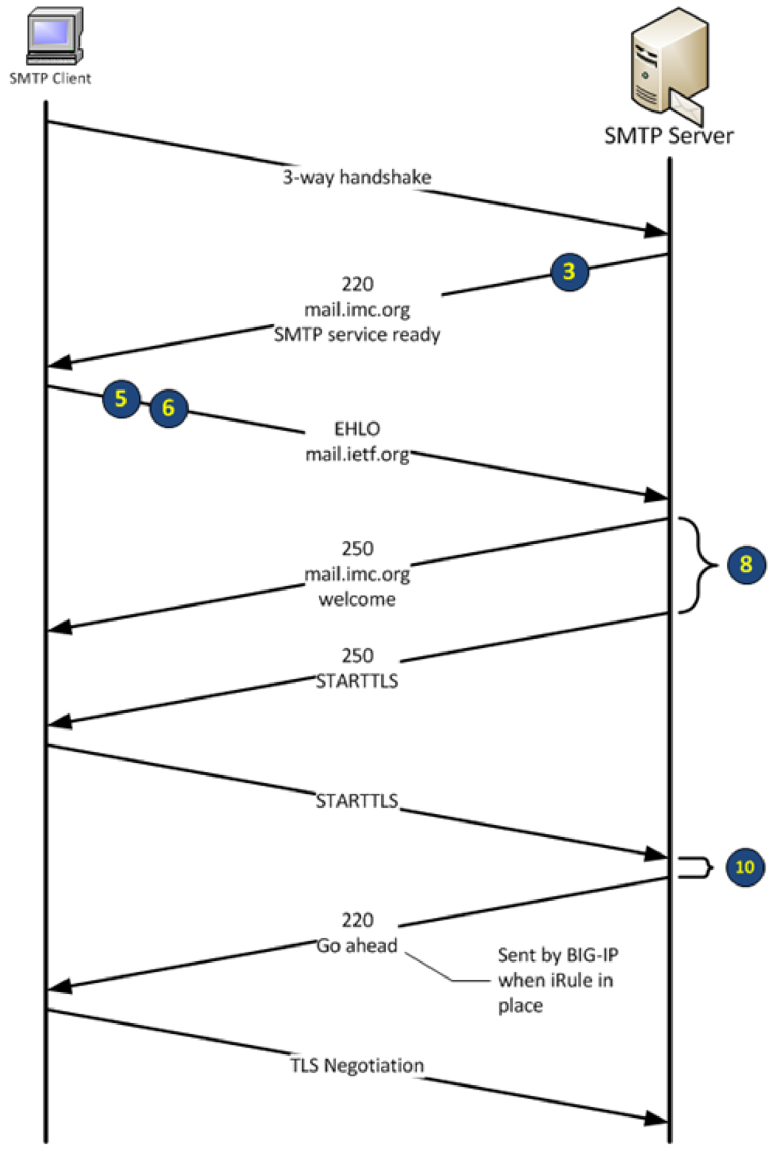
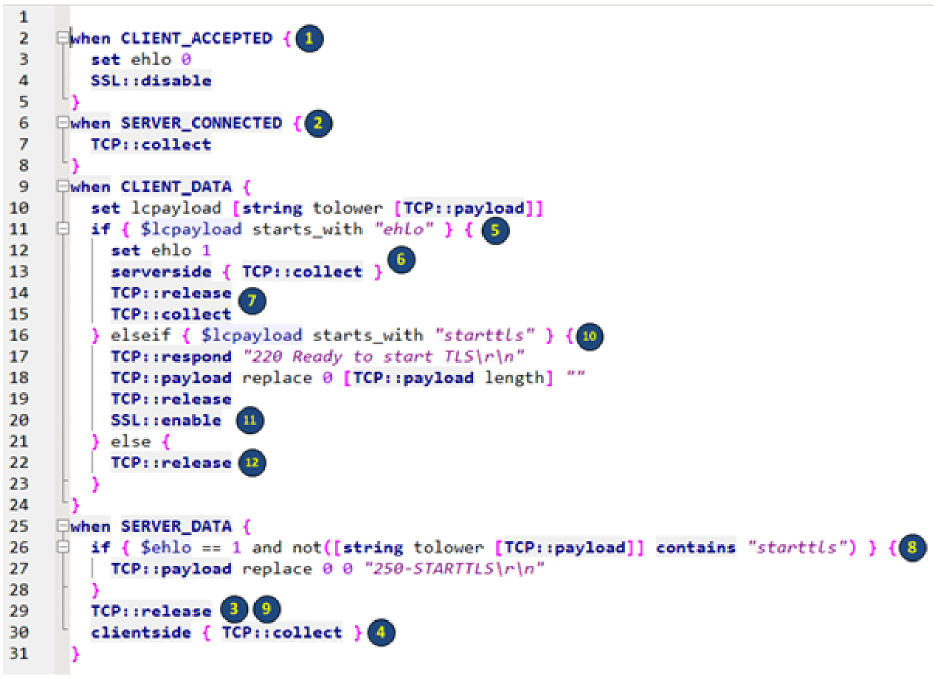
Advanced iRules: SMTP Start TLS

