Specific combination of regexp and string causes 100% CPU and doesn't recover

Specific combination of regexp and string can cause ruby process to hang with 100% CPU.

Reproducing (in irb):
```
/\A(?:%\h\h|[^%]+)*\z/
```

E.g.: `=~ "199542328.1312293792.1.1.utmcsr%3Dgoogle%7Cutmccn%"`

(above hangs indefinably with 100% cpu)

```
/\A(?:%\h\h|[^%]+)*\z/
```

E.g.: `=~ "199542328.1312293792.1.1.utmcsr%3Dgoogle%7Cutmccn"

(same but without % at the end returns succesfully)

The code in question is found in Rack:Utils (v1.3.2, not used in v1.2.1) and can basically "kill" any server process (happened to us in production on a thin machine after we upgraded to newer rack). The above bug means that it is very easy to perform DoS on affected ruby server.

Related issues:
Has duplicate Ruby master - Bug #5322: URI.decode_www_form_component very slo...

Closed 09/14/2011

History

#1 - 08/03/2011 01:15 AM - regularfry (Alex Young)
- File uri.patch added

I'd disagree with the location of this bug. I've had a quick look, and while this doesn't look like a Ruby bug, perhaps it ought to be. The regex as given:
```
/\A(?=%\h\h|[^%]+)*\z/
```
does not appear in Rack, but does appear in lib/ruby/1.9.1/uri/common.rb (line 778 in -p290). Rack has this:
```
/\A(?=%[0-9a-fA-F]{2}|[^%])\^\z/
```

This would not appear to suffer from the same exponential behaviour as that in URI, while apparently validating the same strings. Perhaps the appropriate substitution should be made in uri/common.rb? Patch untested, but "looks right".

#2 - 08/03/2011 03:21 AM - matmarex (Bartosz Dz)

No, this is a buggy regex - a case of catastrophic backtracking. [http://www.regular-expressions.info/catastrophic.html](http://www.regular-expressions.info/catastrophic.html)

Removing the "\^" after \^\d fixes it.

This is because both this "\^" is greedy and the "\*" at the end are greedy, so Ruby tries to match as many \^\d as possible, and then to match the result as many times as possible; obviously it fails (since the next character is the percent sign), then it backtracks to one less character, and tries to match this; then again, and again. Number of repetitions skyrockets and boom, everything hangs while Ruby tries hard to backtrack and backtrack.

#3 - 08/03/2011 05:26 AM - regularfry (Alex Young)

Bartosz Dz wrote:

No, this is a buggy regex

Yes. The buggy regex is in Ruby's stdlib. My trivial patch (which I now realise I managed to mess up the file paths on, apologies for that) fixes it in the manner you describe.

#4 - 08/03/2011 08:09 AM - naruse (Yui NARUSE)
- Tracker changed from Bug to Backport
- Project changed from Ruby master to Backport192
- Status changed from Open to Assigned
- Assignee set to yugui (Yuki Sonoda)
It is already fixed in trunk and 1.9.3, please backport r32622 and r32631 to 1.9.2.

#5 - 05/30/2016 08:33 AM - naruse (Yui NARUSE)
- Status changed from Assigned to Rejected

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri.patch</td>
<td>416 Bytes</td>
<td>08/03/2011</td>
<td>regularfry (Alex Young)</td>
</tr>
</tbody>
</table>