One type of race condition is known as a time-of-check-to-time-of-use, or TOCTTOU (pronounced "TOOK too").

a. What is a TOCTTOU race condition and why is it a security issue?

Step 1:

When a resource is checked for a specific value, such as whether a file exists or not, and that value then changes before the resource is used, invalidating the check's results, this is known as a time-of-check-to-time-of-use (TOCTTOU) race condition. A race condition involving the checking of the status of a system component (such as a security credential) and the usage of the results of that check is what leads to the TOCTOU class of software bugs in software development.

Step 2:

This hypothetical race situation above turns into a "Race Condition Vulnerability" if an attacker exploits it to carry out harmful actions and assist in getting beyond guarded safeguards. When many threads update variable values using the same shared memory, this vulnerability frequently arises.

There are primarily two methods used to carry out this attack:

Interference by a malicious process: The attacker inserts some code in the middle of a secure process's phases.

A trustworthy process interfering - The attacker takes advantage of two separate processes that share certain state

c. Give two real-world examples of TOCTTOU race conditions.

A light switch is an easy illustration of a race condition. In certain homes, a shared ceiling light is controlled by a number of different light switches. The switch position is irrelevant when certain circuit types are employed. Moving either switch from its current position turns the light off if it is on. Similar to this, if either switch is moved from its current position when the light is off, the light will turn on.

In light of this, consider what may occur if two individuals attempted to turn on the light simultaneously using two different switches. One command could override another, or the two operations could trip the circuit breaker.

.