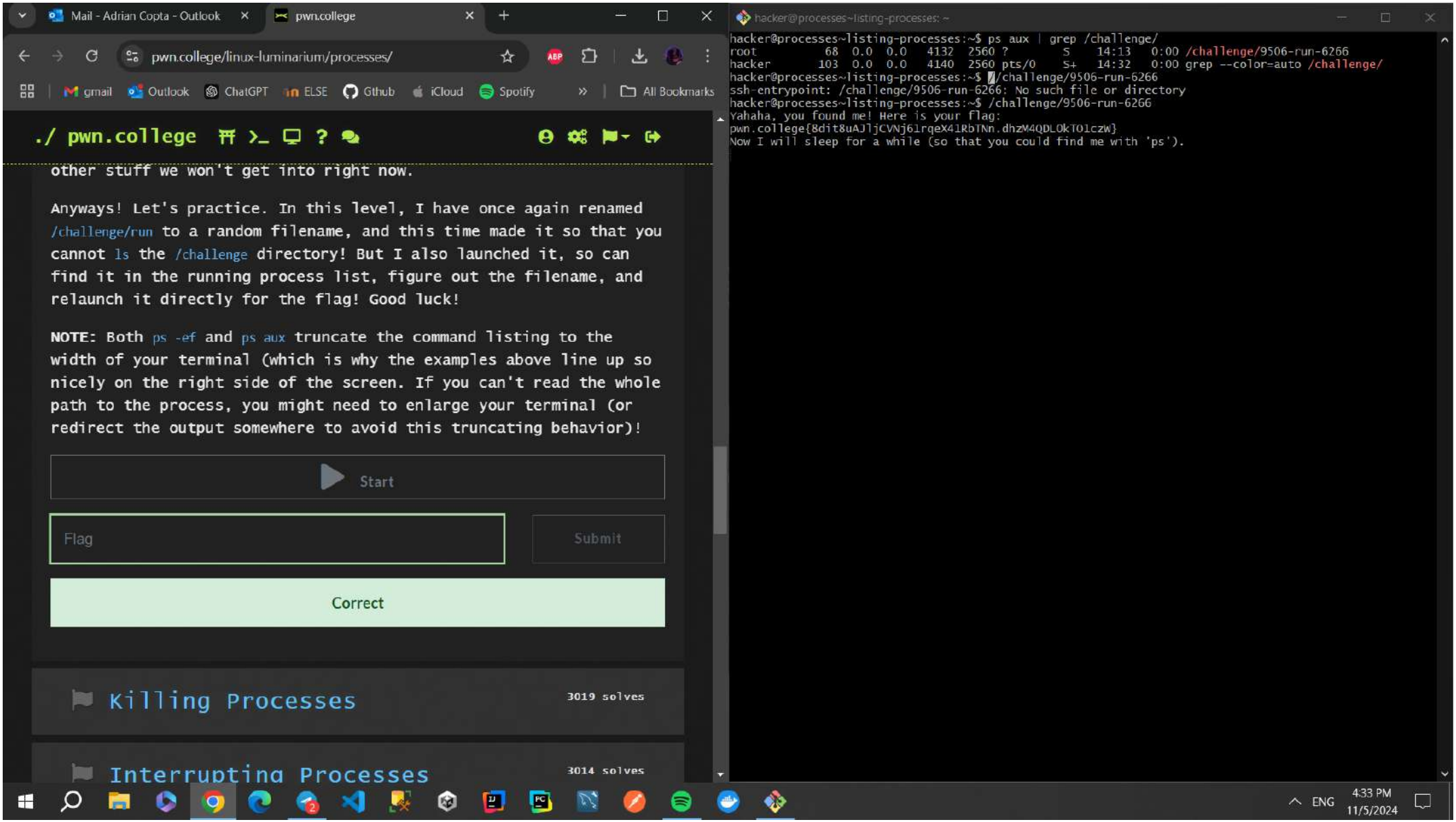

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ VAR=1337
hacker@dojo:~$ export VAR
hacker@dojo:~$ sh
$ echo "VAR is: $VAR"
VAR is: 1337
Here, the child shell received the value of VAR and was able to print it out! You can also combine those first two lines.
hacker@dojo:~$ export VAR=1337
hacker@dojo:~$ sh
$ echo "VAR is: $VAR"
VAR is: 1337
In this challenge, you must invoke /challenge/run with the PWN variable exported and set to the value COLLEGE, and the COLLEGE variable set to the value PWN but not exported (e.g., not inherited by /challenge/run). Good luck!
Start
Flag
Submit
Correct
hacker@variables~exporting-variables: ~
Connected!
hacker@variables~exporting-variables:~$ export PWN=COLLEGE
You've set the PWN variable to the proper value!
hacker@variables~exporting-variables:~$ /challenge/run $PWN
Incorrect...
You have not set the COLLEGE variable to the correct value (it should be 'PWN'). Do that before running this script! Even though it is not exported, in this challenge, we have ways to check...
You've set the PWN variable to the proper value!
hacker@variables~exporting-variables:~$ COLLEGE=PWN
You've set the PWN variable to the proper value!
You've set the COLLEGE variable to the proper value!
hacker@variables~exporting-variables:~$ /challenge/run $PWN
CORRECT!
You have exported PWN=COLLEGE and set, but not exported, COLLEGE=PWN. Great job! Here is your flag:
pwn.college{MlLWMCh1NEhw4Zs8HhD-6fW_fXa.dDjN1QDLOkTO1czW}
You've set the PWN variable to the proper value!
You've set the COLLEGE variable to the proper value!
hacker@variables~exporting-variables:~$
ENG
6:58 PM
10/29/2024

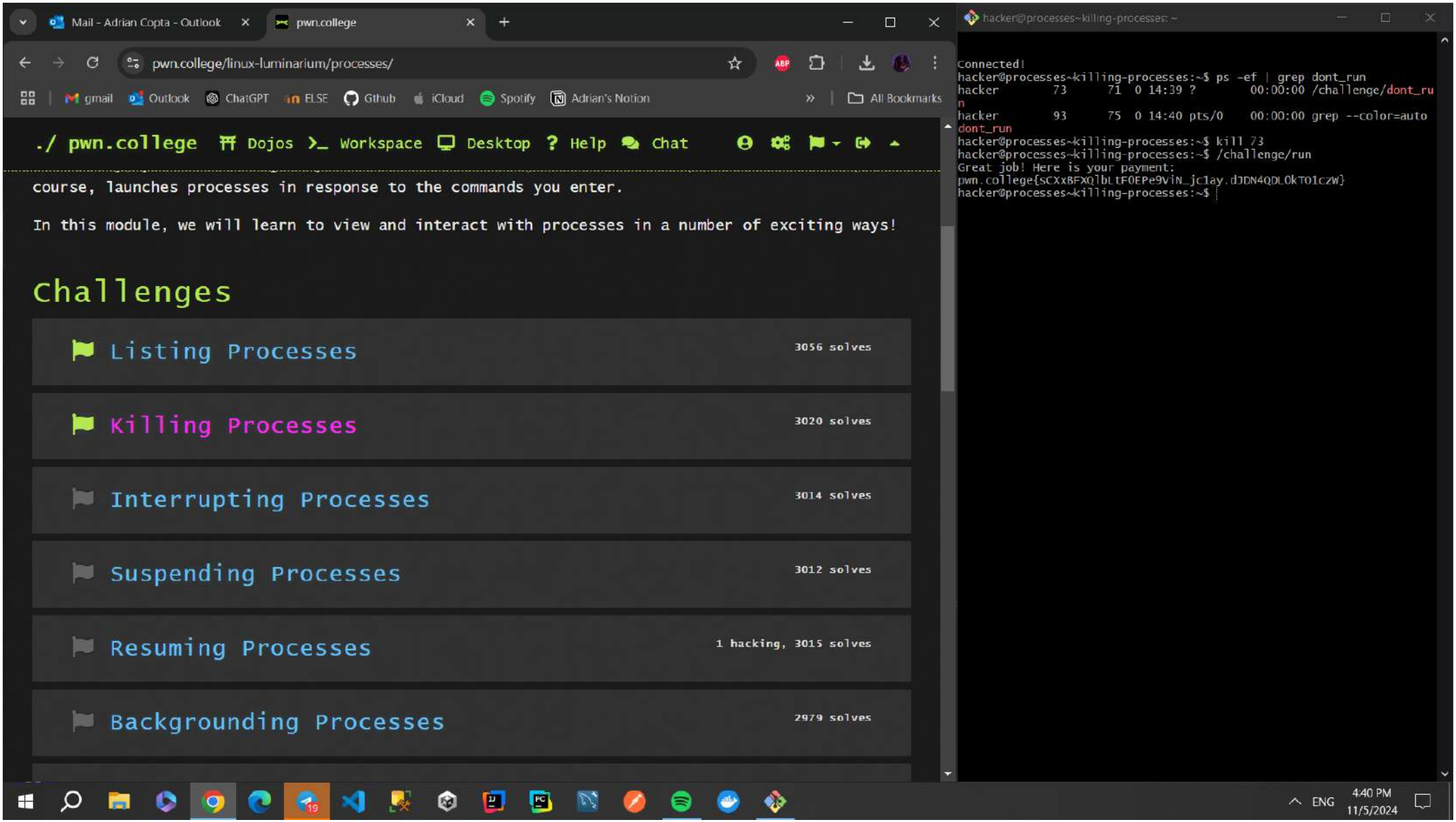

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Exporting Variables
3148 solves
Printing Exported Variables
1 hacking, 3160 solves
There are multiple ways to access variables in bash. echo was just one of them, and we'll now learn at least one more in this challenge.
Try the env command: it'll print out every exported variable set in your shell, and you can look through that output to find the FLAG variable!
Start
Flag
Submit
Correct
Storing Command Output
2566 solves
hacker@variables~printing-exported-variables: ~
Connected!
hacker@variables~printing-exported-variables:~$ env $FLAG
env: 'pwn.college{oxmNa9RB8vImcPPc7K89tNNbVQC.dhTN1QDL0kTO1czW}': No such file or directory
hacker@variables~printing-exported-variables:~$
ENG
6:59 PM
10/29/2024

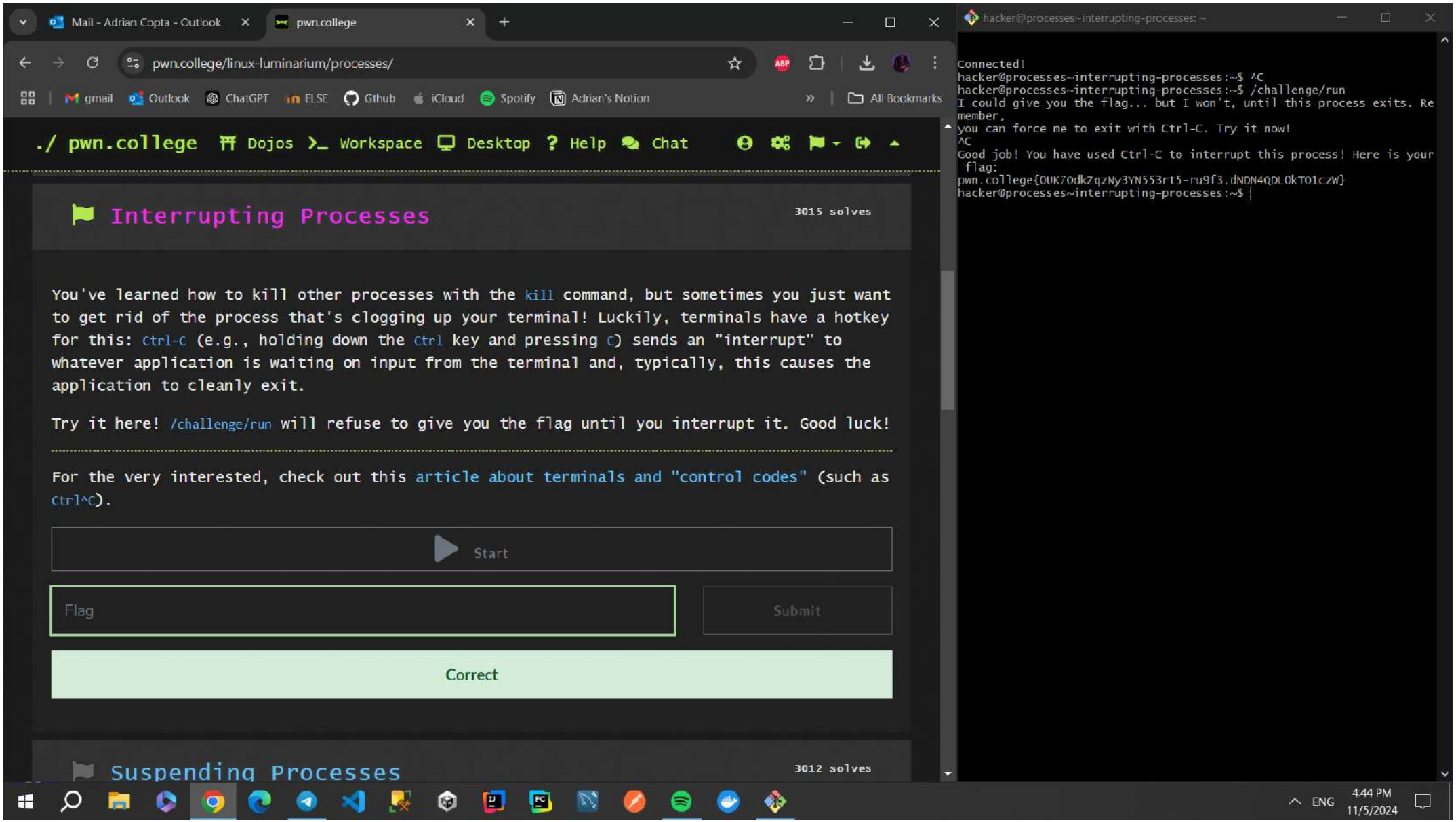

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
command into a variable. Luckily, the shell makes this quite easy using something called Command Substitution! Observe:
hacker@dojo:~$ FLAG=$(cat /flag)
hacker@dojo:~$ echo "$FLAG"
pwn.college{blahblahblah}
hacker@dojo:~$
Neat! Now, you practice. Read the output of the /challenge/run command directly into a variable called PWN, and it will contain the flag!
Trivia: You can also backticks instead of $(): FLAG=`cat /flag` instead of FLAG=$(cat /flag) in the example above. This is an older format, and has some disadvantages (for example, imagine if you wanted to nest command substitutions. How would you do $(cat $(find / -name flag)) with backticks? The official stance of pwn.college is that you should use $(blah) instead of `blah`.
Start
Flag
Submit
Correct
hacker@variables~storing-command-output: ~
Connected!
hacker@variables~storing-command-output:~$ PWN=&(/challenge/run)
[1] 86
You are not running me via Command Substitution! Please make sure to run me
with command substitution.
[1]+ Done PWN=
hacker@variables~storing-command-output:~$ echo $PWN
hacker@variables~storing-command-output:~$ PWN=$(/challenge/run)
Congratulations! You have read the flag into the PWN variable. Now print it out
and submit it!
hacker@variables~storing-command-output:~$ echo $PWN
pwn.college{wGgpSJTjOnEOOFvMAMGqOZqXsSA.dVzNOUDLOkTO1czW}
hacker@variables~storing-command-output:~$
ENG
7:06 PM
10/29/2024

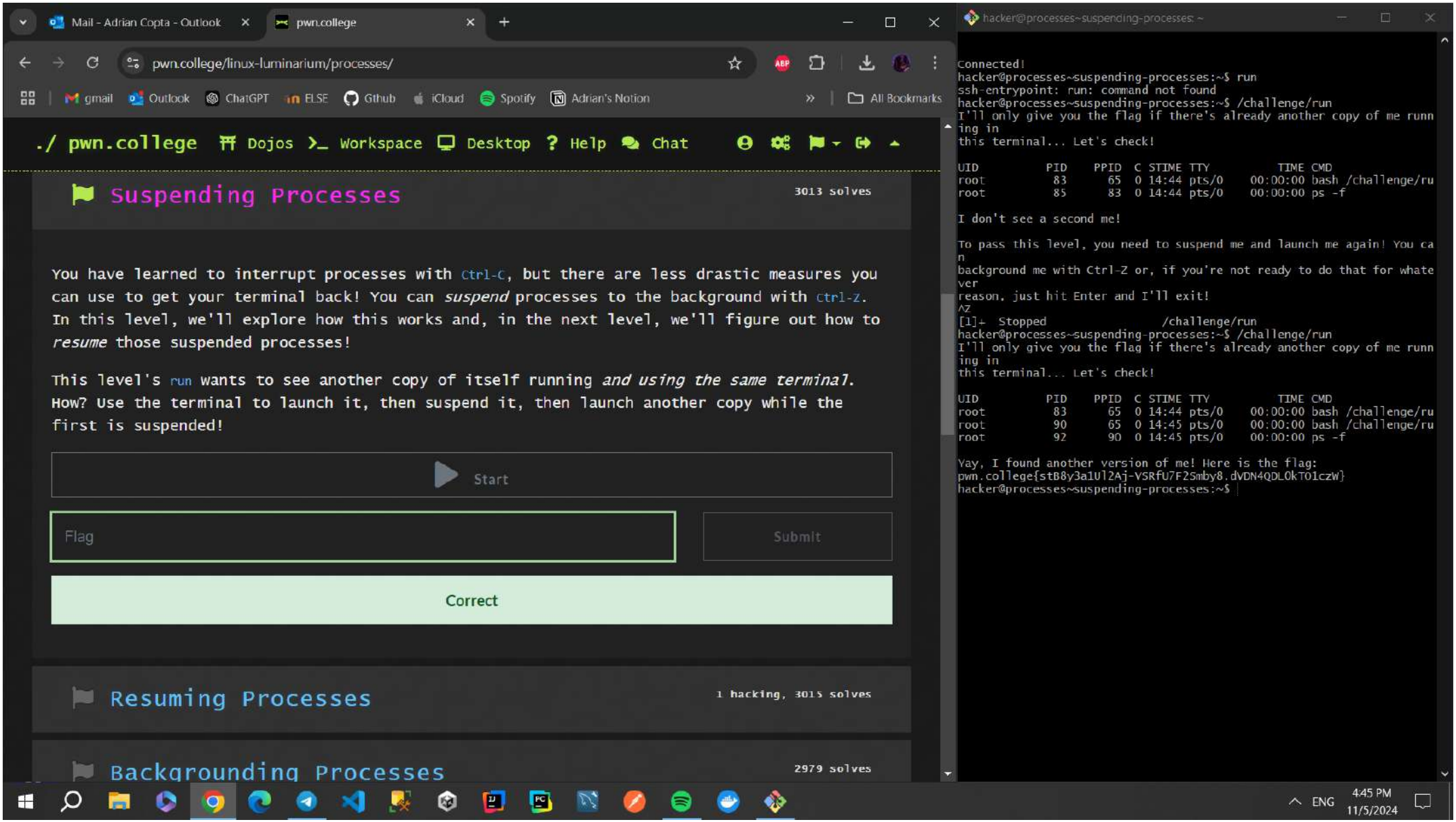

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Keep in mind, read reads data from your standard input! The first Hello!, above, was inputted rather than outputted. Let's try to be more explicit with that. Here, we annotated the beginning of each line with whether the line represents INPUT from the user or OUTPUT to the user:
INPUT: hacker@dojo:~$ echo $MY_VARIABLE
OUTPUT:
INPUT: hacker@dojo:~$ read MY_VARIABLE
INPUT: Hello!
INPUT: hacker@dojo:~$ echo "You entered: $MY_VARIABLE"
OUTPUT: You entered: Hello!
In this challenge, your job is to use read to set the PWN variable to the value COLLEGE. Good luck!
Start
Flag
Submit
Correct
Reading Files
2 hacking, 3065 solves
hacker@variables~reading-input: ~
Connected!
hacker@variables~reading-input:~$ read PWN
COLLEGE
You've set the PWN variable properly! As promised, here is the flag:
pwn.college{kM65rtrYvJt0nrzCl57keOGxoGe.dhzN1QDL0kTO1czW}
hacker@variables~reading-input:~$
ENG
7:08 PM
10/29/2024

