why is potential dynamic constant assignment an error when actual dynamic constant assignment is only a warning

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 2.7.0p0 (2019-12-25 revision 647ee6f901) [x86_64-linux]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN</td>
</tr>
</tbody>
</table>

Description

```ruby
module Mod1
  def self.define_consts
    const_set(:CONST1, :CONST1)
    # this is actual const re-assignment but only a warning
    const_set(:CONST1, :CONST1)

    # this is const initialization but becomes an error
    # because it looks like it could be re-assignment
    # if actual const re-assignment is only a warning
    # why is a possible const re-assignment (which might not be one), an error
    self::*CONST2 = :CONST2 unless const_defined?(:CONST2, false)
  end

  define_consts
end
```