

Ruby - Feature #6409

public_send is easily bypassed

05/08/2012 09:37 AM - postmodern (Hal Brodigan)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:	2.0.0	
<div>Description</div> <div>=begin</div> <div>((public_send))) can easily be bypassed, by using it to call ((send)). ((public_send))) should explicitly not allow calling ((send)).</div> <div><pre>class Test private def secret "top secret" end end t = Test.new t.public_send(:secret) # => NoMethodError: private method `secret' called for #<Test:0x0000000159b950> t.public_send(:send, :secret) # => "top secret" t.public_send(:send, :exec, "rm -rf ~") =end</pre></div>		

History

#1 - 05/08/2012 12:41 PM - marcandre (Marc-Andre Lafortune)

- Tracker changed from Bug to Feature

This is definitely not a bug, as send is public.

I don't understand the rationale behind your request. You are still using send. public_send does not and cannot guarantee that a private method won't be called at some point; only that it won't send the message in case it's a not a public method.

#2 - 05/08/2012 02:18 PM - postmodern (Hal Brodigan)

((public_send))) should only allow calling public methods. By extension, it should not allow calling ((send))), since that would negate the purpose of ((public_send))). In the context of ((public_send))), the ((send))) method has special meaning.

#3 - 05/08/2012 02:34 PM - jeremyevans0 (Jeremy Evans)

I see no reason to special case this. send is a public method, therefore public_send should be allowed to call it. Attempting to deny access to send for safety reasons is pointless considering that instance_eval is public can be used to work around the issue in the same way:

```
t.public_send(:instance_eval, 'secret')
t.public_send(:instance_eval, 'exec("rm -rf ~")')
```

public_send doesn't imply safety, at all, and it was not designed for such a purpose.

#4 - 05/08/2012 04:39 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

The whole purpose of public_send is to prohibit the invocation of non-public methods, probably to help detecting error earlier. In that sense, as Jeremy expressed, we see no reason to prohibit #send the public method. public_send (and method visibility in general) is not the way to ensure anything, e.g. security.

Matz.

#5 - 05/08/2012 07:02 PM - alexeymuranov (Alexey Muranov)

@postmodern, send is a public method, why would public_send refuse to call it? Were you suggesting to remove the send method?

#6 - 05/08/2012 07:48 PM - MartinBosslet (Martin Bosslet)

Wouldn't something as proposed in <http://bugs.ruby-lang.org/issues/5455> help in the long run?

#7 - 05/08/2012 08:02 PM - postmodern (Hal Brodigan)

Now I know public_send should *not* be trusted with arbitrary method names/arguments. Is there even a safe version of send?

#8 - 05/08/2012 08:29 PM - alexeymuranov (Alexey Muranov)

=begin
Maybe something like:

```
class SafeClass
  METHOD_SAFE = { :safe_method_1 => true, :safe_method_2 => true }
```

```
  def safe_send(method, *arguments)
    send(method, *arguments) if METHOD_SAFE[method]
  end
```

end

But this is not completely safe either, because anybody can reopen this class later and change the (METHOD_SAFE) constant or even the (safe_send) method.

=end