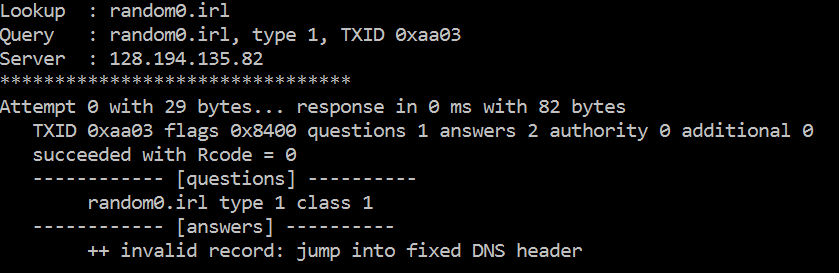
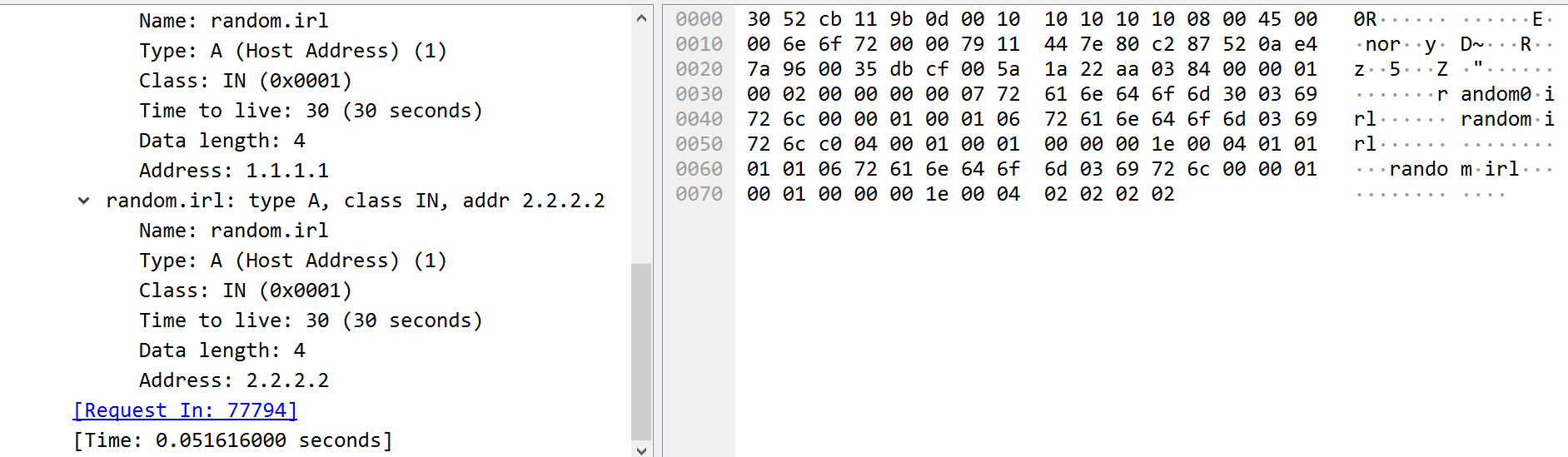
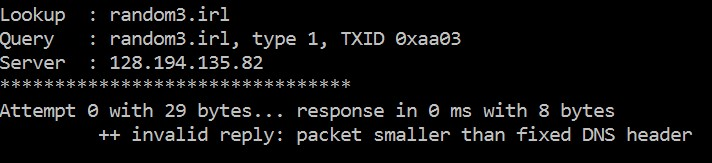
