Ruby - Bug #21394

Memory leak in Prism's RubyVM::InstructionSequence.new

06/02/2025 07:09 PM - peterzhu2118 (Peter Zhu)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	3.2: DONTNEED, 3.3: DONTNEED, 3.4: DONE

Description

Fix: https://github.com/ruby/ruby/pull/13496

There are two ways to make RubyVM::InstructionSequence.new raise which would cause the options->scopes to leak memory:

1. Passing in any (non T_FILE) object where the to_str raises.
2. Passing in a T_FILE object where String#initialize_dup raises. This is because rb_io_path dups the string.

Example 1:

```
10.times do
  100_000.times do
    RubyVM::InstructionSequence.new(nil)
  rescue TypeError
  end

  puts `ps -o rss= -p #{$$}`
end
```

Before:

```
13392
17104
20256
23920
27264
30432
33584
36752
40032
43232
```

After:

```
9392
11072
11648
11648
11648
11712
11712
11712
11744
11744
```

Example 2:

```
require "tempfile"

MyError = Class.new(StandardError)
String.prepend(Module.new do
  def initialize_dup(_)
```

```
    if $raise_on_dup
      raise MyError
    else
      super
    end
  end
end)

Tempfile.create do |f|
  10.times do
    100_000.times do
      $raise_on_dup = true
      RubyVM::InstructionSequence.new(f)
    rescue MyError
    else
      raise "MyError was not raised during RubyVM::InstructionSequence.new"
    end

    puts `ps -o rss= -p #{$$}`
  ensure
    $raise_on_dup = false
  end
end
```

Before:

```
14080
18512
22000
25184
28320
31600
34736
37904
41088
44256
```

After:

```
12016
12464
12880
12880
12880
12912
12912
12912
12912
12912
```

Associated revisions

Revision 34b407a4 - 06/03/2025 02:00 PM - peterzhu2118 (Peter Zhu)

Fix memory leak in Prism's RubyVM::InstructionSequence.new

[Bug #21394]

There are two ways to make RubyVM::InstructionSequence.new raise which would cause the options->scopes to leak memory:

1. Passing in any (non T_FILE) object where the to_str raises.
2. Passing in a T_FILE object where String#initialize_dup raises. This is because rb_io_path dups the string.

Example 1:

```
10.times do
  100_000.times do
```

```
    RubyVM::InstructionSequence.new(nil)
  rescue TypeError
  end

  puts `ps -o rss= -p #{$$}`
end
```

Before:

```
13392
17104
20256
23920
27264
30432
33584
36752
40032
43232
```

After:

```
9392
11072
11648
11648
11648
11712
11712
11712
11744
11744
```

Example 2:

```
require "tempfile"

MyError = Class.new(StandardError)
String.prepend(Module.new do
  def initialize_dup(_)
    if $raise_on_dup
      raise MyError
    else
      super
    end
  end
end)

Tempfile.create do |f|
  10.times do
    100_000.times do
      $raise_on_dup = true
      RubyVM::InstructionSequence.new(f)
    rescue MyError
    else
      raise "MyError was not raised during RubyVM::InstructionSequence.new"
    end

    puts `ps -o rss= -p #{$$}`
  ensure
    $raise_on_dup = false
  end
end
```

Before:

```
14080
18512
22000
25184
28320
31600
34736
37904
41088
```

```
44256
```

After:

```
12016
12464
12880
12880
12880
12912
12912
12912
12912
12912
```

Revision c397c2d1 - 07/14/2025 08:55 PM - k0kubun (Takashi Kokubun)

merge revision(s) 34b407a4a89e69dd04f692e2b29efa2816d4664a: [Backport #21394]

```
    Fix memory leak in Prism's RubyVM::InstructionSequence.new

    [Bug #21394]

    There are two ways to make RubyVM::InstructionSequence.new raise which
    would cause the options->scopes to leak memory:

    1. Passing in any (non T_FILE) object where the to_str raises.
    2. Passing in a T_FILE object where String#initialize_dup raises. This is
       because rb_io_path dups the string.

    Example 1:

        10.times do
          100_000.times do
            RubyVM::InstructionSequence.new(nil)
          rescue TypeError
          end

          puts `ps -o rss= -p #{$$}`
        end

    Before:

        13392
        17104
        20256
        23920
        27264
        30432
        33584
        36752
        40032
        43232

    After:

        9392
        11072
        11648
        11648
        11648
        11712
        11712
        11712
        11744
        11744

    Example 2:

        require "tempfile"

        MyError = Class.new(StandardError)
        String.prepend(Module.new do
          def initialize_dup(_)
            if $raise_on_dup
```

```
              raise MyError
            else
              super
            end
          end
        end)

        Tempfile.create do |f|
          10.times do
            100_000.times do
              $raise_on_dup = true
              RubyVM::InstructionSequence.new(f)
            rescue MyError
            else
              raise "MyError was not raised during RubyVM::InstructionSequence.new"
            end

            puts `ps -o rss= -p #{$$}`
          ensure
            $raise_on_dup = false
          end
        end
```

Before:

```
        14080
        18512
        22000
        25184
        28320
        31600
        34736
        37904
        41088
        44256
```

After:

```
        12016
        12464
        12880
        12880
        12880
        12912
        12912
        12912
        12912
        12912
```

History

#1 - 06/03/2025 02:00 PM - peterzhu2118 (Peter Zhu)

- Status changed from Open to Closed

Applied in changeset git|34b407a4a89e69dd04f692e2b29efa2816d4664a.

Fix memory leak in Prism's RubyVM::InstructionSequence.new

[Bug #21394]

There are two ways to make RubyVM::InstructionSequence.new raise which would cause the options->scopes to leak memory:

1. Passing in any (non T_FILE) object where the to_str raises.
2. Passing in a T_FILE object where String#initialize_dup raises. This is because rb_io_path dups the string.

Example 1:

```
10.times do
  100_000.times do
    RubyVM::InstructionSequence.new(nil)
  rescue TypeError
```

```
  end

  puts `ps -o rss= -p #{$$}`
end
```

Before:

```
13392
17104
20256
23920
27264
30432
33584
36752
40032
43232
```

After:

```
9392
11072
11648
11648
11648
11712
11712
11712
11744
11744
```

Example 2:

```
require "tempfile"

MyError = Class.new(StandardError)
String.prepend(Module.new do
  def initialize_dup(_)
    if $raise_on_dup
      raise MyError
    else
      super
    end
  end
end)

Tempfile.create do |f|
  10.times do
    100_000.times do
      $raise_on_dup = true
      RubyVM::InstructionSequence.new(f)
    rescue MyError
    else
      raise "MyError was not raised during RubyVM::InstructionSequence.new"
    end

    puts `ps -o rss= -p #{$$}`
  ensure
    $raise_on_dup = false
  end
end
```

Before:

```
14080
18512
22000
25184
28320
31600
34736
37904
41088
44256
```

After:

```
12016
12464
12880
12880
12880
12912
12912
12912
12912
12912
```

#2 - 07/14/2025 08:55 PM - k0kubun (Takashi Kokubun)

- Backport changed from 3.2: DONTNEED, 3.3: DONTNEED, 3.4: REQUIRED to 3.2: DONTNEED, 3.3: DONTNEED, 3.4: DONE

ruby_3_4 c397c2d177a0b7fd13b69c5109418fbe1f9eccd1 merged revision(s) 34b407a4a89e69dd04f692e2b29efa2816d4664a.