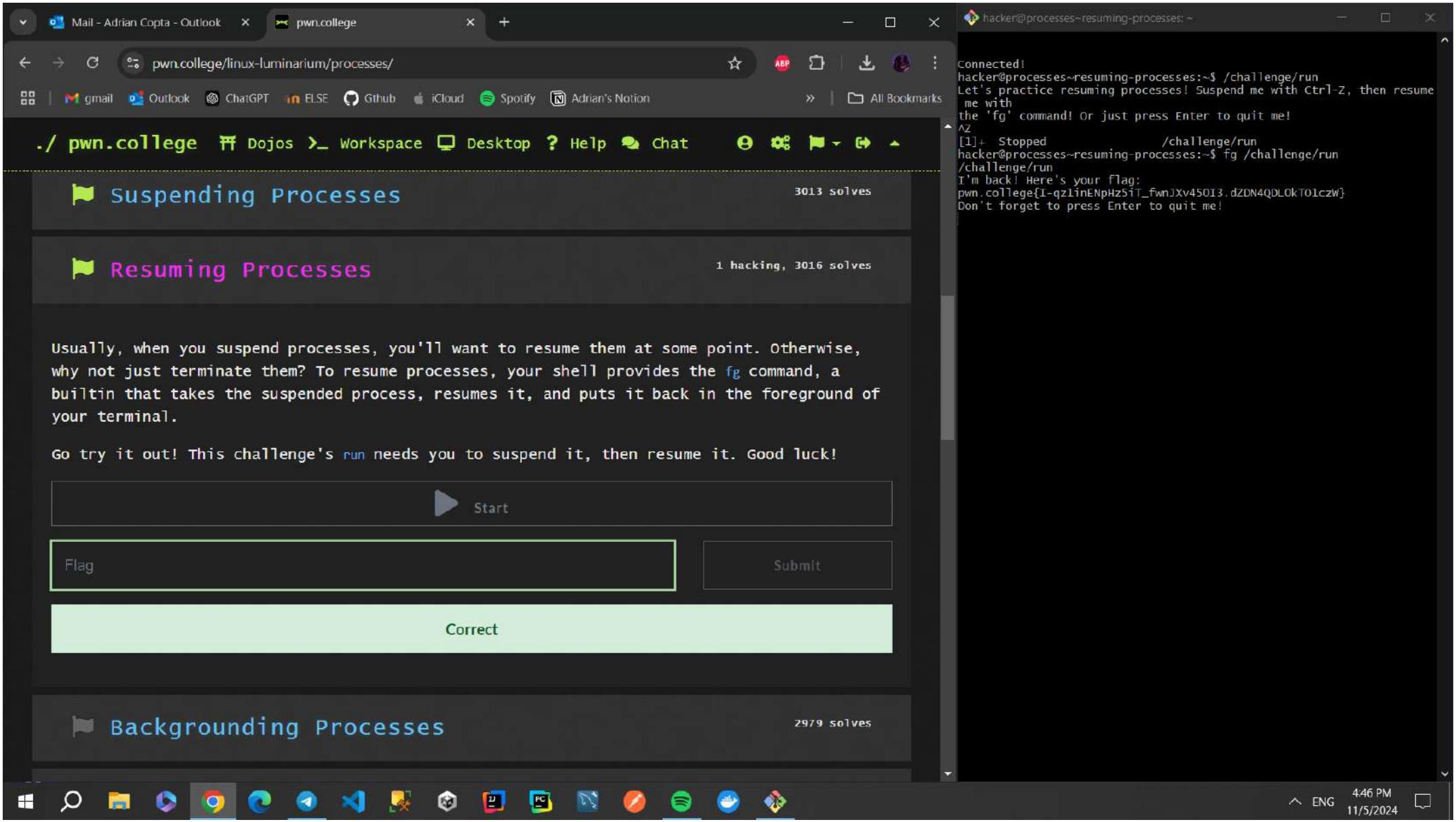

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/variables/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
This works, but it represents what grouchy hackers call a "Useless Use of Cat". That is, running a whole other program just to read the file is a waste. It turns out that you can just use the powers of the shell!
Previously, you read user input into a variable. You've also previously redirected files into command input! Put them together, and you can read files with the shell.
hacker@dojo:~$ echo "test" > some_file
hacker@dojo:~$ read VAR < some_file
hacker@dojo:~$ echo $VAR
test
What happened there? The example redirects some_file into the standard input of read, and so when read reads into VAR, it reads from the file! Now, use that to read /challenge/read_me into the PWN environment variable, and we'll give you the flag! The /challenge/read_me will keep changing, so you'll need to read it right into the PWN variable with one command!
Start
Flag
Submit
Correct
30-Day Scoreboard:
hacker@variables~reading-files: ~
Connected!
hacker@variables~reading-files:~$ read PWN < /challenge/read_me
You've set the PWN variable properly! As promised, here is the flag:
pwn.college{ktOaYCBZ6KMWcIWmHK2LUqLqFmm.dBjM4QDLOkTOlczW}
hacker@variables~reading-files:~$
ENG
7:11 PM
10/29/2024

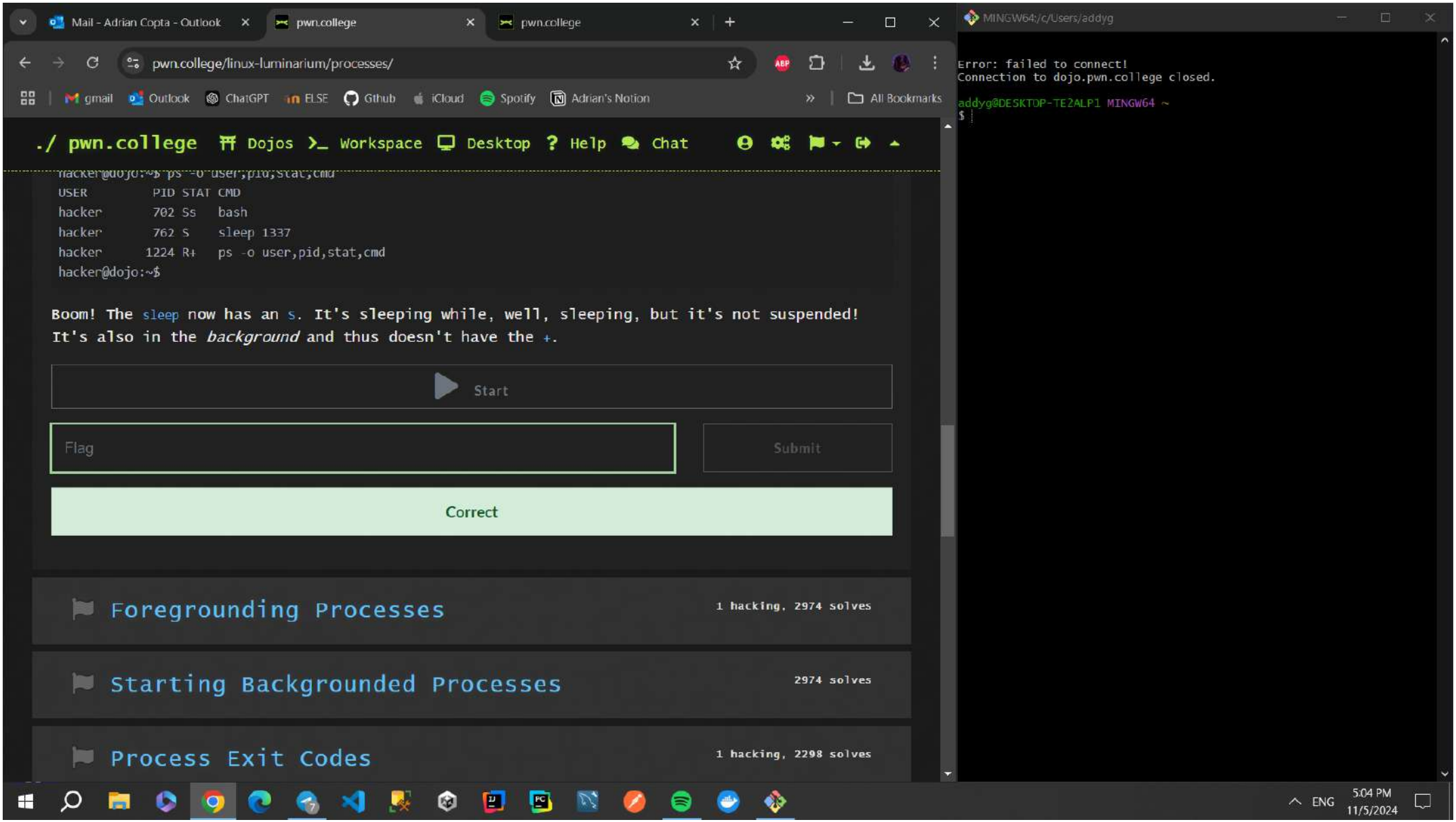

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
All Bookmarks
./ pwn.college
other stuff we won't get into right now.
Anyways! Let's practice. In this level, I have once again renamed /challenge/run to a random filename, and this time made it so that you cannot ls the /challenge directory! But I also launched it, so can find it in the running process list, figure out the filename, and relaunch it directly for the flag! Good luck!
NOTE: Both ps -ef and ps aux truncate the command listing to the width of your terminal (which is why the examples above line up so nicely on the right side of the screen. If you can't read the whole path to the process, you might need to enlarge your terminal (or redirect the output somewhere to avoid this truncating behavior)!
Start
Flag
Submit
Correct
Killing Processes
3019 solves
Interrupting Processes
3014 solves
hacker@processes~listing-processes: ~
hacker@processes~listing-processes:~$ ps aux | grep /challenge/
root 68 0.0 0.0 4132 2560 ? S 14:13 0:00 /challenge/9506-run-6266
hacker 103 0.0 0.0 4140 2560 pts/0 S+ 14:32 0:00 grep --color=auto /challenge/
hacker@processes~listing-processes:~$ /challenge/9506-run-6266
ssh-entrypoint: /challenge/9506-run-6266: No such file or directory
hacker@processes~listing-processes:~$ /challenge/9506-run-6266
Yahaha, you found me! Here is your flag:
pwn.college{8dit8uAJljCVNj61rqeX41RbTNn.dhzM4QDLOkTO1czW}
Now I will sleep for a while (so that you could find me with 'ps').
ENG
4:33 PM
11/5/2024

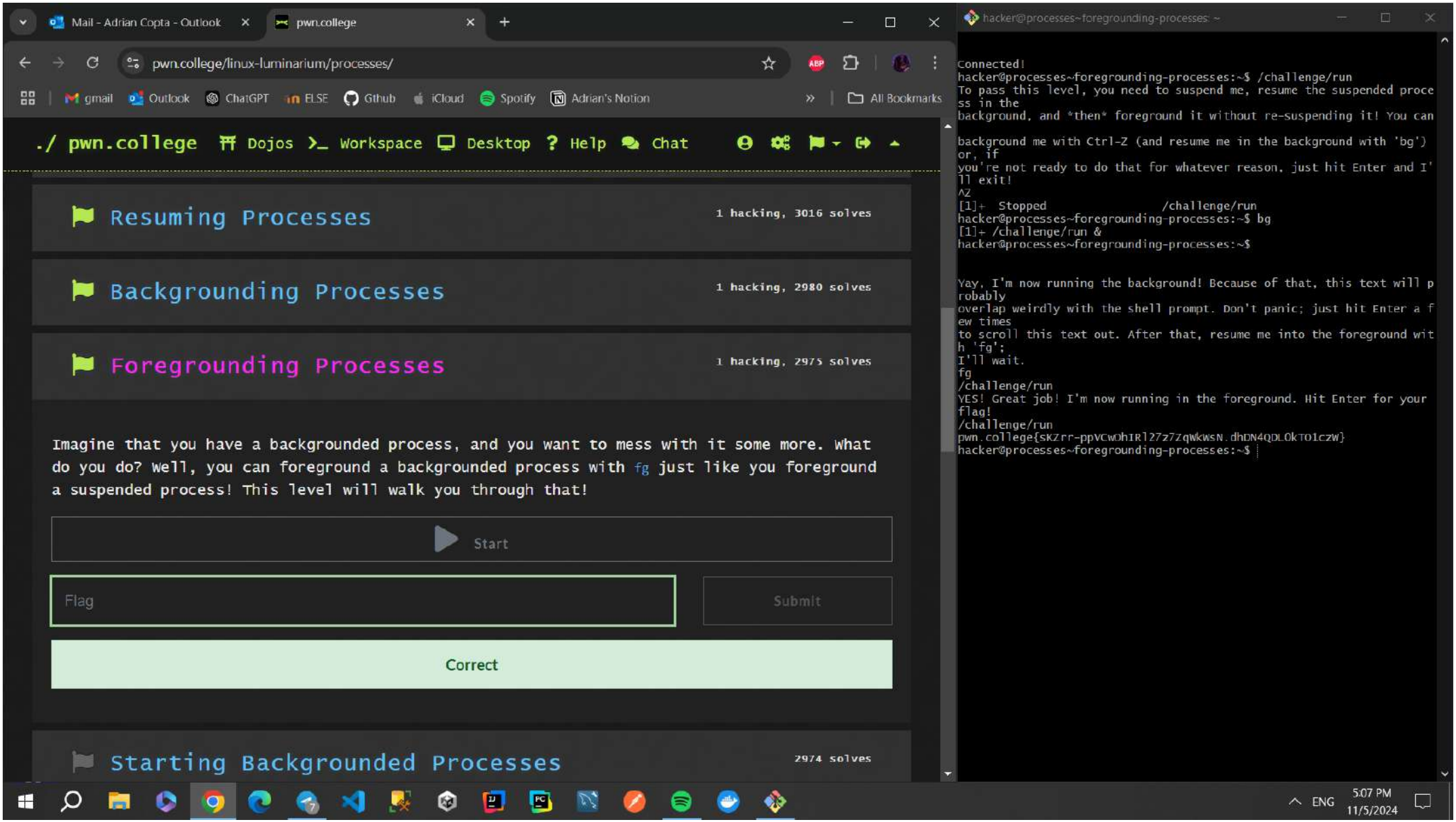

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
course, launches processes in response to the commands you enter.
In this module, we will learn to view and interact with processes in a number of exciting ways!
Challenges
Listing Processes
3056 solves
Killing Processes
3020 solves
Interrupting Processes
3014 solves
Suspending Processes
3012 solves
Resuming Processes
1 hacking, 3015 solves
Backgrounding Processes
2979 solves
hacker@processes~killing-processes: ~
Connected!
hacker@processes~killing-processes:~$ ps -ef | grep dont_run
hacker 73 71 0 14:39 ? 00:00:00 /challenge/dont_run
hacker 93 75 0 14:40 pts/0 00:00:00 grep --color=auto dont_run
hacker@processes~killing-processes:~$ kill 73
hacker@processes~killing-processes:~$ /challenge/run
Great job! Here is your payment:
pwn.college{sCXxBFxQlbLtFOEPe9ViN_jc1ay.dJDN4QDLOkTO1cZW}
hacker@processes~killing-processes:~$
ENG
4:40 PM
11/5/2024

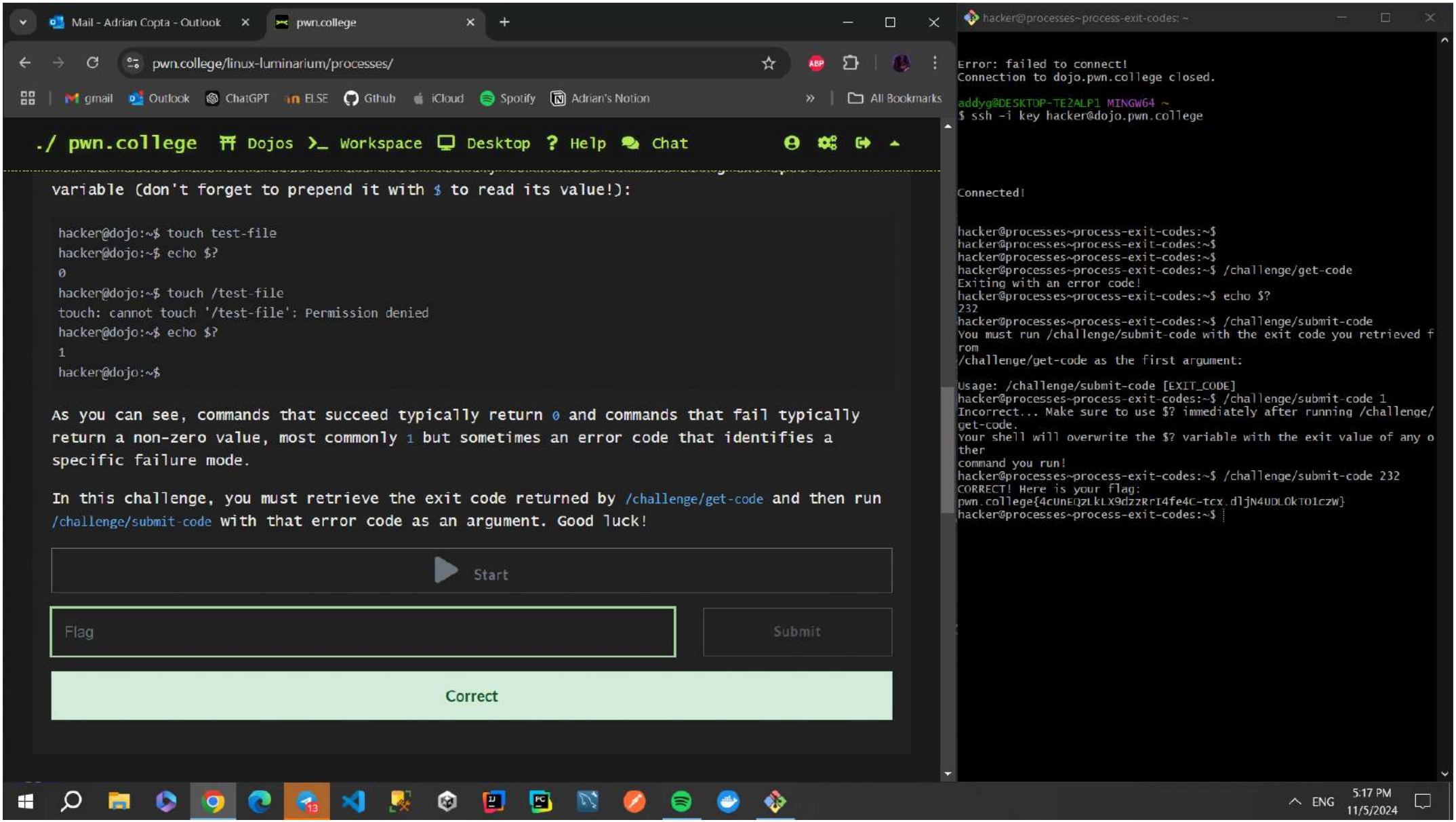

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Interrupting Processes
3015 solves
You've learned how to kill other processes with the kill command, but sometimes you just want to get rid of the process that's clogging up your terminal! Luckily, terminals have a hotkey for this: Ctrl-C (e.g., holding down the Ctrl key and pressing C) sends an "interrupt" to whatever application is waiting on input from the terminal and, typically, this causes the application to cleanly exit.
Try it here! /challenge/run will refuse to give you the flag until you interrupt it. Good luck!
For the very interested, check out this article about terminals and "control codes" (such as Ctrl^C).
Start
Flag
Submit
Correct
Suspending Processes
3012 solves
hacker@processes~interrupting-processes: ~
Connected!
hacker@processes~interrupting-processes:~$ ^C
hacker@processes~interrupting-processes:~$ /challenge/run
I could give you the flag... but I won't, until this process exits. Remember,
you can force me to exit with Ctrl-C. Try it now!
^C
Good job! You have used Ctrl-C to interrupt this process! Here is your flag:
pwn.college{OUK7OdkZqzNy3YN553rt5-ru9f3.dNDN4QDLOkTO1czW}
hacker@processes~interrupting-processes:~$
ENG
4:44 PM
11/5/2024

