NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

You are asked to write a fail-safe read-file function that receives a text file path, reads its contents, and returns a string. If the file does not exist or cannot be read, the function should return an empty string. In the following implementation, the function `read_or_empty(path)` checks the existence and permission of the given file path before reading it. The `os.access(path, mode)` function (from Python's standard library) used below checks for existence and read permission of the file. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
01   import os
02
03   def read_or_empty(path):
04     content = ""
05
06     if os.access(path, os.R_OK): # Is file readable?
07       with open(path) as fileobj:
08         content = fileobj.read()
09
10     return content
```

Questions:

1. What will the program do when executed?

2. If one calls the `read_or_empty` function with a file path, which of the following is correct?

a. If the file exists and is readable, the function returns the file content.
b. If the function returns an empty string, it means the file did not exist or was not readable.
c. both A and B
d. None of the above

[Other statistical questions will be imported here while creating the survey.]

NOTE: ANSWER IS TO BE SHOWN TO THE PUZZLE TAKER AT THE END OF THE SESSION.

ANSWER:

1. The function will first check if the file exists and reads the permissions for the given file path. If the file exists and is readable, the function reads the contents of the file and returns the string. If not, the function returns an empty string.

2. d.

The implementation is vulnerable to TOCTTOU (time of check to time of use). In the time interval between checking for access and actually opening/reading the file, the permission for the file could have changed, or the file could have been deleted by some other process. Thus, the first answer is not generally true. The second answer is false as well, because the file might exist and be readable, but might have no content. The implementation should be rewritten with respect to the accepted practice that it is "easier to ask for forgiveness than permission." This means, instead of checking for permission, complete the function, and if anything is wrong you will get an exception/error.

NOTE: THE REST OF THIS DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO THE PUZZLE TAKERS.

TAGS:

Python, os.access, io-operation, file-operation, race-condition

CATEGORIES:

Blindspot - YES
Type - File
Number of distinct functions - 3
Number of total functions - 3
Blindspot function - `os.access()`
Function call omitted - NO
Blindspot type - TOCTTOU
Number of parameters in the blindspot function - 2 parameters
Cyclomatic complexity - 3

NAME:

os.access(path, mode)

DESCRIPTION:

Use the real uid/gid to test for access to path. Note that most operations will use the effective uid/gid, therefore this routine can be used in a suid/sgid environment to test if the invoking user has the specified access to path. `Mode` should be F_OK to test the existence of the path, or it can be the inclusive OR of one or more of R_OK, W_OK, and X_OK to test permissions. Return True if access is allowed, False if not.

BLINDSPOT:
This code is vulnerable to the TOCTTOU vulnerability. Using access() to check if a user is authorized to open a file before invoking an open(), allows an adversary to manipulate the file between the checking and opening steps. It's preferable to use EAFP techniques. Check out the MORE INFORMATION section.

EXAMPLES:
Below is a simple example showing how to check for a read permission, and then read from that file. If permission is denied, it will return a string "Cannot open file!"

```
if os.access("myfile", os.R_OK):
    with open("myfile") as fp:
        return fp.read()
return "Cannot open file!"
```

MORE INFORMATION:
It is preferable to use EAFP techniques as follows.

```
try:
    fp = open("myfile")
except IOError as e:
    if e.errno == errno.EACCES:
        return "some default data"
    # Not a permission error.
    raise
else:
    with fp:
        return fp.read()
```

REFERENCES:
https://docs.python.org/2/library/os.html#os.access