mtrudel / **bandit** Public

Code | Issues 1 | Pull requests 1 | Actions | Security and quality 5 | Ins

Commit 21612c7

mtrudel authored 6 hours ago · ✕ 23 / 27 · Verified

```
Merge commit from fork

* Define a new max_fragmented_message_size limit on fragmented WebSocket frames

* Set a default max frame size of 8MB

* Disallow zero length non-fin continuation frames
```

main · 1.11.0

1 parent 8156921 commit 21612c7

4 files changed +132 -8

Top

Filter files…

- lib
  - bandit.ex
  - bandit
    - extractor.ex
    - websocket
      - connection.ex
- test/bandit/websocket
  - protocol_test.exs

Search within code

lib/bandit.ex

```
@@ -191,7 +191,10 @@ defmodule Bandit do
191 191
192 192       * `enabled`: Whether or not to serve WebSocket upgrade requests. Defaults to true
193 193       * `max_frame_size`: The maximum size of a single WebSocket frame (expressed as
194     -       a number of bytes on the wire). Defaults to 0 (no limit)
    194 +       a number of bytes on the wire). Use a value of 0 for no limit. Defaults to 8_000_000
    195 +     * `max_fragmented_message_size`: The maximum size of a WebSocket message delivered across
    196 +       multiple continuation frames (expressed as a number of bytes on the wire). Does NOT affect the
    197 +       handling of single-frame messages. Use a value of 0 for no limit. Defaults to 8_000_000
195 198       * `validate_text_frames`: Whether or not to validate text frames as being UTF-8. Strictly
196 199         speaking this is required per RFC6455§5.6, however it can be an expensive operation and one
197 200         that may be safely skipped in some situations. Defaults to true
@@ -209,6 +212,7 @@ defmodule Bandit do
209 212     @type websocket_options :: [
210 213             {:enabled, boolean()}
211 214             | {:max_frame_size, pos_integer()}
    215 +           | {:max_fragmented_message_size, pos_integer()}
212 216             | {:validate_text_frames, boolean()}
213 217             | {:compress, boolean()}
214 218             | {:deflate_options, deflate_options()}
@@ -249,7 +253,7 @@ defmodule Bandit do
249 253     @http_keys ~w(compress response_encodings deflate_options zstd_options log_exceptions_with_status_codes log_protocol_errors log_client_closures)a
250 254     @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
251 255     @http_2_keys ~w(enabled max_header_block_size max_requests max_reset_stream_rate sendfile_chunk_size default_local_settings)a
252     -   @websocket_keys ~w(enabled max_frame_size validate_text_frames compress deflate_options max_inflate_ratio primitive_ops_module)a
```

```
      256 +   @websocket_keys ~w(enabled max_frame_size max_fragmented_message_size
          validate_text_frames compress deflate_options max_inflate_ratio
          primitive_ops_module)a
253   257     @thousand_island_keys ThousandIsland.ServerConfig.__struct__()
254   258                           |> Map.from_struct()
255   259                           |> Map.keys()
```

lib/bandit/extractor.ex

```
          @@ -33,7 +33,7 @@ defmodule Bandit.Extractor do
33    33
34    34      @spec new(module(), module(), Keyword.t()) :: t()
35    35      def new(frame_parser, primitive_ops_module, opts) do
36        -     max_frame_size = Keyword.get(opts, :max_frame_size, 0)
      36    +     max_frame_size = Keyword.get(opts, :max_frame_size, 8_000_000)
37    37
38    38        %__MODULE__{
39    39          max_frame_size: max_frame_size,
```

lib/bandit/websocket/connection.ex

```
          @@ -94,13 +94,26 @@ defmodule Bandit.WebSocket.Connection do
94    94        case frame do
95    95          %Frame.Continuation{fin: true} = frame ->
96    96            data = IO.iodata_to_binary([connection.fragment_frame.data |
          frame.data])
97        -         frame = %{connection.fragment_frame | fin: true, data: data}
98        -         handle_frame(frame, socket, %{connection | fragment_frame: nil})
      97    +
      98    +         if oversize_message?(data, connection.opts) do
      99    +           do_error(1009, "Received oversize fragmented message", socket,
          connection)
      100   +         else
      101   +           frame = %{connection.fragment_frame | fin: true, data: data}
      102   +           handle_frame(frame, socket, %{connection | fragment_frame: nil})
      103   +         end
99    104
100   105       %Frame.Continuation{fin: false} = frame ->
101       -         data = [connection.fragment_frame.data | frame.data]
```

```
102        -          frame = %{connection.fragment_frame | fin: true, data: data}
103        -          {:continue, %{connection | fragment_frame: frame}}
       106 +          if IO.iodata_length(frame.data) == 0 do
       107 +            do_error(1008, "Received zero byte non-fin continuation frame", socket, connection)
       108 +          else
       109 +            data = [connection.fragment_frame.data | frame.data]
       110 +
       111 +            if oversize_message?(data, connection.opts) do
       112 +              do_error(1009, "Received oversize fragmented message", socket, connection)
       113 +            else
       114 +              {:continue, %{connection | fragment_frame: %{connection.fragment_frame | data: data}}}
       115 +            end
       116 +          end
104    117
105    118           %Frame.Text{} ->
106    119             do_error(1002, "Received unexpected text frame (RFC6455§5.4)", socket, connection)
           @@ -113,6 +126,13 @@ defmodule Bandit.WebSocket.Connection do
113    126         end
114    127       end
115    128
       129 +   defp oversize_message?(data, opts) do
       130 +     case Keyword.get(opts, :max_fragmented_message_size, 8_000_000) do
       131 +       0 -> false
       132 +       max_fragmented_message_size -> IO.iodata_length(data) > max_fragmented_message_size
       133 +     end
       134 +   end
       135 +
116    136       defp handle_control_frame(frame, socket, connection) do
117    137         case frame do
118    138           %Frame.ConnectionClose{} = frame ->
```