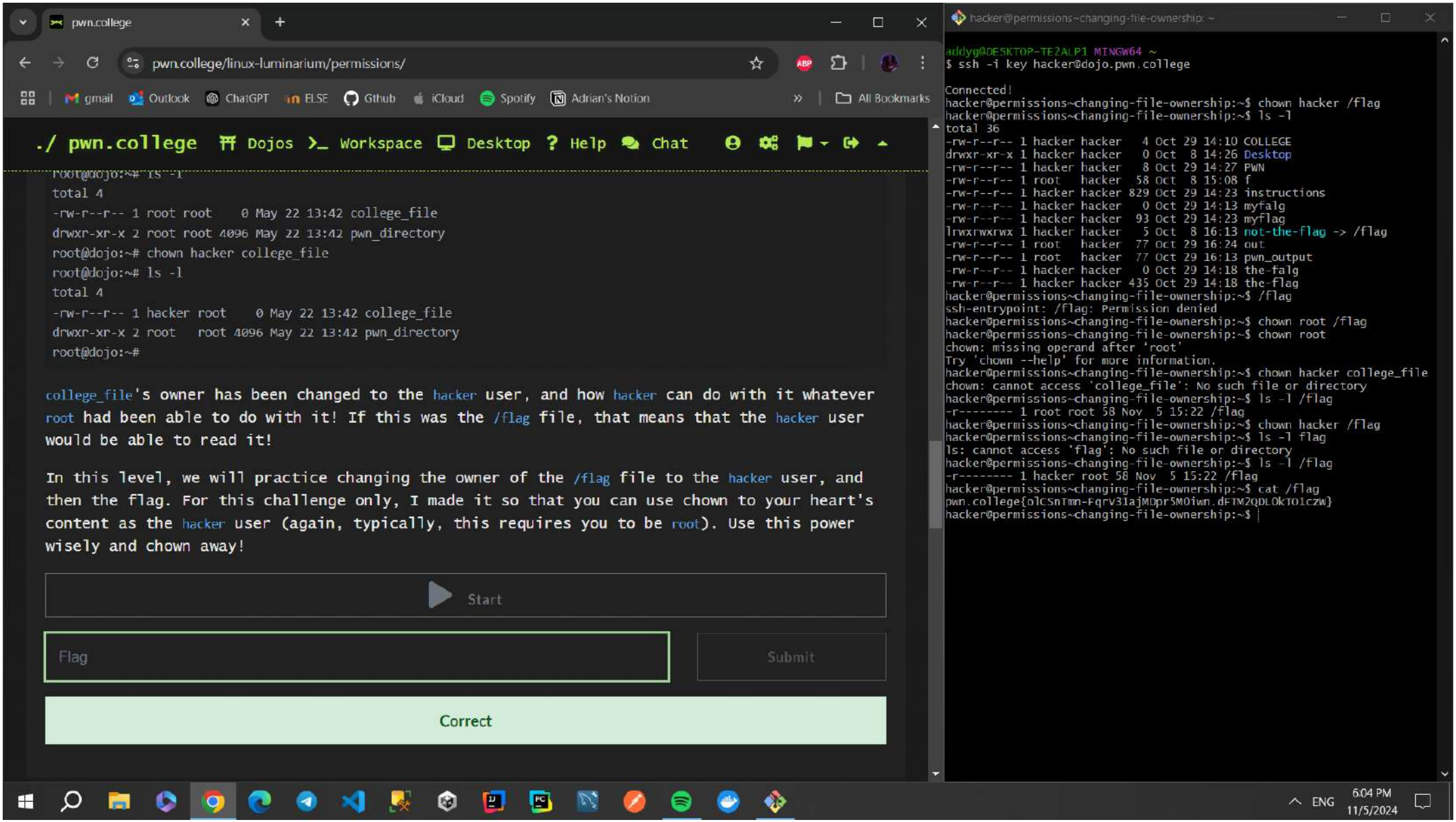

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Suspending Processes
3013 solves
You have learned to interrupt processes with Ctrl-C, but there are less drastic measures you can use to get your terminal back! You can suspend processes to the background with Ctrl-Z. In this level, we'll explore how this works and, in the next level, we'll figure out how to resume those suspended processes!
This level's run wants to see another copy of itself running and using the same terminal. How? Use the terminal to launch it, then suspend it, then launch another copy while the first is suspended!
Start
Flag
Submit
Correct
Resuming Processes
1 hacking, 3015 solves
Backgrounding Processes
2979 solves
hacker@processes~suspending-processes: ~
Connected!
hacker@processes~suspending-processes:~$ run
ssh-entrypoint: run: command not found
hacker@processes~suspending-processes:~$ /challenge/run
I'll only give you the flag if there's already another copy of me running in
this terminal... Let's check!
UID PID PPID C STIME TTY TIME CMD
root 83 65 0 14:44 pts/0 00:00:00 bash /challenge/ru
root 85 83 0 14:44 pts/0 00:00:00 ps -f
I don't see a second me!
To pass this level, you need to suspend me and launch me again! You can
background me with Ctrl-Z or, if you're not ready to do that for whatever
reason, just hit Enter and I'll exit!
^Z
[1]+ Stopped /challenge/run
hacker@processes~suspending-processes:~$ /challenge/run
I'll only give you the flag if there's already another copy of me running in
this terminal... Let's check!
UID PID PPID C STIME TTY TIME CMD
root 83 65 0 14:44 pts/0 00:00:00 bash /challenge/ru
root 90 65 0 14:45 pts/0 00:00:00 bash /challenge/ru
root 92 90 0 14:45 pts/0 00:00:00 ps -f
Yay, I found another version of me! Here is the flag:
pwn.college{stB8y3a1Ul2Aj-VSRfU7F25mby8.dVDN4QDLOkTO1czW}
hacker@processes~suspending-processes:~$
ENG
4:45 PM
11/5/2024

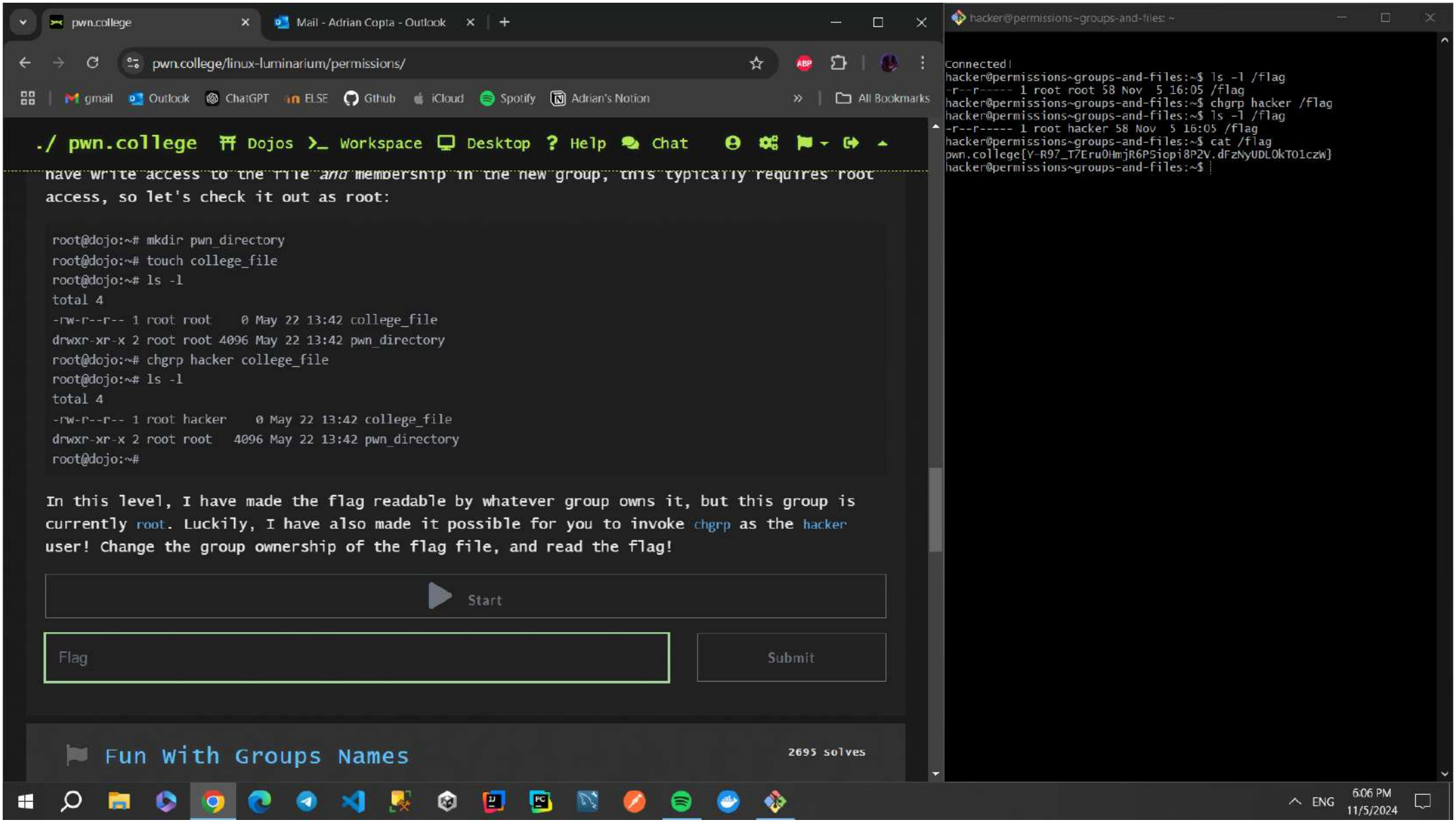

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Suspending Processes
3013 solves
Resuming Processes
1 hacking, 3016 solves
Usually, when you suspend processes, you'll want to resume them at some point. Otherwise, why not just terminate them? To resume processes, your shell provides the fg command, a builtin that takes the suspended process, resumes it, and puts it back in the foreground of your terminal.
Go try it out! This challenge's run needs you to suspend it, then resume it. Good luck!
Start
Flag
Submit
Correct
Backgrounding Processes
2979 solves
hacker@processes~resuming-processes: ~
Connected!
hacker@processes~resuming-processes:~$ /challenge/run
Let's practice resuming processes! Suspend me with Ctrl-Z, then resume me with
the 'fg' command! Or just press Enter to quit me!
^Z
[1]+ Stopped /challenge/run
hacker@processes~resuming-processes:~$ fg /challenge/run
/challenge/run
I'm back! Here's your flag:
pwn.college{I-qzlinENpHz5iT_fwnJXv45OI3.dZDN4QDLOkTO1czW}
Don't forget to press Enter to quit me!
ENG
4:46 PM
11/5/2024

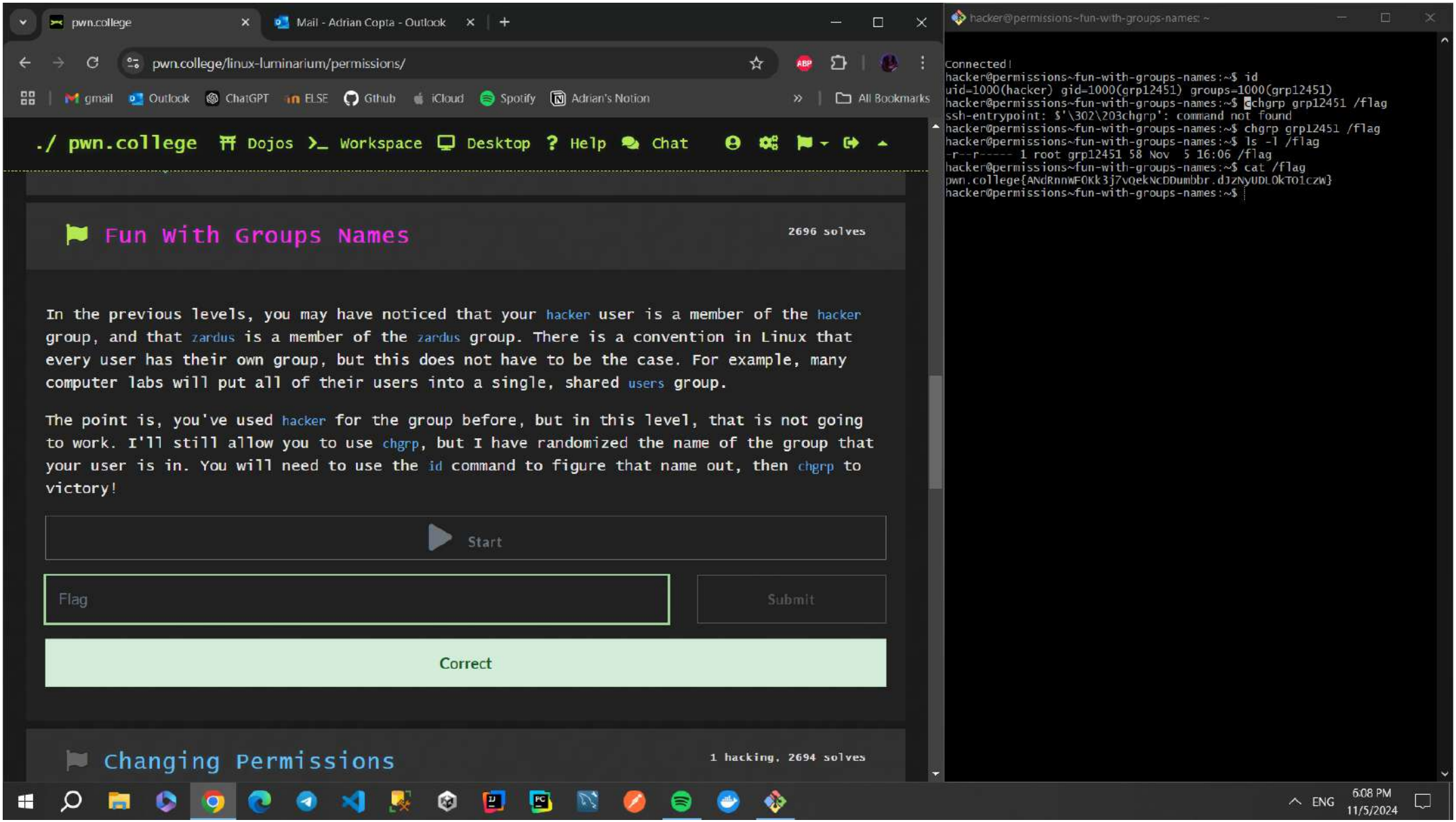

Mail - Adrian Copta - Outlook
pwn.college
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ ps -o user,pid,stat,cmd
USER PID STAT CMD
hacker 702 Ss bash
hacker 762 S sleep 1337
hacker 1224 R+ ps -o user,pid,stat,cmd
hacker@dojo:~$
Boom! The sleep now has an s. It's sleeping while, well, sleeping, but it's not suspended!
It's also in the background and thus doesn't have the +.
Start
Flag
Submit
Correct
Foregrounding Processes
1 hacking, 2974 solves
Starting Backgrounded Processes
2974 solves
Process Exit Codes
1 hacking, 2298 solves
MINGW64:/c/Users/addyg
Error: failed to connect!
Connection to dojo.pwn.college closed.
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$
ENG
5:04 PM
11/5/2024

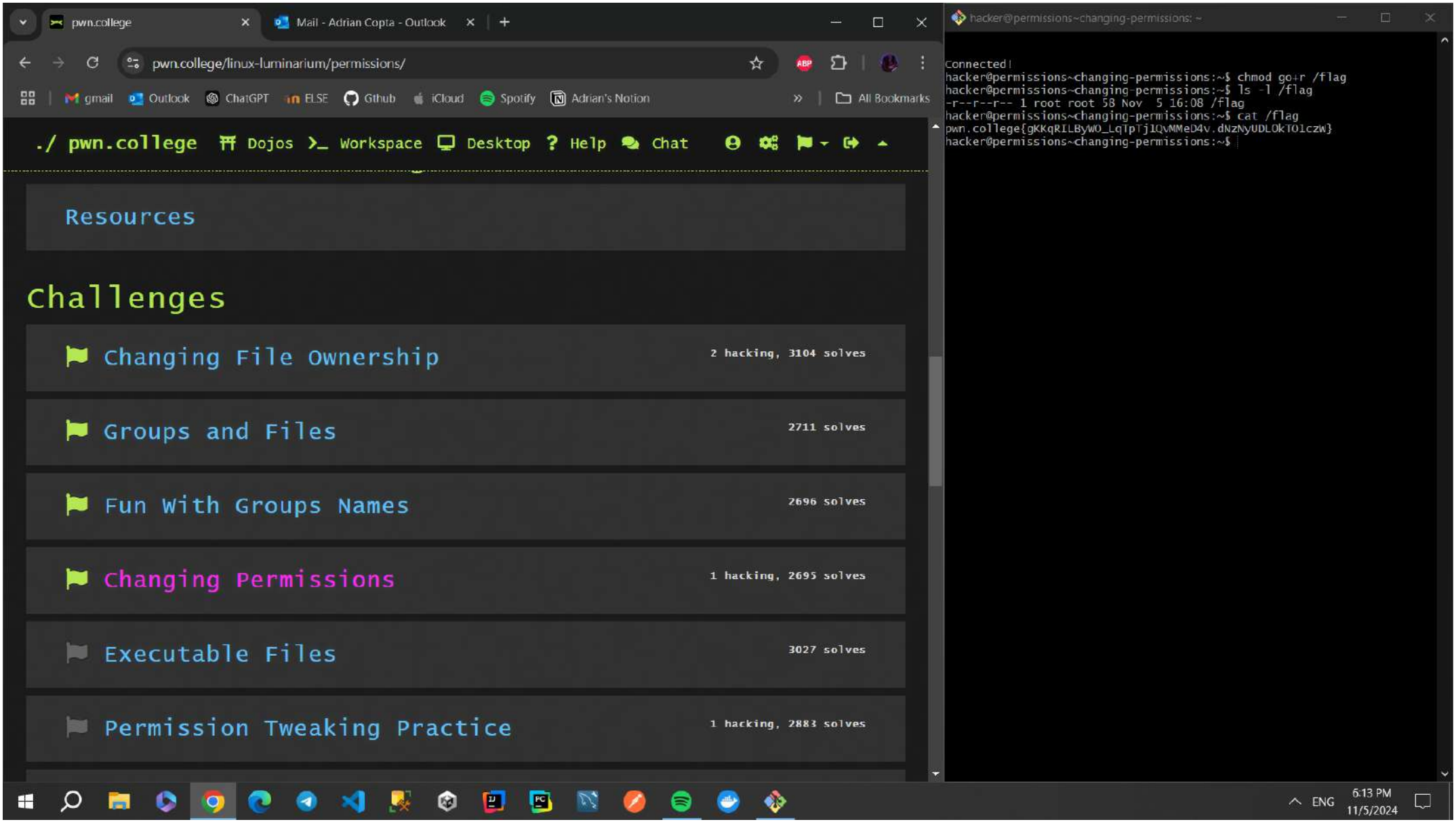

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Resuming Processes
1 hacking, 3016 solves
Backgrounding Processes
1 hacking, 2980 solves
Foregrounding Processes
1 hacking, 2975 solves
Imagine that you have a backgrounded process, and you want to mess with it some more. What do you do? Well, you can foreground a backgrounded process with fg just like you foreground a suspended process! This level will walk you through that!
Start
Flag
Submit
Correct
Starting Backgrounded Processes
2974 solves
hacker@processes~foregrounding-processes: ~
Connected!
hacker@processes~foregrounding-processes:~$ /challenge/run
To pass this level, you need to suspend me, resume the suspended process in the
background, and *then* foreground it without re-suspending it! You can
background me with Ctrl-Z (and resume me in the background with 'bg') or, if
you're not ready to do that for whatever reason, just hit Enter and I'll exit!
^Z
[1]+ Stopped /challenge/run
hacker@processes~foregrounding-processes:~$ bg
[1]+ /challenge/run &
hacker@processes~foregrounding-processes:~$
Yay, I'm now running the background! Because of that, this text will probably
overlap weirdly with the shell prompt. Don't panic; just hit Enter a few times
to scroll this text out. After that, resume me into the foreground with 'fg';
I'll wait.
fg
/challenge/run
YES! Great job! I'm now running in the foreground. Hit Enter for your flag!
/challenge/run
pwn.college{sKZrr-ppVCwOhIRl27z7ZqWkWsN.dhDN4QDLOkTO1czW}
hacker@processes~foregrounding-processes:~$
ENG
5:07 PM
11/5/2024

