

mtrudel / bandit Public

<> Code Issues 1 Pull requests 1 Actions Security and quality 5 Ins

Commit 8156921



mtrudel authored 2 hours ago · 26 / 27 · Verified

Merge commit from fork

* Define a new max_inflate_ratio limit on compressed WebSocket frames

* Move max compression to 25:1

main · 1.11.0

1 parent [fc3cf61](#) commit 8156921

4 files changed

+90 -11

Top



- ✓ lib
 - bandit.ex
 - ✓ bandit/websocket
 - connection.ex
 - permmessage_deflate.ex
- ✓ test/bandit/websocket
 - protocol_test.exs



lib/bandit.ex

@@ -202,13 +202,17 @@ defmodule Bandit do

```
202 202      * `deflate_options`: A keyword list of options to set on the deflate library
      when using the
```

```

203 203     per-message deflate extension. A complete list can be found at
      `t:deflate_options/0`.
204 204     `window_bits` is currently ignored and left to negotiation.
205 + * `max_inflate_ratio`: The maximum allowable ratio to allow decompression of
      received WebSocket
206 +     messages. Intended to prevent 'inflate bomb' attacks where a tiny deflated
      messages inflates to
207 +     a massive one. Defaults to `25` representing a 25:1 allowable inflation
      ratio.

205 208     """
206 209     @type websocket_options :: [
207 210         {:enabled, boolean()}
208 211         | {:max_frame_size, pos_integer()}
209 212         | {:validate_text_frames, boolean()}
210 213         | {:compress, boolean()}
211 214         | {:deflate_options, deflate_options()}
215 +         | {:max_inflate_ratio, pos_integer()}
212 216     ]
213 217
214 218     @typedoc """
      ↓
      ↑
@@ -245,7 +249,7 @@ defmodule Bandit do
245 249     @http_keys ~w(compress response_encodings deflate_options zstd_options
      log_exceptions_with_status_codes log_protocol_errors log_client_closures)a
246 250     @http_1_keys ~w(enabled max_request_line_length max_header_length
      max_header_count max_requests clear_process_dict gc_every_n_keepalive_requests
      log_unknown_messages)a
247 251     @http_2_keys ~w(enabled max_header_block_size max_requests
      max_reset_stream_rate sendfile_chunk_size default_local_settings)a
248 -     @websocket_keys ~w(enabled max_frame_size validate_text_frames compress
      deflate_options primitive_ops_module)a
252 +     @websocket_keys ~w(enabled max_frame_size validate_text_frames compress
      deflate_options max_inflate_ratio primitive_ops_module)a
249 253     @thousand_island_keys ThousandIsland.ServerConfig.__struct__()
250 254         |> Map.from_struct()
251 255         |> Map.keys()
      ↓

```

lib/bandit/websocket/connection.ex

...

↑

@@ -308,6 +308,9 @@ defmodule Bandit.WebSocket.Connection do

```

308 308         {:error, :no_compress} ->
309 309         do_error(1002, "Received unexpected compressed frame (RFC6455§5.2)",
310 310         socket, connection)
311 311 +         {:error, :too_much_inflation} ->
312 312 +         do_error(1009, "Received compressed frame inflating too much", socket,
313 313 +         connection)
311 314         {:error, _reason} ->
312 315         do_error(1007, "Inflation error", socket, connection)
313 316         end

```

lib/bandit/websocket/permessage_deflate.ex

```

@@ -9,17 +9,19 @@ defmodule Bandit.WebSocket.PerMessageDeflate do
 9 9         server_max_window_bits: 8..15,
10 10         client_max_window_bits: 8..15,
11 11         inflate_context: :zlib.zstream(),
12 -         deflate_context: :zlib.zstream()
12 +         deflate_context: :zlib.zstream(),
13 +         max_inflate_ratio: integer()
13 14     }
14 15
15 16     defstruct server_no_context_takeover: false,
16 17         client_no_context_takeover: false,
17 18         server_max_window_bits: 15,
18 19         client_max_window_bits: 15,
19 20         inflate_context: nil,
20 -         deflate_context: nil
21 +         deflate_context: nil,
22 +         max_inflate_ratio: nil
21 23
22 - @valid_params ~w[server_no_context_takeover client_no_context_takeover
23         server_max_window_bits client_max_window_bits]
24 + @valid_params ~w[server_no_context_takeover client_no_context_takeover
25         server_max_window_bits client_max_window_bits max_inflate_ratio]
23 25
24 26     def negotiate(requested_extensions, opts) do
25 27         :proplists.get_all_values("permessage-deflate", requested_extensions)

