phoenixframework / **phoenix** Public

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

Commit 1a67c61

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>
```

main · v1.8.7 v1.8.6

1 parent 8ca76a2 commit 1a67c61

4 files changed +127 -23

Top

Filter files…

- assets
  - js/phoenix
    - constants.js
    - longpoll.js
  - test
    - longpoll_test.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 || globalThis
```

```
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

```
      ...   @@ -1,7 +1,8 @@
1   1     import {
2   2       SOCKET_STATES,
3   3       TRANSPORTS,
4       -   AUTH_TOKEN_PREFIX
        4 +   AUTH_TOKEN_PREFIX,
        5 +   MAX_LONGPOLL_BATCH_SIZE
5   6     } from "./constants"
6   7
7   8     import Ajax from "./ajax"
      ...   @@ -149,16 +150,22 @@ export default class LongPoll {
149 150       }
150 151     }
151 152
152     -   batchSend(messages){
        153 +   batchSend(messages, offset = 0){
153 154       this.awaitingBatchAck = true
154     -     this.ajax("POST", {"Content-Type": "application/x-ndjson"},
      messages.join("\n"), () => this.onerror("timeout"), resp => {
155     -       this.awaitingBatchAck = false
        155 +     const next = offset + MAX_LONGPOLL_BATCH_SIZE
        156 +     const batch = messages.slice(offset, next)
        157 +     this.ajax("POST", {"Content-Type": "application/x-ndjson"},
      batch.join("\n"), () => this.onerror("timeout"), resp => {
156 158         if(!resp || resp.status !== 200){
        159 +         this.awaitingBatchAck = false
157 160           this.onerror(resp && resp.status)
158 161           this.closeAndRetry(1011, "internal server error", false)
        162 +       } else if(next < messages.length){
        163 +         this.batchSend(messages, next)
```

```
159  164          } else if(this.batchBuffer.length > 0){
160  165            this.batchSend(this.batchBuffer)
161  166            this.batchBuffer = []
     167  +       } else {
     168  +         this.awaitingBatchAck = false
162  169          }
163  170        })
164  171      }
```

assets/test/longpoll_test.js

```
@@ -158,6 +158,101 @@ describe("LongPoll", () => {
158  158            expect.any(Function)
159  159          )
160  160        })
     161  +
     162  +    it("coalesces rapid send() calls and buffers sends made during an in-flight batch", () => {
     163  +      jest.useFakeTimers()
     164  +      try {
     165  +        const longpoll = new LongPoll("http://localhost/socket/longpoll", undefined)
     166  +        longpoll.timeout = 1000
     167  +        // suppress the initial poll() that the constructor schedules via setTimeout(0)
     168  +        longpoll.poll = jest.fn()
     169  +
     170  +        const calls = []
     171  +        Ajax.request.mockImplementation((method, url, headers, body, timeout, ontimeout, callback) => {
     172  +          calls.push({method, body, callback})
     173  +          return {abort: jest.fn()}
     174  +        })
     175  +
     176  +        // Three sends in the same tick should collapse into one currentBatch
     177  +        longpoll.send("a")
     178  +        longpoll.send("b")
     179  +        longpoll.send("c")
     180  +
     181  +        expect(calls).toHaveLength(0)
```

```
182 +         expect(longpoll.currentBatch).toEqual(["a", "b", "c"])
183 +
184 +         // Flush the setTimeout(0) — currentBatch becomes one POST
185 +         jest.runOnlyPendingTimers()
186 +
187 +         expect(calls).toHaveLength(1)
188 +         expect(calls[0].method).toBe("POST")
189 +         expect(calls[0].body).toBe("a\nb\nc")
190 +         expect(longpoll.currentBatch).toBeNull()
191 +         expect(longpoll.awaitingBatchAck).toBe(true)
192 +
193 +         // Sends during in-flight ack go to batchBuffer, not a new request
194 +         longpoll.send("d")
195 +         longpoll.send("e")
196 +         expect(calls).toHaveLength(1)
197 +         expect(longpoll.batchBuffer).toEqual(["d", "e"])
198 +
199 +         // Ack the first batch — the buffered sends should be flushed as the
      next POST
200 +         calls[0].callback({status: 200})
201 +
202 +         expect(calls).toHaveLength(2)
203 +         expect(calls[1].body).toBe("d\ne")
204 +         expect(longpoll.batchBuffer).toEqual([])
205 +         expect(longpoll.awaitingBatchAck).toBe(true)
206 +
207 +         // Ack the buffered batch — nothing left to send
208 +         calls[1].callback({status: 200})
209 +         expect(calls).toHaveLength(2)
210 +         expect(longpoll.awaitingBatchAck).toBe(false)
211 +       } finally {
212 +         jest.useRealTimers()
213 +       }
214 +     })
215 +
216 +     it("splits 150 rapid send() calls into two requests in order", () => {
217 +       jest.useFakeTimers()
218 +       try {
219 +         const longpoll = new LongPoll("http://localhost/socket/longpoll",
      undefined)
```