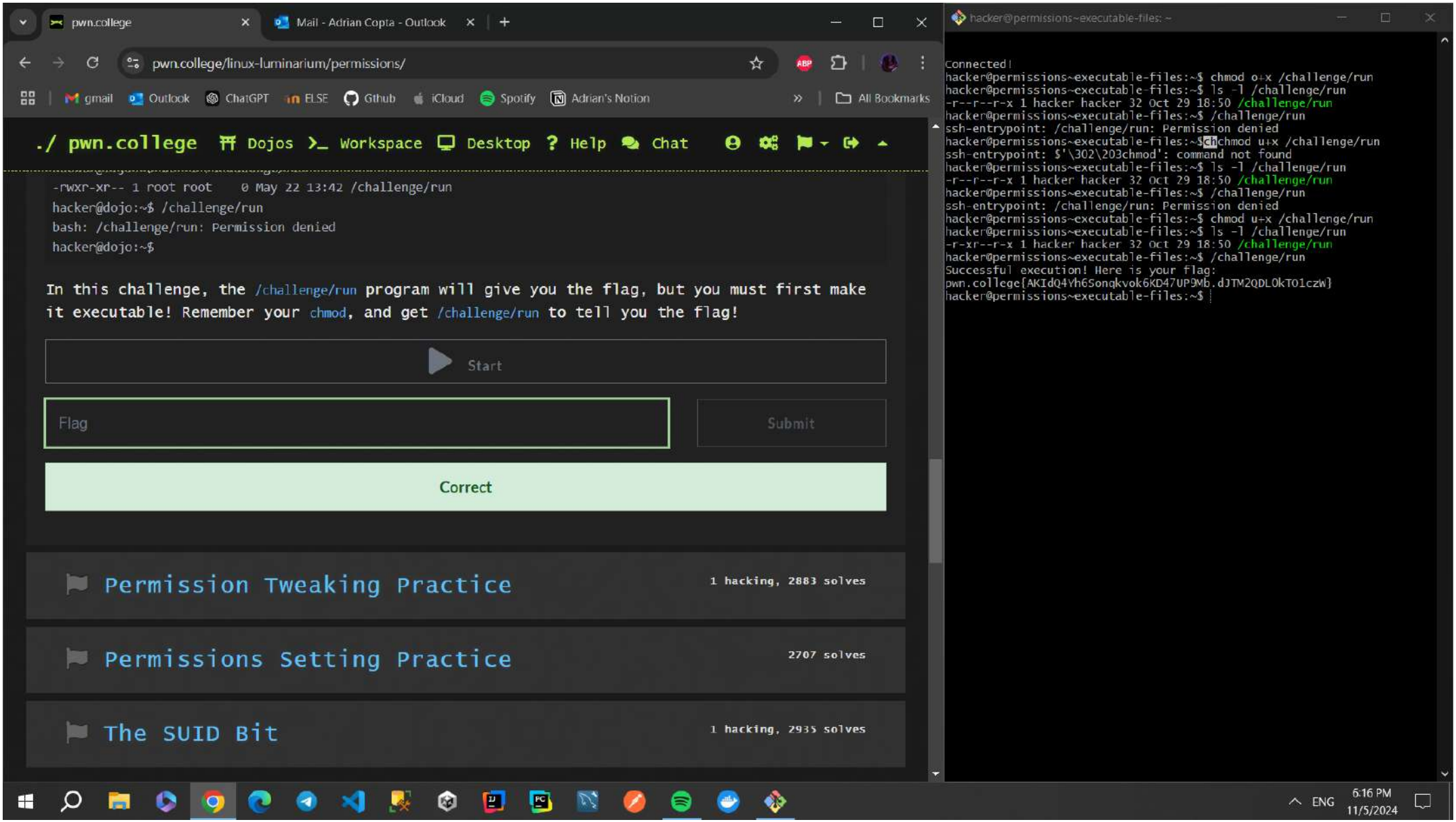

Mail - Adrian Copta - Outlook
pwn.college
pwn.college/linux-luminarium/processes/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
variable (don't forget to prepend it with $ to read its value!):
hacker@dojo:~$ touch test-file
hacker@dojo:~$ echo $?
0
hacker@dojo:~$ touch /test-file
touch: cannot touch '/test-file': Permission denied
hacker@dojo:~$ echo $?
1
hacker@dojo:~$
As you can see, commands that succeed typically return 0 and commands that fail typically return a non-zero value, most commonly 1 but sometimes an error code that identifies a specific failure mode.
In this challenge, you must retrieve the exit code returned by /challenge/get-code and then run /challenge/submit-code with that error code as an argument. Good luck!
Start
Flag
Submit
Correct
hacker@processes~process-exit-codes: ~
Error: failed to connect!
Connection to dojo.pwn.college closed.
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
Connected!
hacker@processes~process-exit-codes:~$
hacker@processes~process-exit-codes:~$
hacker@processes~process-exit-codes:~$
hacker@processes~process-exit-codes:~$ /challenge/get-code
Exiting with an error code!
hacker@processes~process-exit-codes:~$ echo $?
232
hacker@processes~process-exit-codes:~$ /challenge/submit-code
You must run /challenge/submit-code with the exit code you retrieved from
/challenge/get-code as the first argument:
Usage: /challenge/submit-code [EXIT_CODE]
hacker@processes~process-exit-codes:~$ /challenge/submit-code 1
Incorrect... Make sure to use $? immediately after running /challenge/get-code.
Your shell will overwrite the $? variable with the exit value of any other
command you run!
hacker@processes~process-exit-codes:~$ /challenge/submit-code 232
CORRECT! Here is your flag:
pwn.college{4cUnEQzLkLX9dzzRrI4fe4C-tcx.dljN4UDLOkTO1czW}
hacker@processes~process-exit-codes:~$
ENG
5:17 PM
11/5/2024

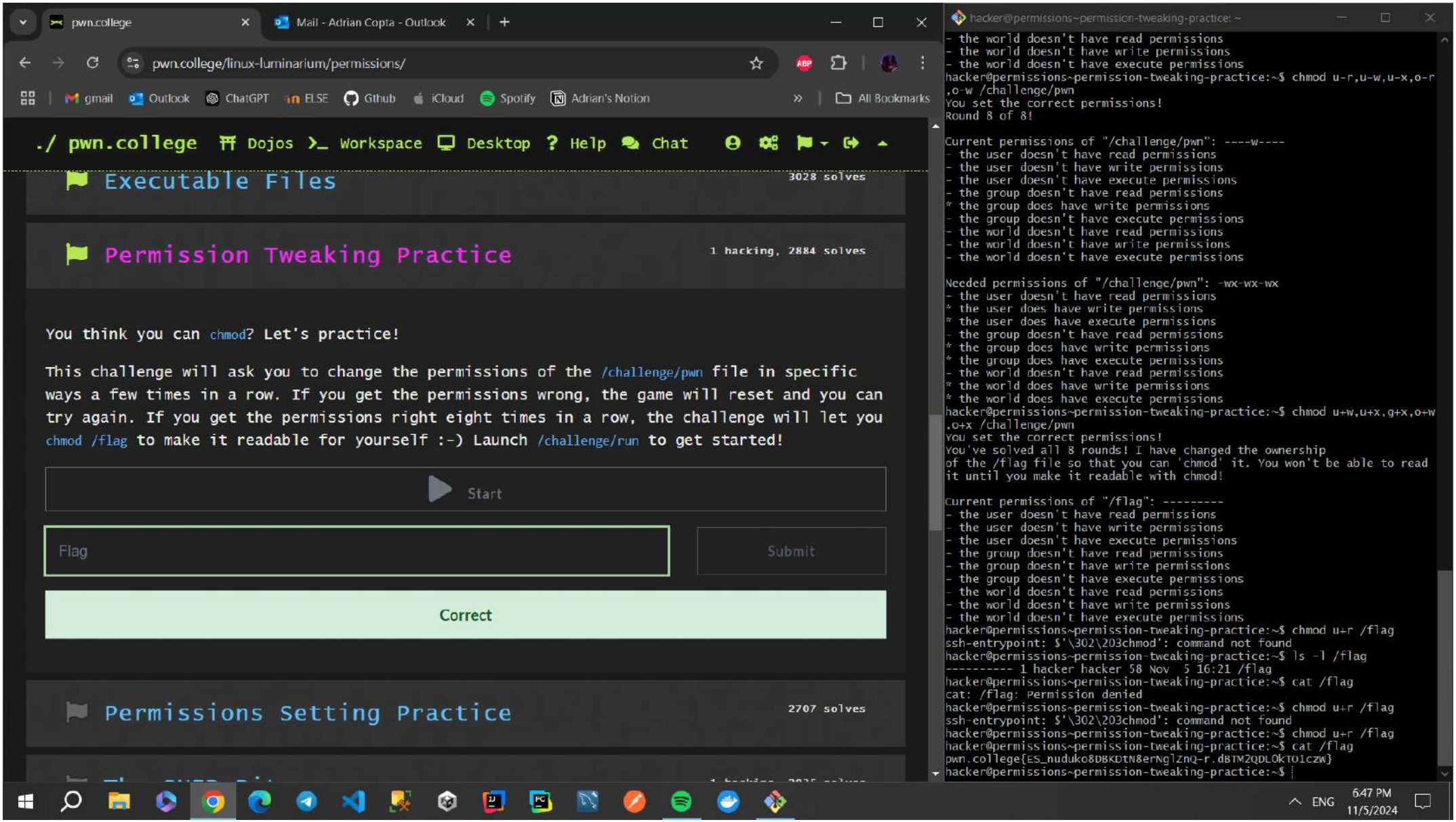

pwn.college
pwn.college/linux-luminarium/permissions/
gmail Outlook ChatGPT ELSE Github iCloud Spotify Adrian's Notion All Bookmarks
./ pwn.college Dojos Workspace Desktop Help Chat
root@dojo:~# ls -l
total 4
-rw-r--r-- 1 root root 0 May 22 13:42 college_file
drwxr-xr-x 2 root root 4096 May 22 13:42 pwn_directory
root@dojo:~# chown hacker college_file
root@dojo:~# ls -l
total 4
-rw-r--r-- 1 hacker root 0 May 22 13:42 college_file
drwxr-xr-x 2 root root 4096 May 22 13:42 pwn_directory
root@dojo:~#
college_file's owner has been changed to the hacker user, and how hacker can do with it whatever root had been able to do with it! If this was the /flag file, that means that the hacker user would be able to read it!
In this level, we will practice changing the owner of the /flag file to the hacker user, and then the flag. For this challenge only, I made it so that you can use chown to your heart's content as the hacker user (again, typically, this requires you to be root). Use this power wisely and chown away!
Start
Flag
Submit
Correct
hacker@permissions~changing-file-ownership: ~
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
Connected!
hacker@permissions~changing-file-ownership:~$ chown hacker /flag
hacker@permissions~changing-file-ownership:~$ ls -l
total 36
-rw-r--r-- 1 hacker hacker 4 Oct 29 14:10 COLLEGE
drwxr-xr-x 1 hacker hacker 0 Oct 8 14:26 Desktop
-rw-r--r-- 1 hacker hacker 8 Oct 29 14:27 PWN
-rw-r--r-- 1 root hacker 58 Oct 8 15:08 f
-rw-r--r-- 1 hacker hacker 829 Oct 29 14:23 instructions
-rw-r--r-- 1 hacker hacker 0 Oct 29 14:13 myfalg
-rw-r--r-- 1 hacker hacker 93 Oct 29 14:23 myflag
lrwxrwxrwx 1 hacker hacker 5 Oct 8 16:13 not-the-flag -> /flag
-rw-r--r-- 1 root hacker 77 Oct 29 16:24 out
-rw-r--r-- 1 root hacker 77 Oct 29 16:13 pwn_output
-rw-r--r-- 1 hacker hacker 0 Oct 29 14:18 the-falg
-rw-r--r-- 1 hacker hacker 435 Oct 29 14:18 the-flag
hacker@permissions~changing-file-ownership:~$ /flag
ssh-entrypoint: /flag: Permission denied
hacker@permissions~changing-file-ownership:~$ chown root /flag
hacker@permissions~changing-file-ownership:~$ chown root
chown: missing operand after 'root'
Try 'chown --help' for more information.
hacker@permissions~changing-file-ownership:~$ chown hacker college_file
chown: cannot access 'college_file': No such file or directory
hacker@permissions~changing-file-ownership:~$ ls -l /flag
-r-------- 1 root root 58 Nov 5 15:22 /flag
hacker@permissions~changing-file-ownership:~$ chown hacker /flag
hacker@permissions~changing-file-ownership:~$ ls -l flag
ls: cannot access 'flag': No such file or directory
hacker@permissions~changing-file-ownership:~$ ls -l /flag
-r-------- 1 hacker root 58 Nov 5 15:22 /flag
hacker@permissions~changing-file-ownership:~$ cat /flag
pwn.college{olCSnTmn-FqrV31ajMDpr5MOiwn.dFTM2QDLOkTO1czW}
hacker@permissions~changing-file-ownership:~$
ENG 6:04 PM 11/5/2024

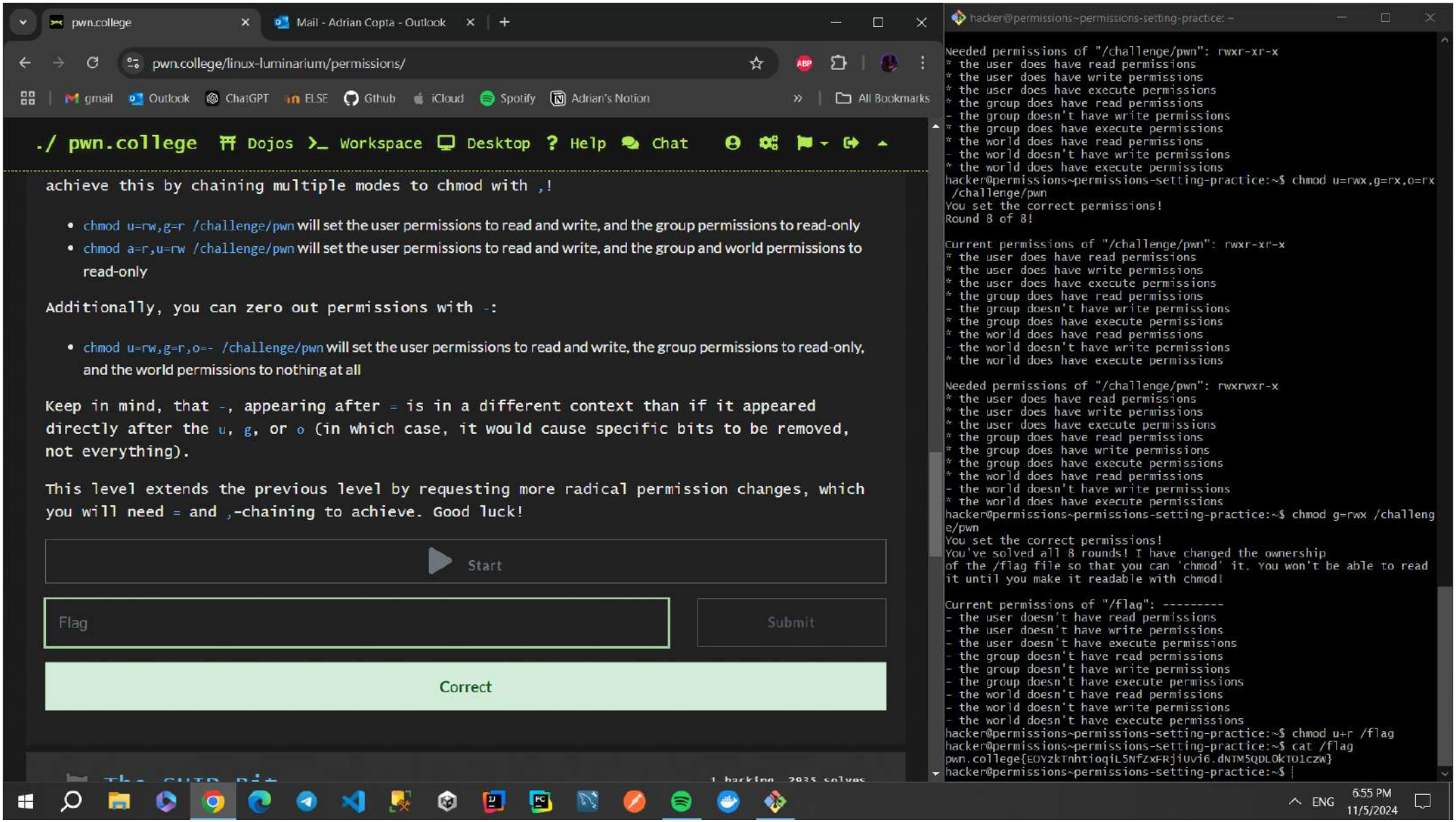

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
have write access to the file and membership in the new group, this typically requires root access, so let's check it out as root:
root@dojo:~# mkdir pwn_directory
root@dojo:~# touch college_file
root@dojo:~# ls -l
total 4
-rw-r--r-- 1 root root 0 May 22 13:42 college_file
drwxr-xr-x 2 root root 4096 May 22 13:42 pwn_directory
root@dojo:~# chgrp hacker college_file
root@dojo:~# ls -l
total 4
-rw-r--r-- 1 root hacker 0 May 22 13:42 college_file
drwxr-xr-x 2 root root 4096 May 22 13:42 pwn_directory
root@dojo:~#
In this level, I have made the flag readable by whatever group owns it, but this group is currently root. Luckily, I have also made it possible for you to invoke chgrp as the hacker user! Change the group ownership of the flag file, and read the flag!
Start
Flag
Submit
Fun With Groups Names
2695 solves
hacker@permissions~groups-and-files: ~
Connected!
hacker@permissions~groups-and-files:~$ ls -l /flag
-r--r----- 1 root root 58 Nov 5 16:05 /flag
hacker@permissions~groups-and-files:~$ chgrp hacker /flag
hacker@permissions~groups-and-files:~$ ls -l /flag
-r--r----- 1 root hacker 58 Nov 5 16:05 /flag
hacker@permissions~groups-and-files:~$ cat /flag
pwn.college{Y-R97_T7EruOHmjR6PSiopi8P2V.dFzNyUDLOkTO1czW}
hacker@permissions~groups-and-files:~$
ENG
6:06 PM
11/5/2024

