

# Re: window of vulnerability

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-02/msg03234.html>

---

- *From:* mei <[mei@xxxxxx](mailto:mei@xxxxxx)>
  - *Date:* Tue, 27 Feb 2007 23:11:01 +0100
- 

Daniel Pitts a écrit :

On Feb 27, 1:23 pm, mei <[m...@xxxxxx](mailto:m...@xxxxxx)> wrote:

Hello,

I heard several times people talking about a security issue known as window of vulnerability.

I am not sure to be capable of explaining it correctly, but it is more or less about having a concurrent thread accessing a code normally protected, that would trigger within a window of vulnerability eventually opened by the thread running the protection mechanism.

I would like to know if it is only a theoretical problem or if it can happen on real conditions? In my mind, I think that we should be able to enforce the executions in a particular order, and this depends on too many parameters to be controlled.

Mei.

Thats just it, you *\*can\** force order, but in a multi-threaded environment, you have to worry about the order to force.

In a single threaded application, everything "seems" to happen in the order you expect, the implicit order you asked for, the compiler, JVM, and underlying CPU are designed to do this for you. However, they may "reorder" certain instructions that don't interact with each other.

This gives them the ability to optimize pipelines, wait times, etc...

The effect is unobservable to a single threaded application.

However, there is no implicit order defined between two threads. You have to be explicit by using some sort of synchronization. In Java, that means using either synchronization and volatile variables, or using the new in java.util.concurrent classes. More than just using them, you have to use them correctly.

I recently picked up a copy of "Java Concurrency in Practice" <<http://jcip.net/>> which discusses the pitfalls of using multi-threaded applications without understanding the underlying behavior.

Hope this helps,

Re: window of vulnerability

Daniel.

Thank you Daniel for your answer. I know that execution order can be controlled using synchronization. However, my question is more oriented toward a security issue. Would it be possible to "attack" a third-party class (whose code wouldn't be modifiable) by running a concurrent thread that would exploit a window of vulnerability?

More precisely, let's say we have a class for which we can't change its synchronization design, but for which we still make the assumption it exhibits a window of vulnerability. In this case, could someone give me a simple example (or a pointer to) of such a class and how a thread could attack it.

I hope I'm clear enough...

.