Jason Rahm, 2011-26-10

F5er and DevCentral member natty76 wrote a few iRules a while back on interactive TLS session starting on the SMTP, IMAP, and POP3 protocols. A lot of the iRules can be understood from a flow perspective by reading the iRule top to bottom. This is not the case for these iRules. In this article, I'll break down the SMTP communication context for the BIG-IP as middleman between client and server. I've saved the iRule as an image below so I reference line numbers as I go. The SMTP iRule as well as the IMAP and POP3 iRules are available in the iRules Codeshare.   Before digging into the iRule, the usage example in section six of RFC 2487 is illustrated in the drawing below with the steps from our description to follow highlighted on each leg of the protocol exchange.



The iRule



```
1
2   when CLIENT_ACCEPTED {
3     set ehlo 0
4     SSL::disable
5   }
6   when SERVER_CONNECTED {
7     TCP::collect
8   }
9   when CLIENT_DATA {
10    set lcpayload [string tolower [TCP::payload]]
11    if { $lcpayload starts_with "ehlo" } {
12      set ehlo 1
13      serverside { TCP::collect }
14      TCP::release
15      TCP::collect
16    } elseif { $lcpayload starts_with "starttls" } {
17      TCP::respond "220 Ready to start TLS\r\n"
18      TCP::payload replace 0 [TCP::payload length] ""
19      TCP::release
20      SSL::enable
21    } else {
22      TCP::release
23    }
24  }
25  when SERVER_DATA {
26    if { $ehlo == 1 and not([string tolower [TCP::payload]] contains "starttls") } {
27      TCP::payload replace 0 0 "250-STARTTLS\r\n"
28    }
29    TCP::release
30    clientside { TCP::collect }
31  }
```

1. The process starts with the standard TCP 3-way handshake, which results in the CLIENT_ACCEPTED event firing (line 2). At this point I don't know if the client is requiring TLS yet, so ehlo is set to 0 and SSL is disabled (set by default in the virtual server profile.)
2. With the SMTP protocol, the client initiates the connection but it's the server that sends data first. This means that after the client connection occurs, we need to collect data on the server side of the connection, which is performed here when the SERVER_CONNECTED event fires (lines 6-7).
3. SERVER_DATA fires when the server sends (and has been collected by the collect in line 7). The **if** will not match yet as the ehlo variable is still zero with no client data to match. The data is also released here (line 29)
4. I do however, want to catch when the client sends data, so I do a clientside collect. (line 30)
5. When the client does send data, the CLIENT_DATA event fires (line 9). The payload is de-cased and stored in the variable lcpayload (line 10) and then is checked for the existence of the ehlo command (line 11).
6. If the ehlo was present, I collect on the serverside again (looking for TLS support messages) and make sure I set ehlo to true (lines 12-13).
7. I release client data to continue flow, then collect again to look for the starttls command from the client.
8. Now on the server side, if ehlo is set and the starttls is not in the message, replace the payload with a starttls message. (line 27)
9. And again release the data (line 29)
10. Now when client data arrives with starttls, LTM responds directly informing the client to start TLS communication (lines 16-17) and then swallows the payload (line 18).
11. After responding to the client, I need to enable SSL to the virtual is ready for the client hello from the client (line 20).
12. Finally, if there is no ehlo or starttls from the client, I'll just release the payload. This is to allow clients not supporting starttls through. (line 22)

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com