```
220 +         longpoll.timeout = 1000
221 +         longpoll.poll = jest.fn()
222 +
223 +         const calls = []
224 +         Ajax.request.mockImplementation((method, url, headers, body, timeout,
      ontimeout, callback) => {
225 +           calls.push({body, callback})
226 +           return {abort: jest.fn()}
227 +         })
228 +
229 +         for(let i = 0; i < 150; i++){ longpoll.send(`m${i}`) }
230 +
231 +         // Flush the setTimeout(0) so batchSend runs on the full 150-entry
      batch
232 +         jest.runOnlyPendingTimers()
233 +
234 +         expect(calls).toHaveLength(1)
235 +         const firstLines = calls[0].body.split("\n")
236 +         expect(firstLines).toHaveLength(100)
237 +         expect(firstLines[0]).toBe("m0")
238 +         expect(firstLines[99]).toBe("m99")
239 +
240 +         // Ack the first chunk — batchSend should recurse with the remaining 50
241 +         calls[0].callback({status: 200})
242 +
243 +         expect(calls).toHaveLength(2)
244 +         const secondLines = calls[1].body.split("\n")
245 +         expect(secondLines).toHaveLength(50)
246 +         expect(secondLines[0]).toBe("m100")
247 +         expect(secondLines[49]).toBe("m149")
248 +
249 +         calls[1].callback({status: 200})
250 +         expect(calls).toHaveLength(2)
251 +         expect(longpoll.awaitingBatchAck).toBe(false)
252 +       } finally {
253 +         jest.useRealTimers()
254 +       }
255 +     })
161 256     })
162 257   })
```

```
163  258
```

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     @connect_info_opts [:check_csrf]
8    11
9    12     import Plug.Conn
     @@ -78,30 +81,28 @@ defmodule Phoenix.Transports.LongPoll do
78   81     defp publish(conn, server_ref, endpoint, opts) do
79   82       case read_body(conn, []) do
80   83         {:ok, body, conn} ->
81        -        # we need to match on both v1 and v2 protocol, as well as wrap for
          backwards compat
82        -        batch =
          84  +        # We need to match on both v1 and v2 protocol, as well as wrap for
          backwards compat
          85  +        status =
83   86             case get_req_header(conn, "content-type") do
84   87               ["application/x-ndjson"] ->
85   88                 body
86        -              |> String.split(["\n", "\r\n"])
87        -              |> Enum.map(fn
88        -                "[" <> _ = txt -> {txt, :text}
89        -                base64 -> {safe_decode64!(base64), :binary}
          89  +              |> String.splitter(["\n", "\r\n"])
          90  +              # |> Stream.take(@max_poll_batch_size)
          91  +              |> Enum.find(fn part ->
          92  +                msg =
          93  +                  case part do
```

```
      94  +                    "[" <> _ = txt -> {txt, :text}
      95  +                    base64 -> {safe_decode64!(base64), :binary}
      96  +                  end
      97  +
      98  +                transport_dispatch(endpoint, server_ref, msg, opts)
90    99                  end)
91   100
92   101              _ ->
93        -              [{body, :text}]
     102  +              transport_dispatch(endpoint, server_ref, {body, :text}, opts)
94   103            end
95   104
96        -        {conn, status} =
97        -          Enum.reduce_while(batch, {conn, nil}, fn msg, {conn, _status} ->
98        -            case transport_dispatch(endpoint, server_ref, msg, opts) do
99        -              :ok -> {:cont, {conn, :ok}}
100       -              :request_timeout = timeout -> {:halt, {conn, timeout}}
101       -            end
102       -          end)
103       -
104       -        conn |> put_status(status) |> status_json()
     105  +        conn |> put_status(status || :ok) |> status_json()
105  106
106  107        _ ->
107  108          raise Plug.BadRequestError
       @@ -121,8 +122,8 @@ defmodule Phoenix.Transports.LongPoll do
121  122      broadcast_from!(endpoint, server_ref, {:dispatch, client_ref(server_ref),
         body, ref})
122  123
123  124      receive do
124       -      {:ok, ^ref} -> :ok
125       -      {:error, ^ref} -> :ok
     125  +      {:ok, ^ref} -> nil
     126  +      {:error, ^ref} -> nil
126  127      after
127  128        opts[:window_ms] -> :request_timeout
128  129      end
```

Comments 0

Please sign in to comment.