```

```

@@ -86,6 +88,7 @@ defmodule Bandit.WebSocket.PerMessageDeflate do
 86 88     instance = struct(__MODULE__, params)
 87 89     inflate_context = :zlib.open()
 88 90     :ok = :zlib.inflateInit(inflate_context, fix_bits(-
        instance.client_max_window_bits))
 91 +
 89 92     deflate_context = :zlib.open()
 90 93     deflate_opts = Keyword.get(opts, :deflate_options, [])
 91 94
@@ -99,7 +102,14 @@ defmodule Bandit.WebSocket.PerMessageDeflate do
 99 102         Keyword.get(deflate_opts, :strategy, :default)
100 103     )
101 104
102 -     %{instance | inflate_context: inflate_context, deflate_context:
        deflate_context}
105 +     max_inflate_ratio = Keyword.get(opts, :max_inflate_ratio, 25)
106 +
107 +     %{
108 +     instance
109 +     | inflate_context: inflate_context,
110 +     deflate_context: deflate_context,
111 +     max_inflate_ratio: max_inflate_ratio
112 +     }
103 113     end
104 114
105 115     # https://www.erlang.org/doc/man/zlib.html#deflateInit-6
@@ -109,19 +119,45 @@ defmodule Bandit.WebSocket.PerMessageDeflate do
109 119     # Note that we pass back the context to the caller even though it is
        unmodified locally
110 120
111 121     def inflate(data, %__MODULE__{} = context) do
112 -     inflated_data =
113 -     context.inflate_context
114 -     |> :zlib.inflate(<<data::binary, 0x00, 0x00, 0xFF, 0xFF>>)
115 -     |> IO.iodata_to_binary()
122 +     safe_inflate(
123 +     context.inflate_context,
124 +     :zlib.safeInflate(context.inflate_context, <<data::binary, 0x00, 0x00,
        0xFF, 0xFF>>),
125 +     [],

```

```

126 +     byte_size(data) * context.max_inflate_ratio
127 + )
128 + |> case do
129 +     {:ok, inflated_iodata, inflate_context} ->
130 +         if context.client_no_context_takeover, do:
131 +             :zlib.inflateReset(context.inflate_context)
132 +             {:ok, IO.iodata_to_binary(inflated_iodata), %{context |
133 +                 inflate_context: inflate_context}}
134 +         else
135 +             {:error, reason} ->
136 +             {:error, reason}
137 +         end
138 +     rescue
139 +         e -> {:error, "Error encountered #{inspect(e)}"}
140 +     end
141 +
142 +     def inflate(_data, nil), do: {:error, :no_compress}
143 +
144 +     defp safe_inflate(inflate_context, {:continue, deflated}, buffer,
145 +         bytes_remaining)
146 +         when bytes_remaining > 0 do
147 +         safe_inflate(
148 +             inflate_context,
149 +             :zlib.safeInflate(inflate_context, <<>>),
150 +             [buffer | deflated],
151 +             bytes_remaining - IO.iodata_length(deflated)
152 +         )
153 +     end
154 +
155 +     defp safe_inflate(_inflate_context, {:continue, _deflated}, _buffer,
156 +         bytes_remaining)
157 +         when bytes_remaining <= 0 do
158 +         {:error, :too_much_inflation}
159 +     end
160 +
161 +     defp safe_inflate(inflate_context, {:finished, deflated}, buffer,
162 +         _bytes_remaining) do

```

```

158 +     {:ok, [buffer | deflated], inflate_context}
159 +   end
160 +
125 161   def deflate(data, %__MODULE__{} = context) do
126 162     deflated_data =
127 163       context.deflate_context

```



test/bandit/websocket/protocol_test.exs



```

..... @@ -239,6 +239,42 @@ defmodule WebSocketProtocolTest do
239 239     assert SimpleWebSocketClient.recv_pong_frame(client) == {:ok, "OK"}
240 240     assert SimpleWebSocketClient.recv_ping_frame(client) == {:ok, "OK"}
241 241   end
242 +
243 +   test "server sends a 1009 on an overly compressed frame", context do
244 +     output =
245 +       capture_log(fn ->
246 +         zstream = :zlib.open()
247 +         :ok = :zlib.deflateInit(zstream, :default, :deflated, -15, 8,
248 +           :default)
249 +         deflated_chunks =
250 +           Enum.map(
251 +             1..1_000_000,
252 +             fn _ -> :zlib.deflate(zstream, "aaaaaaaaa", :none) end
253 +           )
254 +         final_flush = :zlib.deflate(zstream, <<>>, :sync)
255 +         :zlib.close(zstream)
256 +         deflated = IO.iodata_to_binary([deflated_chunks, final_flush])
257 +         trailer_size = byte_size(deflated) - 4
258 +         <<payload::binary-size(trailer_size), 0x00, 0x00, 0xFF, 0xFF>> =
259 +           deflated
260 +
261 +         client = SimpleWebSocketClient.tcp_client(context)
262 +         SimpleWebSocketClient.http1_handshake(client, TerminateWebSock, [],
263 +           true)
263 +         SimpleWebSocketClient.send_text_frame(client, payload, 0xC)
264 +

```

```
265 +           # Get the error that terminate saw, to ensure we're closing for the
      +           expected reason
266 +           assert_receive {:error, "Received compressed frame inflating too
      +           much"}, 500
267 +
268 +           # Validate that the server has started the shutdown handshake from
      +           RFC6455§7.1.2
269 +           assert SimpleWebSocketClient.recv_connection_close_frame(client) ==
      +           {:ok, <<1009::16>>}
270 +
271 +           # Verify that the server didn't send any extraneous frames
272 +           assert SimpleWebSocketClient.connection_closed_for_reading?(client)
273 +           Process.sleep(500)
274 +       end)
275 +
276 +       assert output =~ "Received compressed frame inflating too much"
277 +   end
```

```
242 278     end
```

```
243 279
```

```
244 280     describe "ping frames" do
```



Comments 0



Please [sign in](#) to comment.