phoenixframework / **phoenix** Public

Code | Issues 17 | Pull requests 30 | Actions | Security and quality 1

Commit 912ea18

SteffenDE and **josevalim** committed 2 days ago Verified

```
prevent unexpected memory usage on nd-json body splitting

Co-authored-by: José Valim <jose.valim@gmail.com>
```

v1.7 · v1.7.23 v1.7.22

1 parent e4815f0 commit 912ea18

3 files changed +32 -23

Top

Filter files...

- assets/js/phoenix
  - constants.js
  - longpoll.js
- lib/phoenix/transports
  - long_poll.ex

Search within code

assets/js/phoenix/constants.js

```
@@ -3,6 +3,7 @@ export const phxWindow = typeof window !== "undefined" ?
window : null
3  3  export const global = globalSelf || phxWindow || global
4  4  export const DEFAULT_VSN = "2.0.0"
5  5  export const SOCKET_STATES = {connecting: 0, open: 1, closing: 2, closed: 3}
   6  + export const MAX_LONGPOLL_BATCH_SIZE = 100;
6  7  export const DEFAULT_TIMEOUT = 10000
```

```
7   8   export const WS_CLOSE_NORMAL = 1000
8   9   export const CHANNEL_STATES = {
```

assets/js/phoenix/longpoll.js

```
    ...   @@ -1,6 +1,7 @@
1   1     import {
2   2       SOCKET_STATES,
3       -   TRANSPORTS
        3   +   TRANSPORTS,
        4   +   MAX_LONGPOLL_BATCH_SIZE
4   5     } from "./constants"
5   6
6   7     import Ajax from "./ajax"
          @@ -132,16 +133,22 @@ export default class LongPoll {
132 133       }
133 134     }
134 135
135     -   batchSend(messages){
        136 +   batchSend(messages, offset = 0){
136 137       this.awaitingBatchAck = true
137     -     this.ajax("POST", "application/x-ndjson", messages.join("\n"), () =>
      this.onerror("timeout"), resp => {
138     -       this.awaitingBatchAck = false
        138 +     const next = offset + MAX_LONGPOLL_BATCH_SIZE
        139 +     const batch = messages.slice(offset, next)
        140 +     this.ajax("POST", {"Content-Type": "application/x-ndjson"},
      batch.join("\n"), () => this.onerror("timeout"), resp => {
139 141         if(!resp || resp.status !== 200){
        142 +         this.awaitingBatchAck = false
140 143           this.onerror(resp && resp.status)
141 144           this.closeAndRetry(1011, "internal server error", false)
        145 +       } else if(next < messages.length){
        146 +         this.batchSend(messages, next)
142 147         } else if(this.batchBuffer.length > 0){
143 148           this.batchSend(this.batchBuffer)
144 149           this.batchBuffer = []
        150 +       } else {
        151 +         this.awaitingBatchAck = false
```

```
145  152          }
146  153        })
147  154      }
```

lib/phoenix/transports/long_poll.ex

```
@@ -2,8 +2,11 @@ defmodule Phoenix.Transports.LongPoll do
2    2    @moduledoc false
3    3    @behaviour Plug
4    4
5         -  # 10MB
          5  +  # The maximum is 10MB but read_body will cap the whole request at ~8MB,
          6  +  # so this acts as a secondary protection mechanism.
6    7    @max_base64_size 10_000_000
          8  +  # TODO: enforce batch size on the server in the next release
          9  +  # @max_poll_batch_size 100
7    10
8    11   import Plug.Conn
9    12   alias Phoenix.Socket.{V1, V2, Transport}
@@ -77,30 +80,28 @@ defmodule Phoenix.Transports.LongPoll do
77   80     defp publish(conn, server_ref, endpoint, opts) do
78   81       case read_body(conn, []) do
79   82         {:ok, body, conn} ->
80        -         # we need to match on both v1 and v2 protocol, as well as wrap for backwards compat
81        -         batch =
          83 +         # We need to match on both v1 and v2 protocol, as well as wrap for backwards compat
          84 +         status =
82   85             case get_req_header(conn, "content-type") do
83   86               ["application/x-ndjson"] ->
84   87                 body
85        -                |> String.split(["\n", "\r\n"])
86        -                |> Enum.map(fn
87        -                  "[" <> _ = txt -> {txt, :text}
88        -                  base64 -> {safe_decode64!(base64), :binary}
          88 +                |> String.splitter(["\n", "\r\n"])
          89 +                # |> Stream.take(@max_poll_batch_size)
          90 +                |> Enum.find(fn part ->
```

```
                  msg =
                    case part do
                      "[" <> _ = txt -> {txt, :text}
                      base64 -> {safe_decode64!(base64), :binary}
                    end

                  transport_dispatch(endpoint, server_ref, msg, opts)
                end)

              _ ->
-               [{body, :text}]
+               transport_dispatch(endpoint, server_ref, {body, :text}, opts)
            end

-         {conn, status} =
-           Enum.reduce_while(batch, {conn, nil}, fn msg, {conn, _status} ->
-             case transport_dispatch(endpoint, server_ref, msg, opts) do
-               :ok -> {:cont, {conn, :ok}}
-               :request_timeout = timeout -> {:halt, {conn, timeout}}
-             end
-           end)
-
-         conn |> put_status(status) |> status_json()
+         conn |> put_status(status || :ok) |> status_json()

        _ ->
          raise Plug.BadRequestError
@@ -120,8 +121,8 @@ defmodule Phoenix.Transports.LongPoll do
      broadcast_from!(endpoint, server_ref, {:dispatch, client_ref(server_ref), body, ref})

      receive do
-       {:ok, ^ref} -> :ok
-       {:error, ^ref} -> :ok
+       {:ok, ^ref} -> nil
+       {:error, ^ref} -> nil
      after
        opts[:window_ms] -> :request_timeout
      end
```

Comments 0

Please sign in to comment.