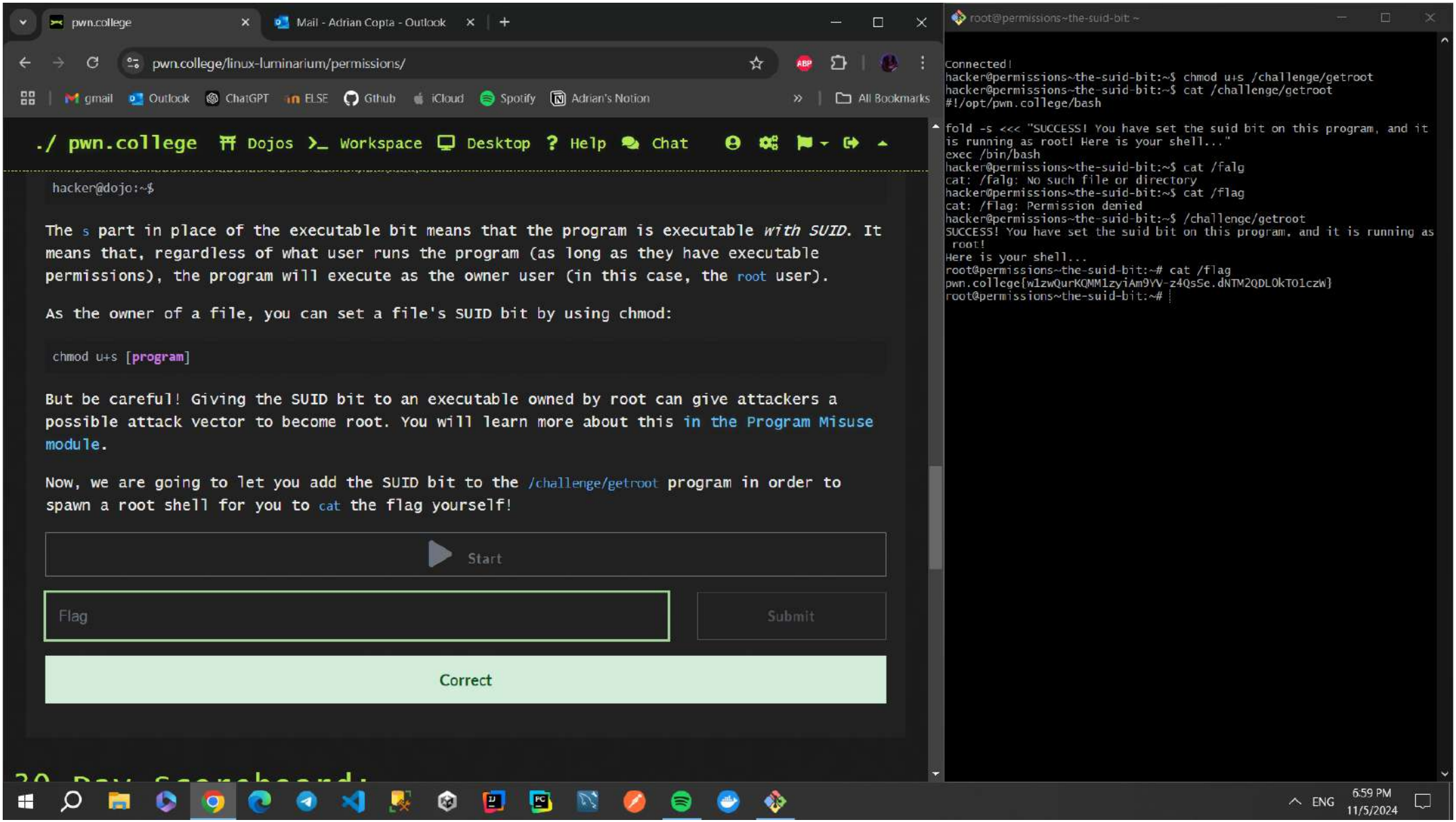

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Fun With Groups Names
2696 solves
In the previous levels, you may have noticed that your hacker user is a member of the hacker group, and that zardus is a member of the zardus group. There is a convention in Linux that every user has their own group, but this does not have to be the case. For example, many computer labs will put all of their users into a single, shared users group.
The point is, you've used hacker for the group before, but in this level, that is not going to work. I'll still allow you to use chgrp, but I have randomized the name of the group that your user is in. You will need to use the id command to figure that name out, then chgrp to victory!
Start
Flag
Submit
Correct
Changing Permissions
1 hacking, 2694 solves
hacker@permissions~fun-with-groups-names: ~
Connected!
hacker@permissions~fun-with-groups-names:~$ id
uid=1000(hacker) gid=1000(grp12451) groups=1000(grp12451)
hacker@permissions~fun-with-groups-names:~$ chgrp grp12451 /flag
ssh-entrypoint: $'\302\203chgrp': command not found
hacker@permissions~fun-with-groups-names:~$ chgrp grp12451 /flag
hacker@permissions~fun-with-groups-names:~$ ls -l /flag
-r--r----- 1 root grp12451 58 Nov 5 16:06 /flag
hacker@permissions~fun-with-groups-names:~$ cat /flag
pwn.college{ANdRnnWFOKk3j7vQekNcDDumbbr.dJzNyUDLOkTO1czW}
hacker@permissions~fun-with-groups-names:~$
ENG
6:08 PM
11/5/2024

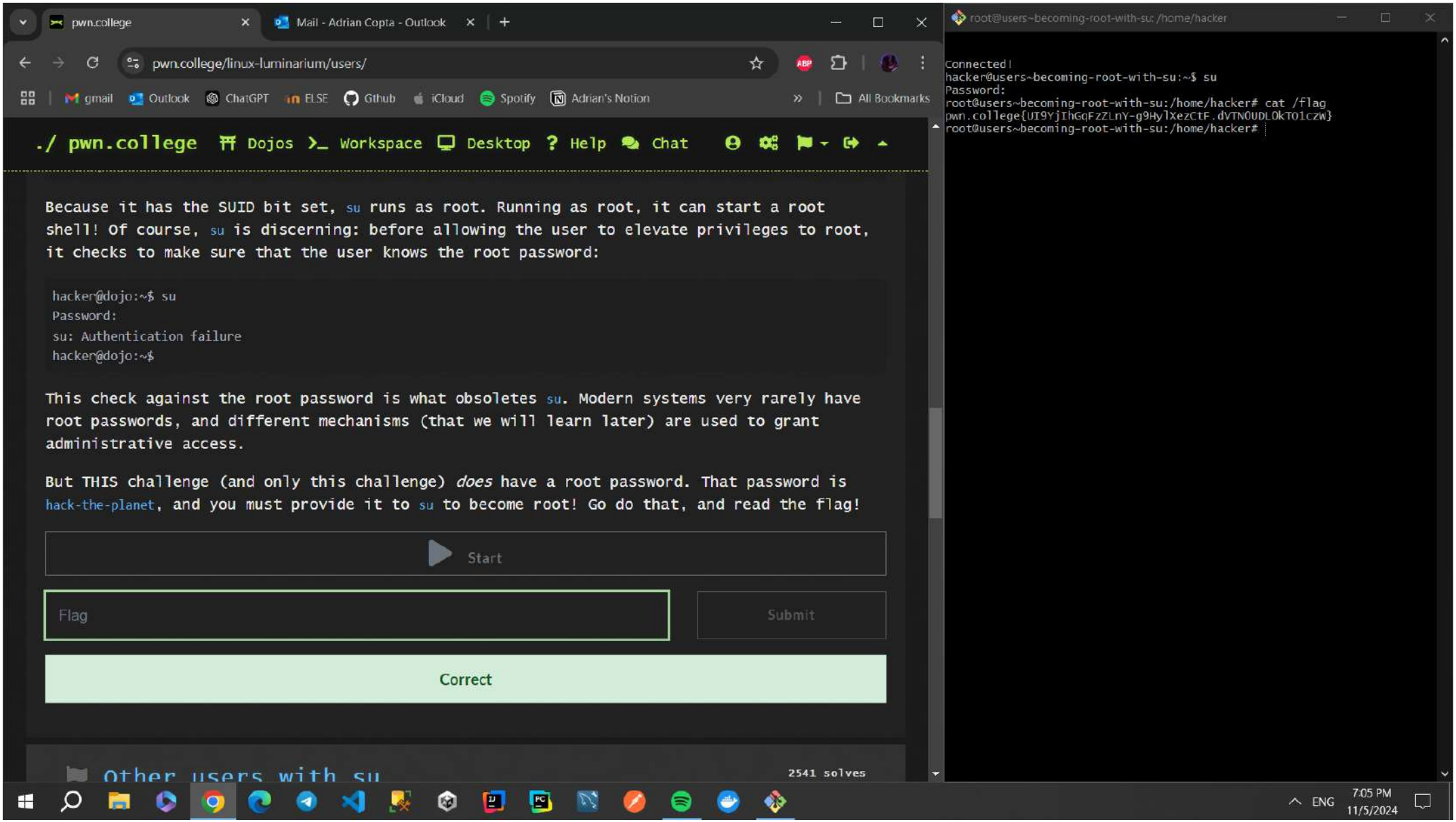

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Resources
Challenges
Changing File Ownership
2 hacking, 3104 solves
Groups and Files
2711 solves
Fun With Groups Names
2696 solves
Changing Permissions
1 hacking, 2695 solves
Executable Files
3027 solves
Permission Tweaking Practice
1 hacking, 2883 solves
hacker@permissions~changing-permissions: ~
Connected!
hacker@permissions~changing-permissions:~$ chmod go+r /flag
hacker@permissions~changing-permissions:~$ ls -l /flag
-r--r--r-- 1 root root 58 Nov 5 16:08 /flag
hacker@permissions~changing-permissions:~$ cat /flag
pwn.college{gKKqRILByWO_LqIpTj1QvMMeD4v.dNzNyUDLOkTO1czW}
hacker@permissions~changing-permissions:~$
ENG
6:13 PM
11/5/2024

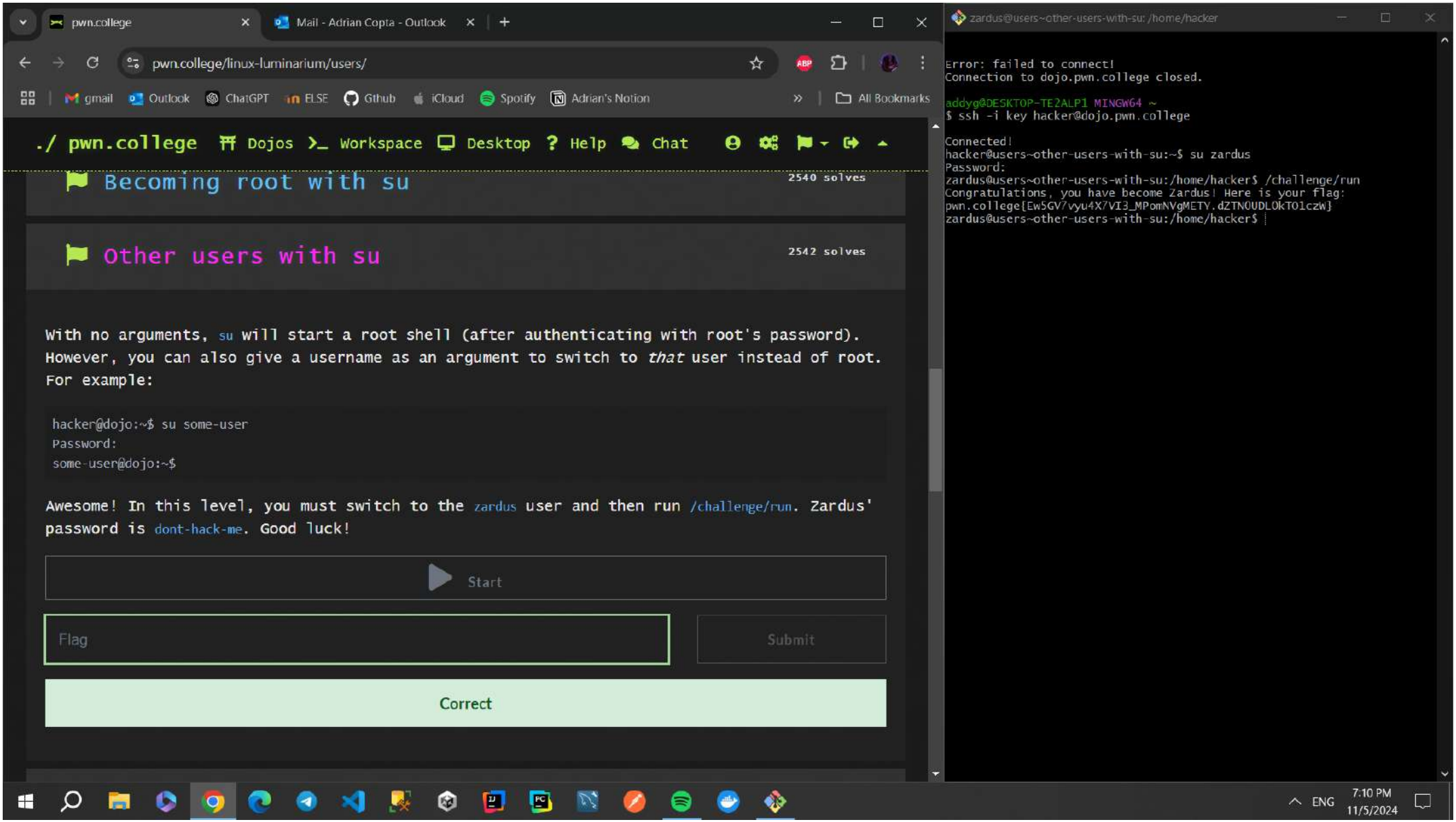

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
-rwxr-xr-- 1 root root 0 May 22 13:42 /challenge/run
hacker@dojo:~$ /challenge/run
bash: /challenge/run: Permission denied
hacker@dojo:~$
In this challenge, the /challenge/run program will give you the flag, but you must first make it executable! Remember your chmod, and get /challenge/run to tell you the flag!
Start
Flag
Submit
Correct
Permission Tweaking Practice
1 hacking, 2883 solves
Permissions Setting Practice
2707 solves
The SUID Bit
1 hacking, 2935 solves
hacker@permissions~executable-files: ~
Connected!
hacker@permissions~executable-files:~$ chmod o+x /challenge/run
hacker@permissions~executable-files:~$ ls -l /challenge/run
-r--r--r-x 1 hacker hacker 32 Oct 29 18:50 /challenge/run
hacker@permissions~executable-files:~$ /challenge/run
ssh-entrypoint: /challenge/run: Permission denied
hacker@permissions~executable-files:~$chchmod u+x /challenge/run
ssh-entrypoint: $'\302\203chmod': command not found
hacker@permissions~executable-files:~$ ls -l /challenge/run
-r--r--r-x 1 hacker hacker 32 Oct 29 18:50 /challenge/run
hacker@permissions~executable-files:~$ /challenge/run
ssh-entrypoint: /challenge/run: Permission denied
hacker@permissions~executable-files:~$ chmod u+x /challenge/run
hacker@permissions~executable-files:~$ ls -l /challenge/run
-r-xr--r-x 1 hacker hacker 32 Oct 29 18:50 /challenge/run
hacker@permissions~executable-files:~$ /challenge/run
Successful execution! Here is your flag:
pwn.college{AKIdQ4Yh6Sonqkvok6KD47UP9Mb.dJTM2QDL0kTO1czW}
hacker@permissions~executable-files:~$
ENG
5:16 PM
11/5/2024