test/bandit/websocket/protocol_test.exs

```
           @@ -150,6 +150,106 @@ defmodule WebSocketProtocolTest do
150    150           expected_payload = String.duplicate(payload, 3)
```

```
151  151          assert SimpleWebSocketClient.recv_binary_frame(client) == {:ok,
     expected_payload}
152  152        end
     153  +
     154  +     test "zero byte fin continuation frames are accepted", context do
     155  +       client = SimpleWebSocketClient.tcp_client(context)
     156  +       SimpleWebSocketClient.http1_handshake(client, EchoWebSock)
     157  +
     158  +       payload = String.duplicate("0123456789", 1_000)
     159  +       SimpleWebSocketClient.send_binary_frame(client, payload, 0x0)
     160  +       SimpleWebSocketClient.send_continuation_frame(client, payload, 0x0)
     161  +       SimpleWebSocketClient.send_continuation_frame(client, <<>>)
     162  +
     163  +       expected_payload = String.duplicate(payload, 2)
     164  +       assert SimpleWebSocketClient.recv_binary_frame(client) == {:ok,
          expected_payload}
     165  +     end
     166  +
     167  +     test "zero byte non-fin continuation frames are rejected", context do
     168  +       output =
     169  +         capture_log(fn ->
     170  +           client = SimpleWebSocketClient.tcp_client(context)
     171  +           SimpleWebSocketClient.http1_handshake(client, TerminateWebSock)
     172  +
     173  +           SimpleWebSocketClient.send_binary_frame(client, "0123456789", 0x0)
     174  +           SimpleWebSocketClient.send_continuation_frame(client, <<>>, 0x0)
     175  +
     176  +           # Get the error that terminate saw, to ensure we're closing for the
          expected reason
     177  +           assert_receive {:error, "Received zero byte non-fin continuation
          frame"}, 500
     178  +
     179  +           assert SimpleWebSocketClient.recv_connection_close_frame(client) ==
          {:ok, <<1008::16>>}
     180  +
     181  +           # Verify that the server didn't send any extraneous frames
     182  +           assert SimpleWebSocketClient.connection_closed_for_reading?(client)
     183  +           Process.sleep(500)
     184  +         end)
     185  +
```

```
186 +       assert output =~ "Received zero byte non-fin continuation frame"
187 +     end
188 +
189 +     test "max_fragmented_message_size enforced for continuation frames", context do
190 +       output =
191 +         capture_log(fn ->
192 +           context =
193 +             http_server(context, websocket_options: [max_fragmented_message_size: 2_000_000])
194 +
195 +           client = SimpleWebSocketClient.tcp_client(context)
196 +           SimpleWebSocketClient.http1_handshake(client, TerminateWebSock)
197 +
198 +           payload = String.duplicate("0123456789", 99_999)
199 +           SimpleWebSocketClient.send_binary_frame(client, payload, 0x0)
200 +           SimpleWebSocketClient.send_continuation_frame(client, payload, 0x0)
201 +
202 +           # We should still be alive here
203 +           refute_receive {:error, "Received oversize fragmented message"}, 50
204 +
205 +           # This should send us over the edge
206 +           SimpleWebSocketClient.send_continuation_frame(client, payload, 0x0)
207 +
208 +           # Get the error that terminate saw, to ensure we're closing for the expected reason
209 +           assert_receive {:error, "Received oversize fragmented message"}, 500
210 +
211 +           assert SimpleWebSocketClient.recv_connection_close_frame(client) == {:ok, <<1009::16>>}
212 +
213 +           # Verify that the server didn't send any extraneous frames
214 +           assert SimpleWebSocketClient.connection_closed_for_reading?(client)
215 +           Process.sleep(500)
216 +         end)
217 +
218 +       assert output =~ "Received oversize fragmented message"
219 +     end
220 +
```

```
+     test "max_fragmented_message_size enforced for continuation fin frames", context do
+       output =
+         capture_log(fn ->
+           context =
+             http_server(context, websocket_options: [max_fragmented_message_size: 2_000_000])
+
+           client = SimpleWebSocketClient.tcp_client(context)
+           SimpleWebSocketClient.http1_handshake(client, TerminateWebSock)
+
+           payload = String.duplicate("0123456789", 99_999)
+           SimpleWebSocketClient.send_binary_frame(client, payload, 0x0)
+
+           SimpleWebSocketClient.send_continuation_frame(client, payload, 0x0)
+
+           # We should still be alive here
+           refute_receive {:error, "Received oversize fragmented message"}, 50
+
+           # This should send us over the edge
+           SimpleWebSocketClient.send_continuation_frame(client, payload)
+
+           # Get the error that terminate saw, to ensure we're closing for the expected reason
+           assert_receive {:error, "Received oversize fragmented message"}, 500
+
+           assert SimpleWebSocketClient.recv_connection_close_frame(client) == {:ok, <<1009::16>>}
+
+           # Verify that the server didn't send any extraneous frames
+           assert SimpleWebSocketClient.connection_closed_for_reading?(client)
+           Process.sleep(500)
+         end)
+
+       assert output =~ "Received oversize fragmented message"
+     end
   end

   describe "compressed frames" do
```

Comments 0

Please sign in to comment.