```
From a7d5dfd1862aafa43e5eaca0fdb6acf4cc15b2d0 Mon Sep 17 00:00:00 2001
From: Tokuhiro Matsuno <tokuhirom@gmail.com>
Date: Tue, 19 Nov 2019 19:19:17 +0900
Subject: [PATCH] Prevent HTTP Smuggling.

- Deny request contains both `transfer-encoding` and `content-length`
  headers.
- Deny request contains non-digits in `content-length` header.
---
 lib/Starlet/Server.pm                          | 15 +++++
 ...ing-content-length-and-transfer-encoding.t | 59 ++++++++++++++++++
 ...smuggling-multiple-content-length-header.t | 57 ++++++++++++++++++
 3 files changed, 131 insertions(+)
 create mode 100644 t/15smuggling-content-length-and-transfer-encoding.t
 create mode 100644 t/16smuggling-multiple-content-length-header.t

diff --git a/lib/Starlet/Server.pm b/lib/Starlet/Server.pm
index 0513a42..d68d02a 100644
--- a/lib/Starlet/Server.pm
+++ b/lib/Starlet/Server.pm
@@ -296,6 +296,17 @@ sub handle_connection {
             $buf = substr $buf, $reqlen;
             my $chunked = do { no warnings; lc delete $env->{HTTP_TRANSFER_ENCODING} eq
'chunked' };

+            # If a message is received with both a Transfer-Encoding and a
+            # Content-Length header field, the Transfer-Encoding overrides the
+            # Content-Length.  Such a message might indicate an attempt to
+            # perform request smuggling (Section 9.5) or response splitting
+            # (Section 9.4) and ought to be handled as an error.  A sender MUST
+            # remove the received Content-Length field prior to forwarding such
+            # a message downstream.
+            if ($chunked && $env->{CONTENT_LENGTH}) {
+                last; # Return bad response.
+            }
+
             if ( $env->{HTTP_EXPECT} ) {
                 if ( lc $env->{HTTP_EXPECT} eq '100-continue' ) {
                     $self->write_all($conn, "HTTP/1.1 100 Continue\015\012\015\012")
@@ -307,6 +318,10 @@ sub handle_connection {
             }

             if (my $cl = $env->{CONTENT_LENGTH}) {
+                if ($cl !~ /^[0-9]+$/) { # content-length header must be digits.
+                    last; # Return bad response
+                }
+
                 my $buffer = Plack::TempBuffer->new($cl);
                 while ($cl > 0) {
                     my $chunk;
diff --git a/t/15smuggling-content-length-and-transfer-encoding.t b/t/15smuggling-content-
length-and-transfer-encoding.t
new file mode 100644
index 0000000..c6a0645
--- /dev/null
+++ b/t/15smuggling-content-length-and-transfer-encoding.t
@@ -0,0 +1,59 @@
+use strict;
+use Test::TCP;
+use Plack::Test;
+use HTTP::Request;
+use HTTP::Message::PSGI;
+use Test::More;
+use Digest::MD5;
```

```
+use Plack::Test::Server;
+use Test::TCP;
+use IO::Socket::INET;
+
+$ENV{PLACK_SERVER} = 'Starlet';
+
+my $app = sub {
+    my $env = shift;
+    my $body;
+    my $clen = $env->{CONTENT_LENGTH};
+    while ($clen > 0) {
+        $env->{'psgi.input'}->read(my $buf, $clen) or last;
+        $clen -= length $buf;
+        $body .= $buf;
+    }
+    return [ 200, [ 'Content-Type', 'text/plain', 'X-Content-Length', $env->
{CONTENT_LENGTH} ], [ $body ] ];
+};
+
+my $server = Test::TCP->new(
+    code => sub {
+        my $sock_or_port = shift;
+        my $server = Plack::Loader->auto(
+            port => $sock_or_port,
+            host => '127.0.0.1'
+        );
+        $server->run($app);
+        exit;
+    },
+);
+
+my $sock = IO::Socket::INET->new(
+    PeerAddr => '127.0.0.1',
+    PeerPort => $server->port,
+    Proto => 'tcp',
+);
+
+print {$sock} (
+    "GET / HTTP/1.1\015\012"
+    . "content-length: 3\015\012"
+    . "Transfer-Encoding: chunked\015\012"
+    . "connection: close\015\012"
+    . "\015\012"
+    . "8\015\012"
+    . "SMUGGLED\015\012"
+    . "0\015\012"
+);
+
+my $res_str = do { local $/; <$sock> };
+my ($status_line, ) = split /\015\012/, $res_str;
+is $status_line, 'HTTP/1.1 400 Bad Request';
+
+done_testing;
diff --git a/t/16smuggling-multiple-content-length-header.t b/t/16smuggling-multiple-
content-length-header.t
new file mode 100644
index 0000000..55e69db
--- /dev/null
+++ b/t/16smuggling-multiple-content-length-header.t
@@ -0,0 +1,57 @@
+use strict;
+use Test::TCP;
+use Plack::Test;
+use HTTP::Request;
+use HTTP::Message::PSGI;
```

```
+use Test::More;
+use Digest::MD5;
+use Plack::Test::Server;
+use Test::TCP;
+use IO::Socket::INET;
+
+$ENV{PLACK_SERVER} = 'Starlet';
+
+my $app = sub {
+    my $env = shift;
+    my $body;
+    my $clen = $env->{CONTENT_LENGTH};
+    while ($clen > 0) {
+        $env->{'psgi.input'}->read(my $buf, $clen) or last;
+        $clen -= length $buf;
+        $body .= $buf;
+    }
+    return [ 200, [ 'Content-Type', 'text/plain', 'X-Content-Length', $env->
{CONTENT_LENGTH} ], [ $body ] ];
+};
+
+my $server = Test::TCP->new(
+    code => sub {
+        my $sock_or_port = shift;
+        my $server = Plack::Loader->auto(
+            port => $sock_or_port,
+            host => '127.0.0.1'
+        );
+        $server->run($app);
+        exit;
+    },
+);
+
+my $sock = IO::Socket::INET->new(
+    PeerAddr => '127.0.0.1',
+    PeerPort => $server->port,
+    Proto => 'tcp',
+);
+
+print {$sock} (
+    "GET / HTTP/1.1\015\012"
+    . "content-length: 3\015\012"
+    . "content-length: 9\015\012"
+    . "connection: close\015\012"
+    . "\015\012"
+    . "123456789"
+);
+
+my $res_str = do { local $/; <$sock> };
+my ($status_line, ) = split /\015\012/, $res_str;
+is $status_line, 'HTTP/1.1 400 Bad Request';
+
+done_testing;
```