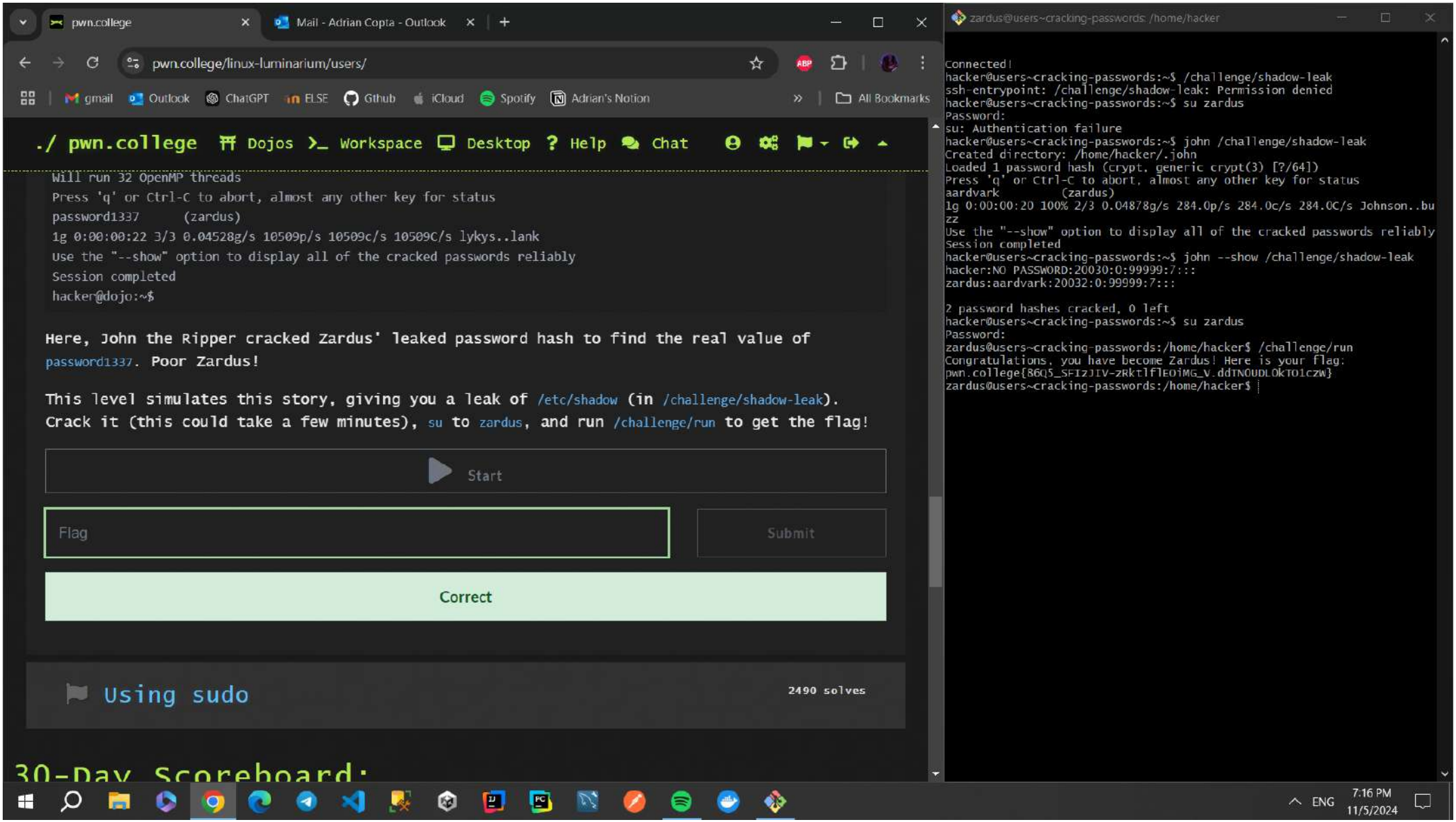

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Executable Files
3028 solves
Permission Tweaking Practice
1 hacking, 2884 solves
You think you can chmod? Let's practice!
This challenge will ask you to change the permissions of the /challenge/pwn file in specific ways a few times in a row. If you get the permissions wrong, the game will reset and you can try again. If you get the permissions right eight times in a row, the challenge will let you chmod /flag to make it readable for yourself :-) Launch /challenge/run to get started!
Start
Flag
Submit
Correct
Permissions Setting Practice
2707 solves
hacker@permissions~permission-tweaking-practice: ~
- the world doesn't have read permissions
- the world doesn't have write permissions
- the world doesn't have execute permissions
hacker@permissions~permission-tweaking-practice:~$ chmod u-r,u-w,u-x,o-r,o-w /challenge/pwn
You set the correct permissions!
Round 8 of 8!
Current permissions of "/challenge/pwn": ----w----
- the user doesn't have read permissions
- the user doesn't have write permissions
- the user doesn't have execute permissions
- the group doesn't have read permissions
* the group does have write permissions
- the group doesn't have execute permissions
- the world doesn't have read permissions
- the world doesn't have write permissions
- the world doesn't have execute permissions
Needed permissions of "/challenge/pwn": -wx-wx-wx
- the user doesn't have read permissions
* the user does have write permissions
* the user does have execute permissions
- the group doesn't have read permissions
* the group does have write permissions
* the group does have execute permissions
- the world doesn't have read permissions
* the world does have write permissions
* the world does have execute permissions
hacker@permissions~permission-tweaking-practice:~$ chmod u+w,u+x,g+x,o+w,o+x /challenge/pwn
You set the correct permissions!
You've solved all 8 rounds! I have changed the ownership of the /flag file so that you can 'chmod' it. You won't be able to read it until you make it readable with chmod!
Current permissions of "/flag": ---------
- the user doesn't have read permissions
- the user doesn't have write permissions
- the user doesn't have execute permissions
- the group doesn't have read permissions
- the group doesn't have write permissions
- the group doesn't have execute permissions
- the world doesn't have read permissions
- the world doesn't have write permissions
- the world doesn't have execute permissions
hacker@permissions~permission-tweaking-practice:~$ chmod u+r /flag
ssh-entrypoint: $'\302\203chmod': command not found
hacker@permissions~permission-tweaking-practice:~$ ls -l /flag
---------- 1 hacker hacker 58 Nov 5 16:21 /flag
hacker@permissions~permission-tweaking-practice:~$ cat /flag
cat: /flag: Permission denied
hacker@permissions~permission-tweaking-practice:~$ chmod u+r /flag
ssh-entrypoint: $'\302\203chmod': command not found
hacker@permissions~permission-tweaking-practice:~$ chmod u+r /flag
hacker@permissions~permission-tweaking-practice:~$ cat /flag
pwn.college{ES_nuduko8DBKDtN8erNglZnQ-r.dBTM2QDLOkTO1CzW}
hacker@permissions~permission-tweaking-practice:~$
ENG
6:47 PM
11/5/2024

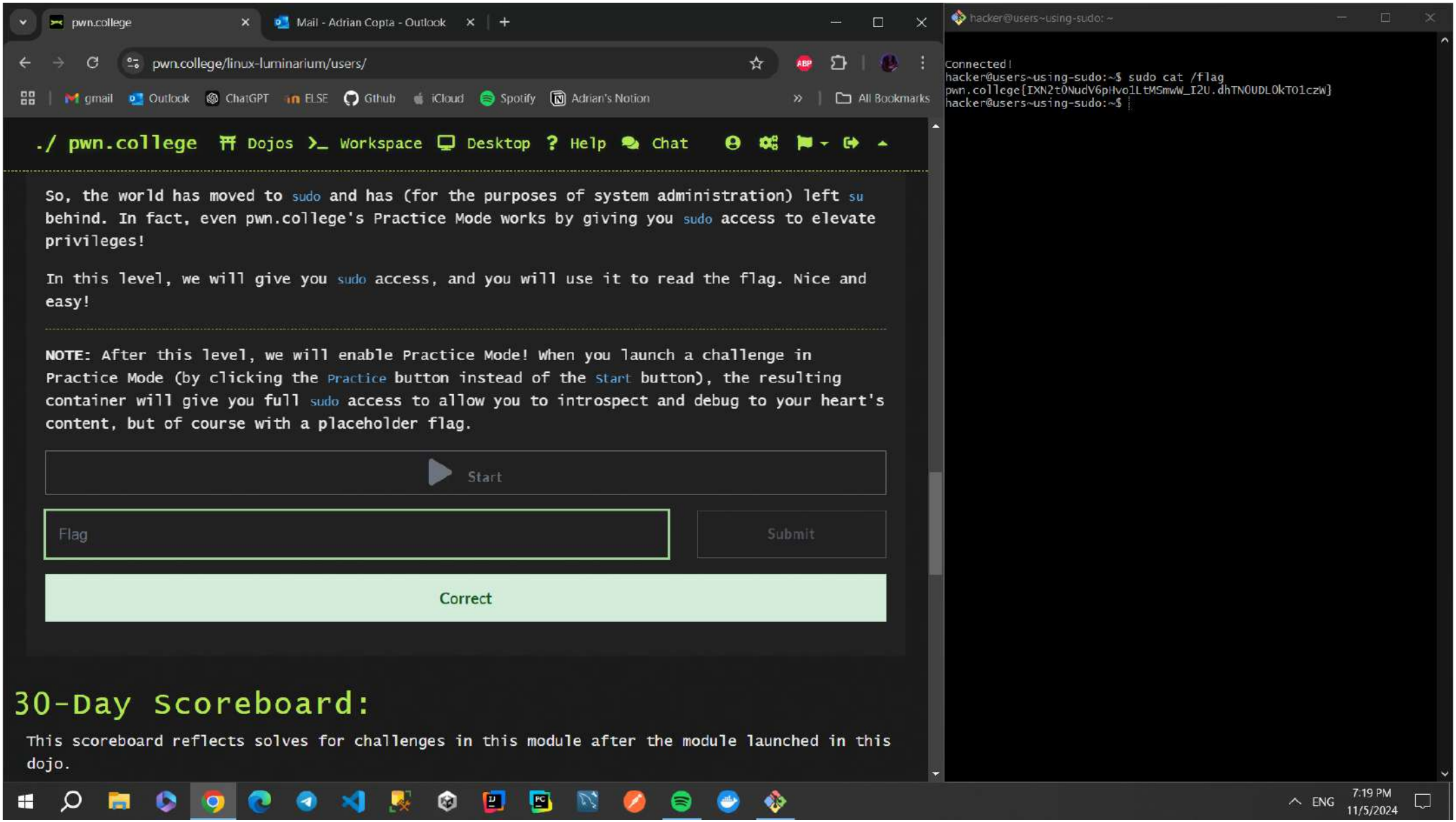

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
achieve this by chaining multiple modes to chmod with ,!
chmod u=rw,g=r /challenge/pwn will set the user permissions to read and write, and the group permissions to read-only
chmod a=r,u=rw /challenge/pwn will set the user permissions to read and write, and the group and world permissions to read-only
Additionally, you can zero out permissions with -:
chmod u=rw,g=r,o=- /challenge/pwn will set the user permissions to read and write, the group permissions to read-only, and the world permissions to nothing at all
Keep in mind, that -, appearing after = is in a different context than if it appeared directly after the u, g, or o (in which case, it would cause specific bits to be removed, not everything).
This level extends the previous level by requesting more radical permission changes, which you will need = and ,-chaining to achieve. Good luck!
Start
Flag
Submit
Correct
hacker@permissions~permissions-setting-practice: ~
Needed permissions of "/challenge/pwn": rwxr-xr-x
* the user does have read permissions
* the user does have write permissions
* the user does have execute permissions
* the group does have read permissions
- the group doesn't have write permissions
* the group does have execute permissions
* the world does have read permissions
- the world doesn't have write permissions
* the world does have execute permissions
hacker@permissions~permissions-setting-practice:~$ chmod u=rwx,g=rx,o=rx /challenge/pwn
You set the correct permissions!
Round 8 of 8!
Current permissions of "/challenge/pwn": rwxr-xr-x
* the user does have read permissions
* the user does have write permissions
* the user does have execute permissions
* the group does have read permissions
- the group doesn't have write permissions
* the group does have execute permissions
* the world does have read permissions
- the world doesn't have write permissions
* the world does have execute permissions
Needed permissions of "/challenge/pwn": rwxrwxr-x
* the user does have read permissions
* the user does have write permissions
* the user does have execute permissions
* the group does have read permissions
* the group does have write permissions
* the group does have execute permissions
* the world does have read permissions
- the world doesn't have write permissions
* the world does have execute permissions
hacker@permissions~permissions-setting-practice:~$ chmod g=rwx /challenge/pwn
You set the correct permissions!
You've solved all 8 rounds! I have changed the ownership of the /flag file so that you can 'chmod' it. You won't be able to read it until you make it readable with chmod!
Current permissions of "/flag": ---------
- the user doesn't have read permissions
- the user doesn't have write permissions
- the user doesn't have execute permissions
- the group doesn't have read permissions
- the group doesn't have write permissions
- the group doesn't have execute permissions
- the world doesn't have read permissions
- the world doesn't have write permissions
- the world doesn't have execute permissions
hacker@permissions~permissions-setting-practice:~$ chmod u+r /flag
hacker@permissions~permissions-setting-practice:~$ cat /flag
pwn.college{EOYzkTnhtioqiL5nfZxFRjiUvi6.dNTM5QDLOkTO1czW}
hacker@permissions~permissions-setting-practice:~$
ENG
6:55 PM
11/5/2024

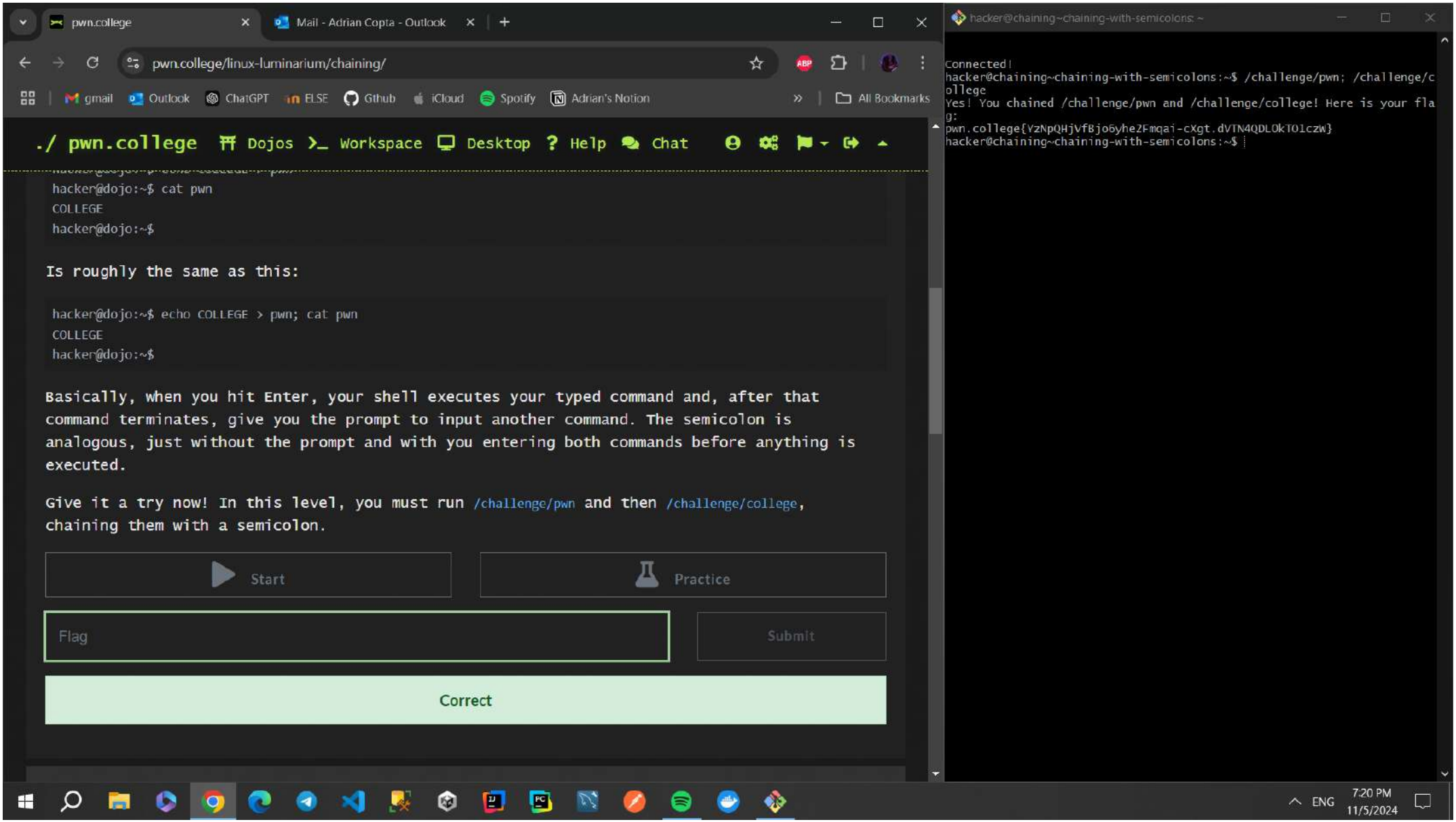

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/permissions/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$
The s part in place of the executable bit means that the program is executable with SUID. It means that, regardless of what user runs the program (as long as they have executable permissions), the program will execute as the owner user (in this case, the root user).
As the owner of a file, you can set a file's SUID bit by using chmod:
chmod u+s [program]
But be careful! Giving the SUID bit to an executable owned by root can give attackers a possible attack vector to become root. You will learn more about this in the Program Misuse module.
Now, we are going to let you add the SUID bit to the /challenge/getroot program in order to spawn a root shell for you to cat the flag yourself!
Start
Flag
Submit
Correct
root@permissions~the-suid-bit: ~
Connected!
hacker@permissions~the-suid-bit:~$ chmod u+s /challenge/getroot
hacker@permissions~the-suid-bit:~$ cat /challenge/getroot
#!/opt/pwn.college/bash
fold -s <<< "SUCCESS! You have set the suid bit on this program, and it is running as root! Here is your shell..."
exec /bin/bash
hacker@permissions~the-suid-bit:~$ cat /falg
cat: /falg: No such file or directory
hacker@permissions~the-suid-bit:~$ cat /flag
cat: /flag: Permission denied
hacker@permissions~the-suid-bit:~$ /challenge/getroot
SUCCESS! You have set the suid bit on this program, and it is running as root!
Here is your shell...
root@permissions~the-suid-bit:~# cat /flag
pwn.college{w1zwQurKQMM1zyiAm9YV-z4QsSe.dNTM2QDL0kTO1czW}
root@permissions~the-suid-bit:~#
ENG
6:59 PM
11/5/2024

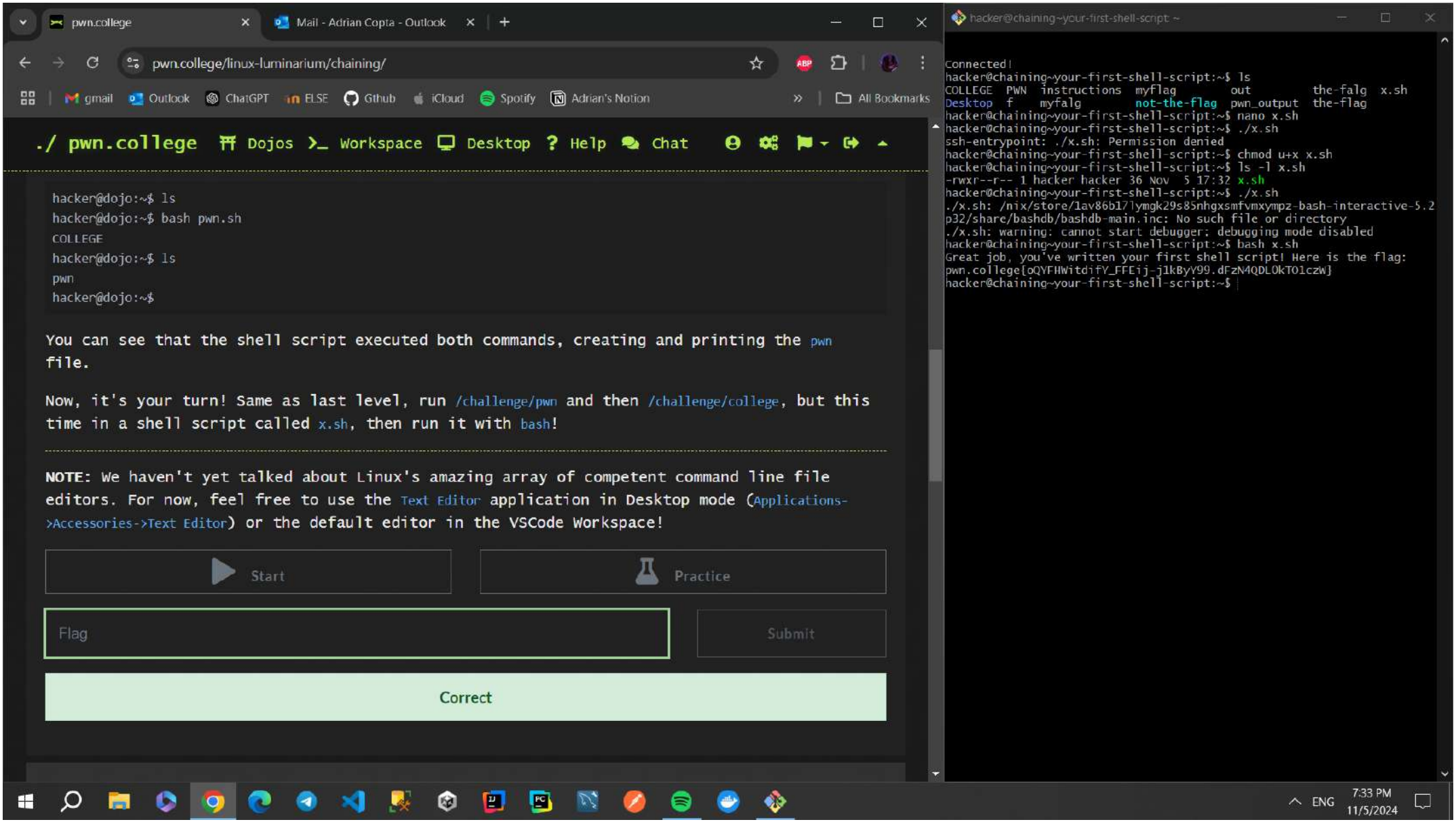

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/users/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Because it has the SUID bit set, su runs as root. Running as root, it can start a root shell! Of course, su is discerning: before allowing the user to elevate privileges to root, it checks to make sure that the user knows the root password:
hacker@dojo:~$ su
Password:
su: Authentication failure
hacker@dojo:~$
This check against the root password is what obsoletes su. Modern systems very rarely have root passwords, and different mechanisms (that we will learn later) are used to grant administrative access.
But THIS challenge (and only this challenge) does have a root password. That password is hack-the-planet, and you must provide it to su to become root! Go do that, and read the flag!
Start
Flag
Submit
Correct
Other users with su
2541 solves
root@users~becoming-root-with-su: /home/hacker
Connected!
hacker@users~becoming-root-with-su:~$ su
Password:
root@users~becoming-root-with-su:/home/hacker# cat /flag
pwn.college{UI9YjThGqFzZLnY-g9HylXezCtF.dVTN0UDL0kTO1czW}
root@users~becoming-root-with-su:/home/hacker#
ENG
7:05 PM
11/5/2024

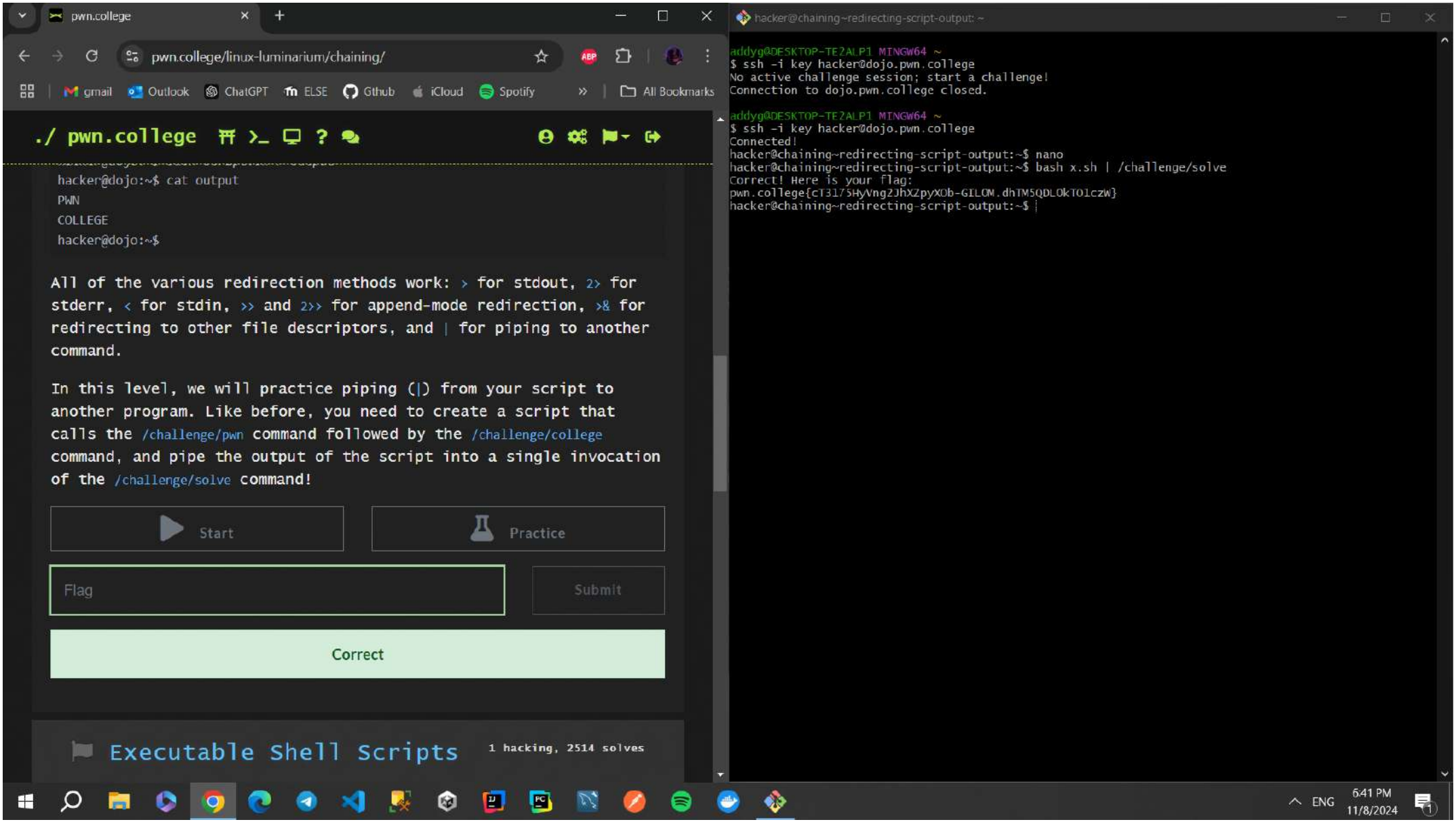

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/users/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Becoming root with su
2540 solves
Other users with su
2542 solves
With no arguments, su will start a root shell (after authenticating with root's password). However, you can also give a username as an argument to switch to that user instead of root. For example:
hacker@dojo:~$ su some-user
Password:
some-user@dojo:~$
Awesome! In this level, you must switch to the zardus user and then run /challenge/run. Zardus' password is dont-hack-me. Good luck!
Start
Flag
Submit
Correct
zardus@users~other-users-with-su: /home/hacker
Error: failed to connect!
Connection to dojo.pwn.college closed.
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
Connected!
hacker@users~other-users-with-su:~$ su zardus
Password:
zardus@users~other-users-with-su:/home/hacker$ /challenge/run
Congratulations, you have become Zardus! Here is your flag:
pwn.college{Ew5GV7vyu4X7VI3_MPomNVgMETY.dZTN0UDL0kTO1czW}
zardus@users~other-users-with-su:/home/hacker$
ENG
7:10 PM
11/5/2024

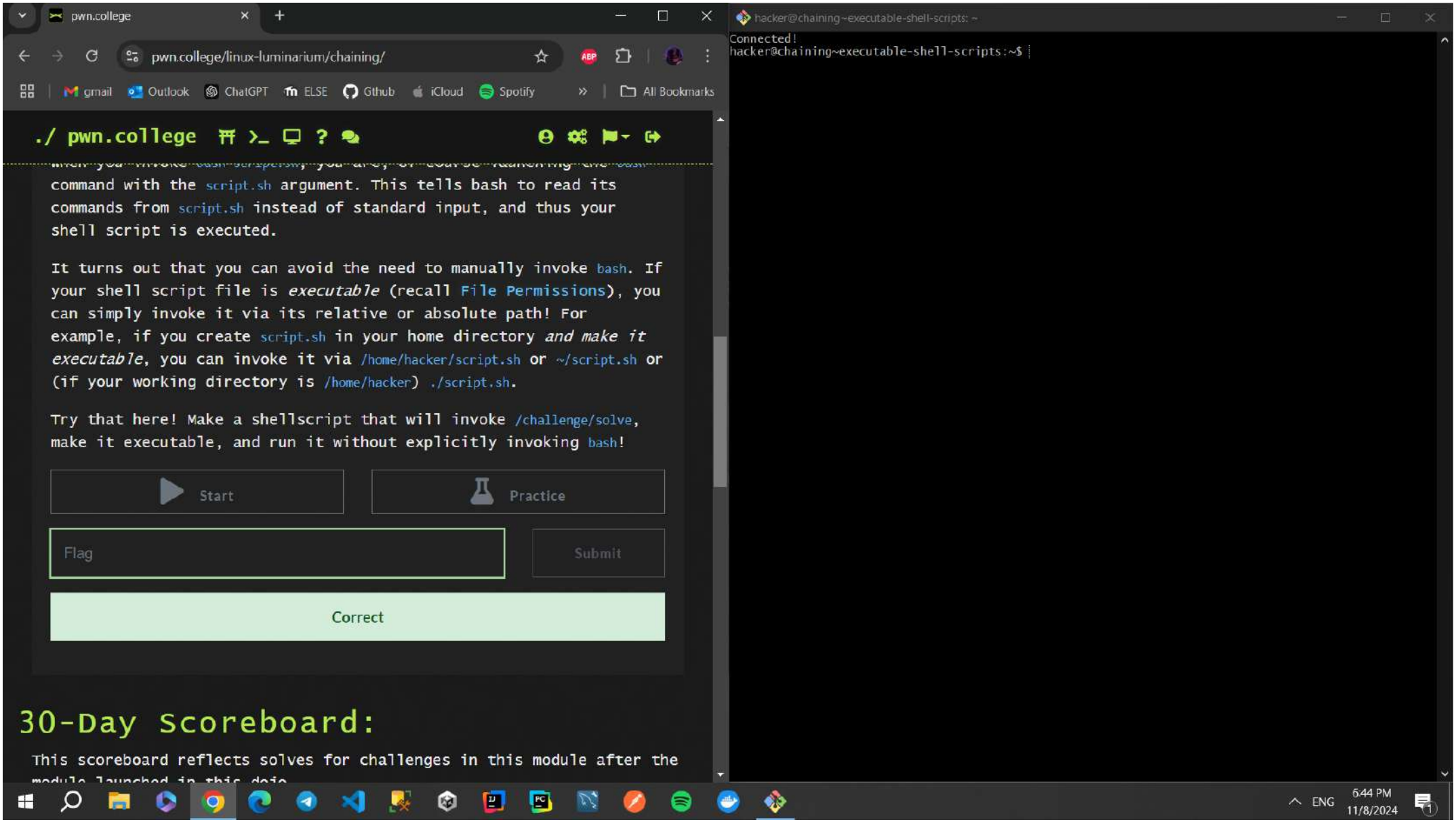

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/users/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Will run 32 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password1337 (zardus)
1g 0:00:00:22 3/3 0.04528g/s 10509p/s 10509c/s 10509C/s lykys..lank
Use the "--show" option to display all of the cracked passwords reliably
Session completed
hacker@dojo:~$
Here, John the Ripper cracked Zardus' leaked password hash to find the real value of password1337. Poor Zardus!
This level simulates this story, giving you a leak of /etc/shadow (in /challenge/shadow-leak).
Crack it (this could take a few minutes), su to zardus, and run /challenge/run to get the flag!
Start
Flag
Submit
Correct
Using sudo
2490 solves
30-Day Scoreboard:
zardus@users~cracking-passwords: /home/hacker
Connected!
hacker@users~cracking-passwords:~$ /challenge/shadow-leak
ssh-entrypoint: /challenge/shadow-leak: Permission denied
hacker@users~cracking-passwords:~$ su zardus
Password:
su: Authentication failure
hacker@users~cracking-passwords:~$ john /challenge/shadow-leak
Created directory: /home/hacker/.john
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
aardvark (zardus)
1g 0:00:00:20 100% 2/3 0.04878g/s 284.0p/s 284.0c/s 284.0C/s Johnson..buzz
Use the "--show" option to display all of the cracked passwords reliably
Session completed
hacker@users~cracking-passwords:~$ john --show /challenge/shadow-leak
hacker:NO PASSWORD:20030:0:99999:7:::
zardus:aardvark:20032:0:99999:7:::
2 password hashes cracked, 0 left
hacker@users~cracking-passwords:~$ su zardus
Password:
zardus@users~cracking-passwords:/home/hacker$ /challenge/run
Congratulations, you have become Zardus! Here is your flag:
pwn.college{86Q5_SFIzJIV-zRktlflEOiMG_V.ddTNOUDLOkTO1czW}
zardus@users~cracking-passwords:/home/hacker$
ENG
7:16 PM
11/5/2024

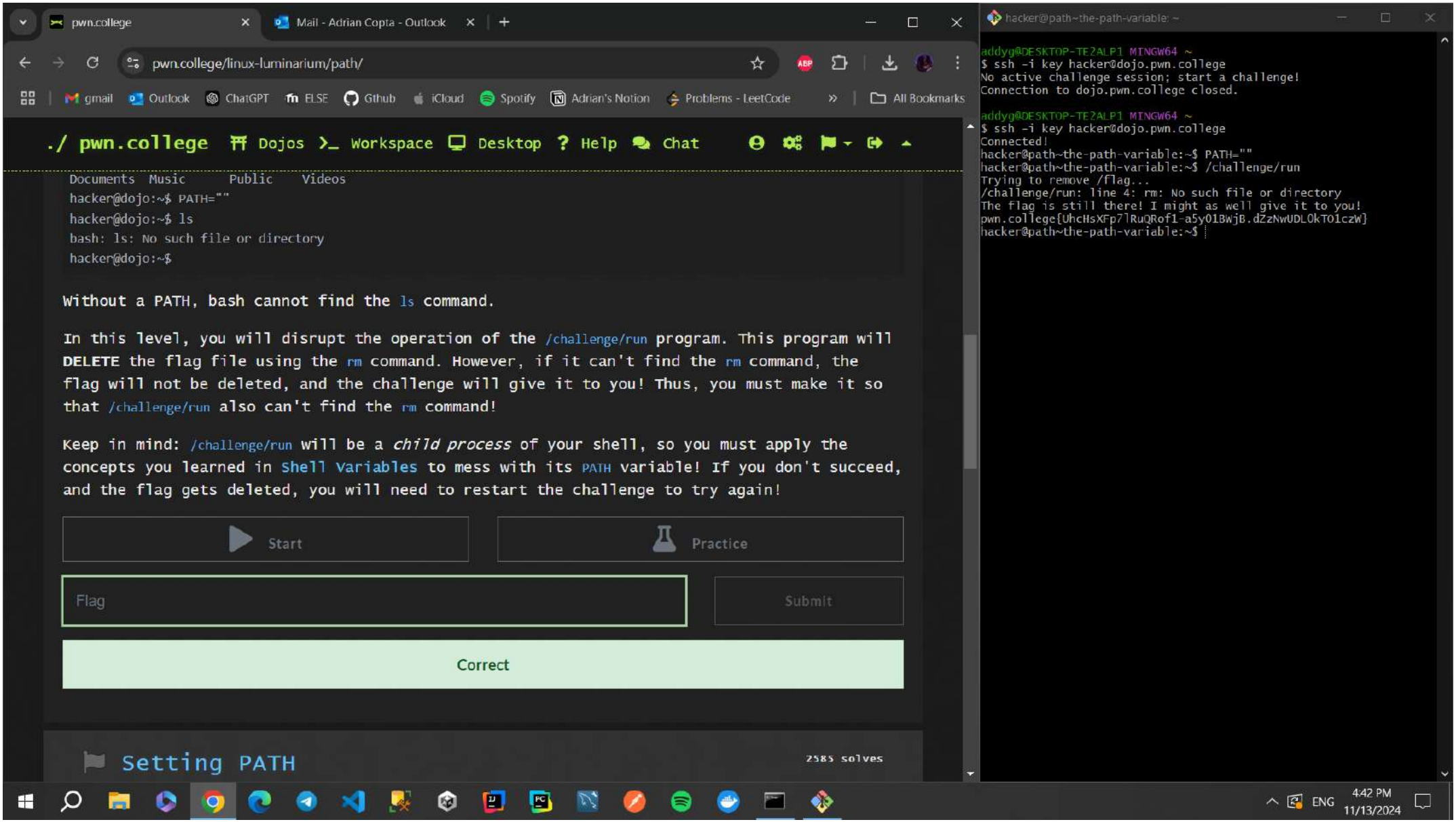

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/users/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
So, the world has moved to sudo and has (for the purposes of system administration) left su behind. In fact, even pwn.college's Practice Mode works by giving you sudo access to elevate privileges!
In this level, we will give you sudo access, and you will use it to read the flag. Nice and easy!
NOTE: After this level, we will enable Practice Mode! When you launch a challenge in Practice Mode (by clicking the Practice button instead of the Start button), the resulting container will give you full sudo access to allow you to introspect and debug to your heart's content, but of course with a placeholder flag.
Start
Flag
Submit
Correct
30-Day Scoreboard:
This scoreboard reflects solves for challenges in this module after the module launched in this dojo.
hacker@users~using-sudo: ~
Connected!
hacker@users~using-sudo:~$ sudo cat /flag
pwn.college{IXN2t0NudV6pHvo1LtMSmwW_I2U.dhTN0UDL0kTO1czW}
hacker@users~using-sudo:~$
ENG
7:19 PM
11/5/2024

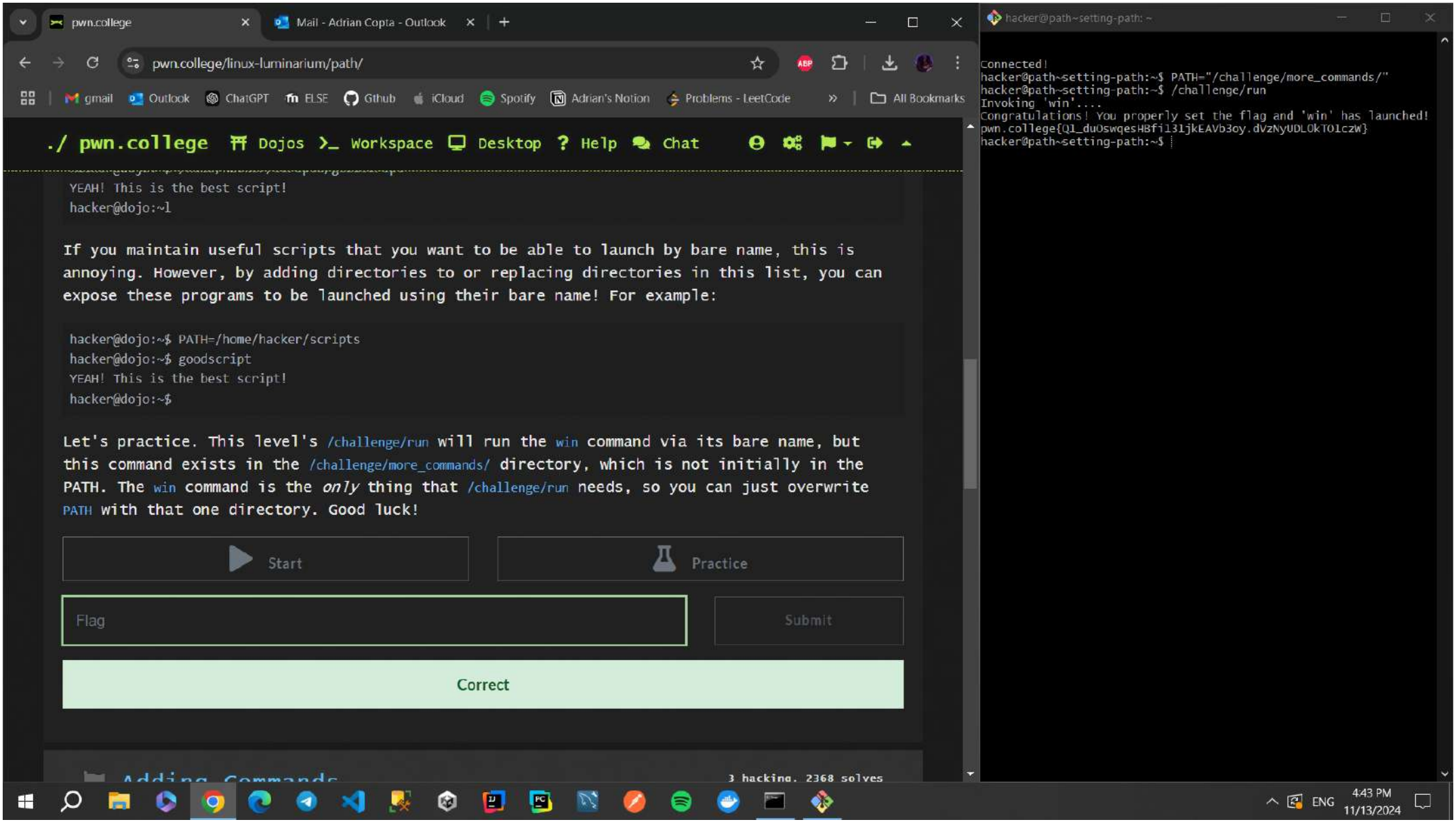

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/chaining/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ cat pwn
COLLEGE
hacker@dojo:~$
Is roughly the same as this:
hacker@dojo:~$ echo COLLEGE > pwn; cat pwn
COLLEGE
hacker@dojo:~$
Basically, when you hit Enter, your shell executes your typed command and, after that command terminates, give you the prompt to input another command. The semicolon is analogous, just without the prompt and with you entering both commands before anything is executed.
Give it a try now! In this level, you must run /challenge/pwn and then /challenge/college, chaining them with a semicolon.
Start
Practice
Flag
Submit
Correct
hacker@chaining~chaining-with-semicolons: ~
Connected!
hacker@chaining~chaining-with-semicolons:~$ /challenge/pwn; /challenge/college
Yes! You chained /challenge/pwn and /challenge/college! Here is your flag:
pwn.college{YzNpQHjVfBjo6yheZFmqai-cXgt.dVTN4QDLOkTO1czW}
hacker@chaining~chaining-with-semicolons:~$
ENG
7:20 PM
11/5/2024

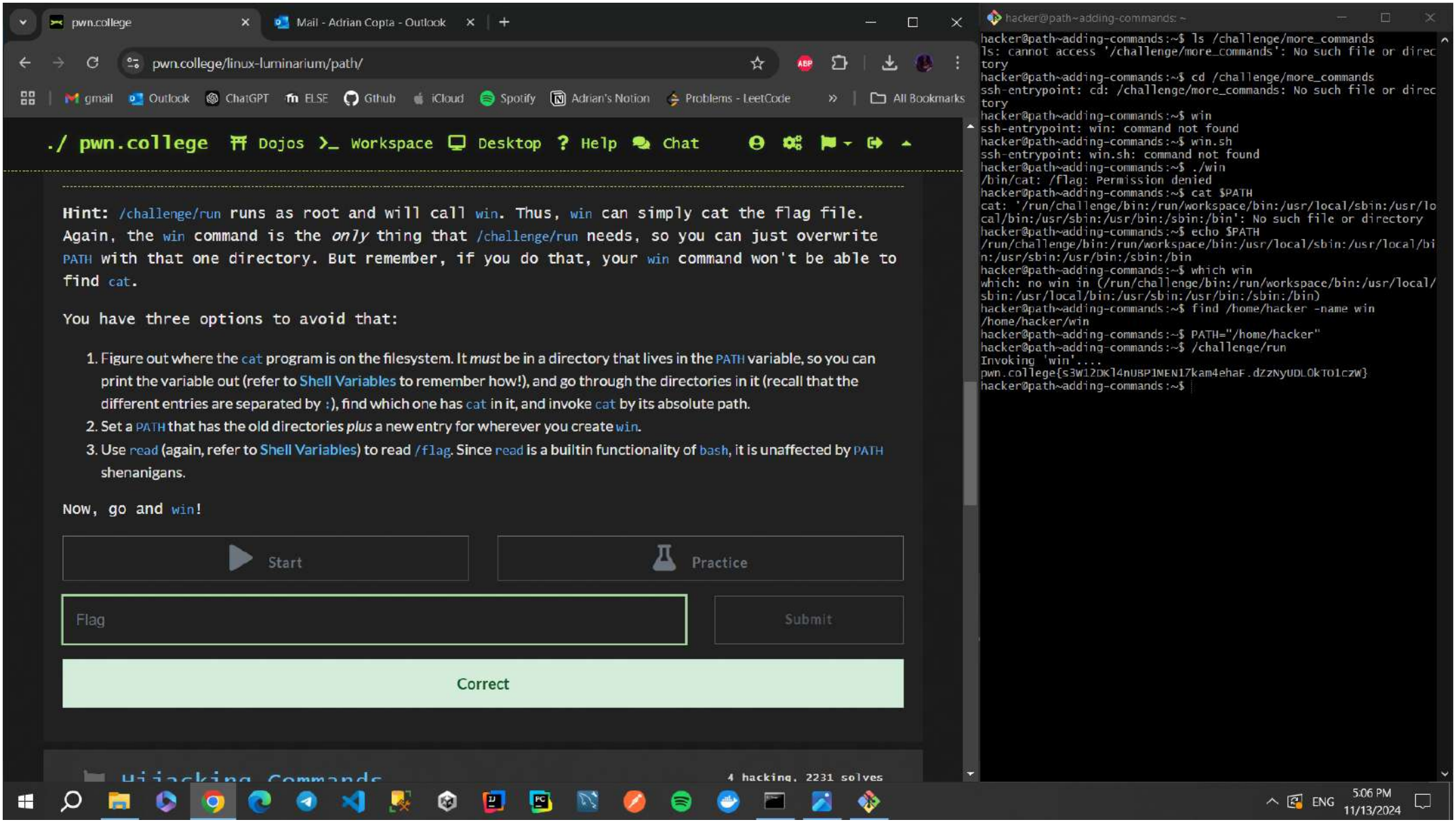

pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/chaining/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
hacker@dojo:~$ ls
hacker@dojo:~$ bash pwn.sh
COLLEGE
hacker@dojo:~$ ls
pwn
hacker@dojo:~$
You can see that the shell script executed both commands, creating and printing the pwn file.
Now, it's your turn! Same as last level, run /challenge/pwn and then /challenge/college, but this time in a shell script called x.sh, then run it with bash!
NOTE: We haven't yet talked about Linux's amazing array of competent command line file editors. For now, feel free to use the Text Editor application in Desktop mode (Applications->Accessories->Text Editor) or the default editor in the VSCode Workspace!
Start
Practice
Flag
Submit
Correct
hacker@chaining~your-first-shell-script: ~
Connected!
hacker@chaining~your-first-shell-script:~$ ls
COLLEGE PWN instructions myflag out the-falg x.sh
Desktop f myfalg not-the-flag pwn_output the-flag
hacker@chaining~your-first-shell-script:~$ nano x.sh
hacker@chaining~your-first-shell-script:~$ ./x.sh
ssh-entrypoint: ./x.sh: Permission denied
hacker@chaining~your-first-shell-script:~$ chmod u+x x.sh
hacker@chaining~your-first-shell-script:~$ ls -l x.sh
-rwxr--r-- 1 hacker hacker 36 Nov 5 17:32 x.sh
hacker@chaining~your-first-shell-script:~$ ./x.sh
./x.sh: /nix/store/1av86b17lymgk29s85nhgxsmfvmxympz-bash-interactive-5.2p32/share/bashdb/bashdb-main.inc: No such file or directory
./x.sh: warning: cannot start debugger; debugging mode disabled
hacker@chaining~your-first-shell-script:~$ bash x.sh
Great job, you've written your first shell script! Here is the flag:
pwn.college{oQYFHWitdifV_FFEij-j1kByY99.dFzN4QDLOkTO1czW}
hacker@chaining~your-first-shell-script:~$
ENG
7:33 PM
11/5/2024

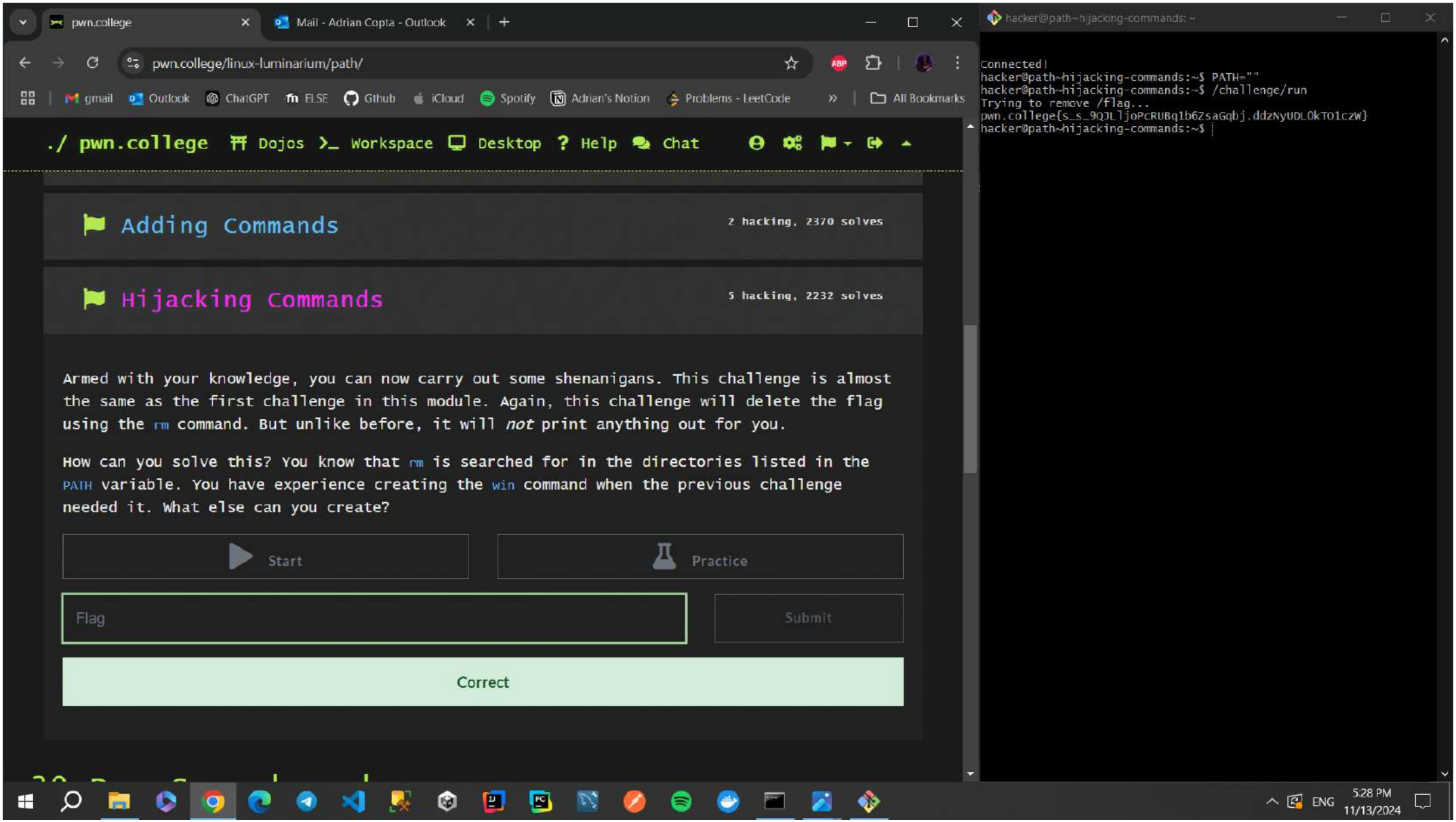

pwn.college
pwn.college/linux-luminarium/chaining/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
All Bookmarks
./ pwn.college
hacker@dojo:~$ cat output
PWN
COLLEGE
hacker@dojo:~$
All of the various redirection methods work: > for stdout, 2> for stderr, < for stdin, >> and 2>> for append-mode redirection, >& for redirecting to other file descriptors, and | for piping to another command.
In this level, we will practice piping (|) from your script to another program. Like before, you need to create a script that calls the /challenge/pwn command followed by the /challenge/college command, and pipe the output of the script into a single invocation of the /challenge/solve command!
Start
Practice
Flag
Submit
Correct
Executable Shell Scripts
1 hacking, 2514 solves
hacker@chaining~redirecting-script-output: ~
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
No active challenge session; start a challenge!
Connection to dojo.pwn.college closed.
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
Connected!
hacker@chaining~redirecting-script-output:~$ nano
hacker@chaining~redirecting-script-output:~$ bash x.sh | /challenge/solve
Correct! Here is your flag:
pwn.college{cT3175HyVng2JhXZpyXOb-GILOM.dhTM5QDLOkTO1czW}
hacker@chaining~redirecting-script-output:~$
ENG
6:41 PM
11/8/2024


pwn.college
pwn.college/linux-luminarium/chaining/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
All Bookmarks
./ pwn.college
command with the script.sh argument. This tells bash to read its commands from script.sh instead of standard input, and thus your shell script is executed.
It turns out that you can avoid the need to manually invoke bash. If your shell script file is executable (recall File Permissions), you can simply invoke it via its relative or absolute path! For example, if you create script.sh in your home directory and make it executable, you can invoke it via /home/hacker/script.sh or ~/script.sh or (if your working directory is /home/hacker) ./script.sh.
Try that here! Make a shellscript that will invoke /challenge/solve, make it executable, and run it without explicitly invoking bash!
Start
Practice
Flag
Submit
Correct
30-Day Scoreboard:
This scoreboard reflects solves for challenges in this module after the
hacker@chaining~executable-shell-scripts: ~
Connected!
hacker@chaining~executable-shell-scripts:~$
ENG
6:44 PM
11/8/2024


pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/path/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Documents Music Public Videos
hacker@dojo:~$ PATH=""
hacker@dojo:~$ ls
bash: ls: No such file or directory
hacker@dojo:~$
Without a PATH, bash cannot find the ls command.
In this level, you will disrupt the operation of the /challenge/run program. This program will DELETE the flag file using the rm command. However, if it can't find the rm command, the flag will not be deleted, and the challenge will give it to you! Thus, you must make it so that /challenge/run also can't find the rm command!
Keep in mind: /challenge/run will be a child process of your shell, so you must apply the concepts you learned in Shell Variables to mess with its PATH variable! If you don't succeed, and the flag gets deleted, you will need to restart the challenge to try again!
Start
Practice
Flag
Submit
Correct
Setting PATH
2585 solves
hacker@path~the-path-variable: ~
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
No active challenge session; start a challenge!
Connection to dojo.pwn.college closed.
addyg@DESKTOP-TE2ALP1 MINGW64 ~
$ ssh -i key hacker@dojo.pwn.college
Connected!
hacker@path~the-path-variable:~$ PATH=""
hacker@path~the-path-variable:~$ /challenge/run
Trying to remove /flag...
/challenge/run: line 4: rm: No such file or directory
The flag is still there! I might as well give it to you!
pwn.college{UhcHsXFp7lRuQRof1-a5yO1BWjB.dZzNwUDLOkTO1czW}
hacker@path~the-path-variable:~$
ENG
4:42 PM
11/13/2024


pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/path/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
YEAH! This is the best script!
hacker@dojo:~$
If you maintain useful scripts that you want to be able to launch by bare name, this is annoying. However, by adding directories to or replacing directories in this list, you can expose these programs to be launched using their bare name! For example:
hacker@dojo:~$ PATH=/home/hacker/scripts
hacker@dojo:~$ goodscript
YEAH! This is the best script!
hacker@dojo:~$
Let's practice. This level's /challenge/run will run the win command via its bare name, but this command exists in the /challenge/more_commands/ directory, which is not initially in the PATH. The win command is the only thing that /challenge/run needs, so you can just overwrite PATH with that one directory. Good luck!
Start
Practice
Flag
Submit
Correct
3 hacking, 2368 solves
hacker@path~setting-path: ~
Connected!
hacker@path~setting-path:~$ PATH="/challenge/more_commands/"
hacker@path~setting-path:~$ /challenge/run
Invoking 'win'....
Congratulations! You properly set the flag and 'win' has launched!
pwn.college{Q1_duOswqesHBfi131jkEAVb3oy.dVzNyUDLOkTO1czW}
hacker@path~setting-path:~$
ENG
4:43 PM
11/13/2024


pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/path/
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Hint: /challenge/run runs as root and will call win. Thus, win can simply cat the flag file. Again, the win command is the only thing that /challenge/run needs, so you can just overwrite PATH with that one directory. But remember, if you do that, your win command won't be able to find cat.
You have three options to avoid that:
1. Figure out where the cat program is on the filesystem. It must be in a directory that lives in the PATH variable, so you can print the variable out (refer to Shell Variables to remember how!), and go through the directories in it (recall that the different entries are separated by :), find which one has cat in it, and invoke cat by its absolute path.
2. Set a PATH that has the old directories plus a new entry for wherever you create win.
3. Use read (again, refer to Shell Variables) to read /flag. Since read is a builtin functionality of bash, it is unaffected by PATH shenanigans.
Now, go and win!
Start
Practice
Flag
Submit
Correct
4 hacking, 2231 solves
hacker@path~adding-commands: ~
hacker@path~adding-commands:~$ ls /challenge/more_commands
ls: cannot access '/challenge/more_commands': No such file or directory
hacker@path~adding-commands:~$ cd /challenge/more_commands
ssh-entrypoint: cd: /challenge/more_commands: No such file or directory
hacker@path~adding-commands:~$ win
ssh-entrypoint: win: command not found
hacker@path~adding-commands:~$ win.sh
ssh-entrypoint: win.sh: command not found
hacker@path~adding-commands:~$ ./win
/bin/cat: /flag: Permission denied
hacker@path~adding-commands:~$ cat $PATH
cat: '/run/challenge/bin:/run/workspace/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin': No such file or directory
hacker@path~adding-commands:~$ echo $PATH
/run/challenge/bin:/run/workspace/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
hacker@path~adding-commands:~$ which win
which: no win in (/run/challenge/bin:/run/workspace/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin)
hacker@path~adding-commands:~$ find /home/hacker -name win
/home/hacker/win
hacker@path~adding-commands:~$ PATH="/home/hacker"
hacker@path~adding-commands:~$ /challenge/run
Invoking 'win'....
pwn.college{s3W12DKl4nUBP1MEN17kam4ehaF.dZzNyUDLOkTO1czW}
hacker@path~adding-commands:~$
ENG
5:06 PM
11/13/2024


pwn.college
Mail - Adrian Copta - Outlook
pwn.college/linux-luminarium/path/
gmail
Outlook
ChatGPT
ELSE
Github
iCloud
Spotify
Adrian's Notion
Problems - LeetCode
All Bookmarks
./ pwn.college
Dojos
Workspace
Desktop
Help
Chat
Adding Commands
2 hacking, 2370 solves
Hijacking Commands
5 hacking, 2232 solves
Armed with your knowledge, you can now carry out some shenanigans. This challenge is almost the same as the first challenge in this module. Again, this challenge will delete the flag using the rm command. But unlike before, it will not print anything out for you.
How can you solve this? You know that rm is searched for in the directories listed in the PATH variable. You have experience creating the win command when the previous challenge needed it. What else can you create?
Start
Practice
Flag
Submit
Correct
hacker@path~hijacking-commands: ~
Connected!
hacker@path~hijacking-commands:~$ PATH=""
hacker@path~hijacking-commands:~$ /challenge/run
Trying to remove /flag...
pwn.college{s_s_9QJLljoPcRUBq1b6ZsaGqbj.ddzNyUDL0kTO1czW}
hacker@path~hijacking-commands:~$
ENG
5:28 PM
11/13/2024