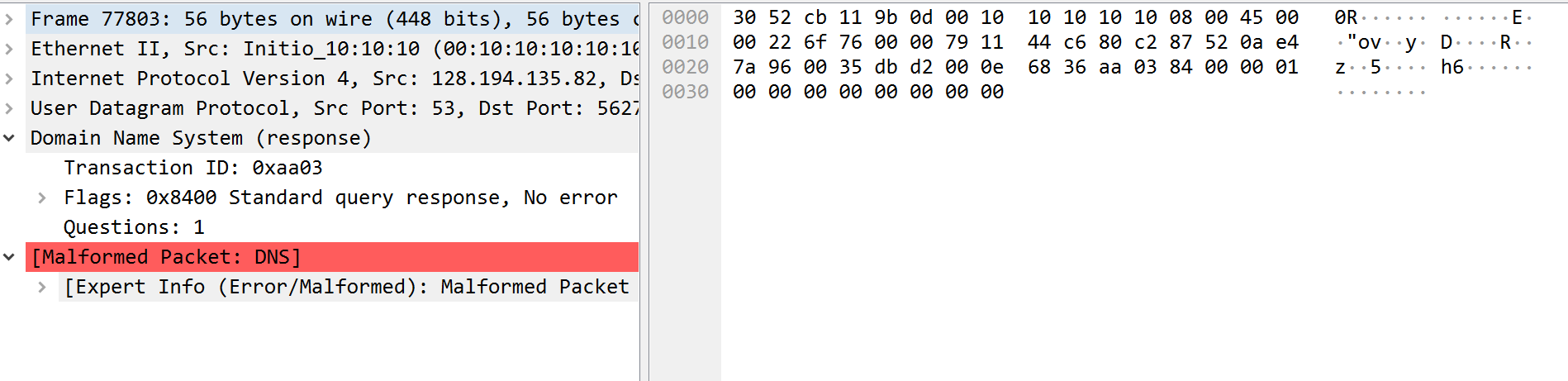
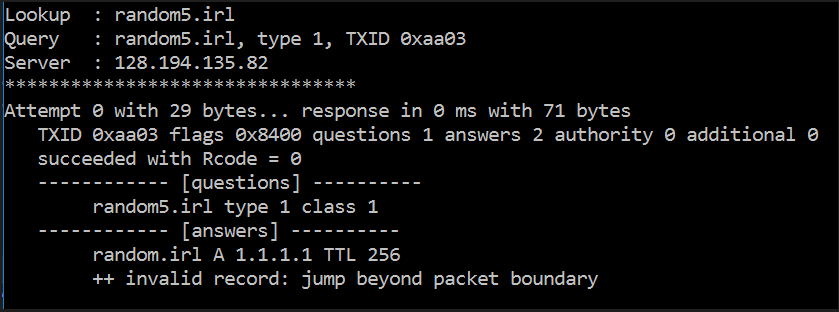
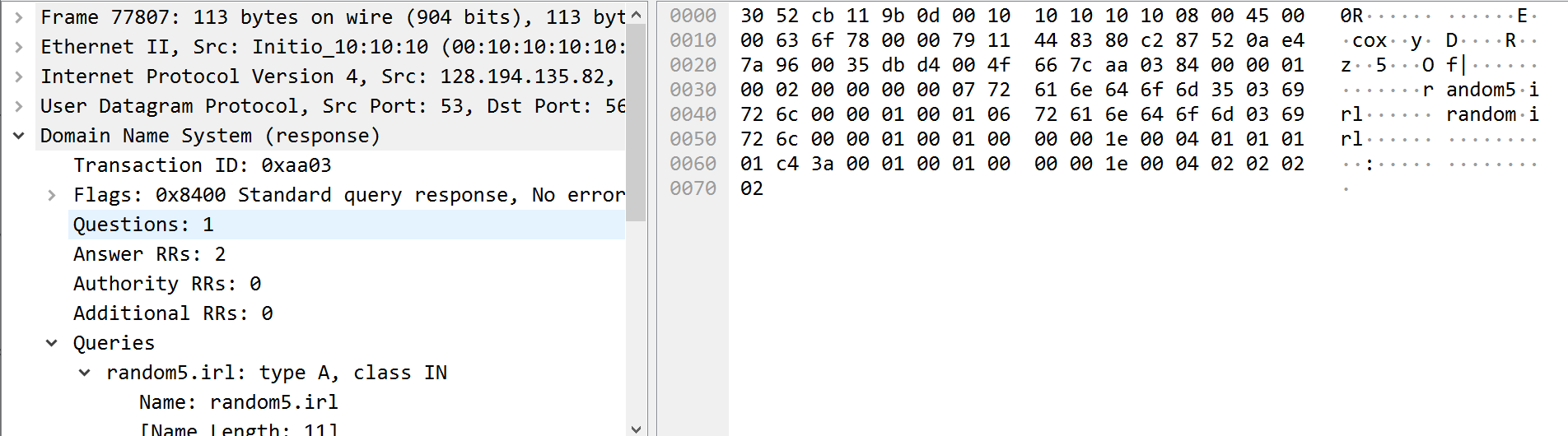
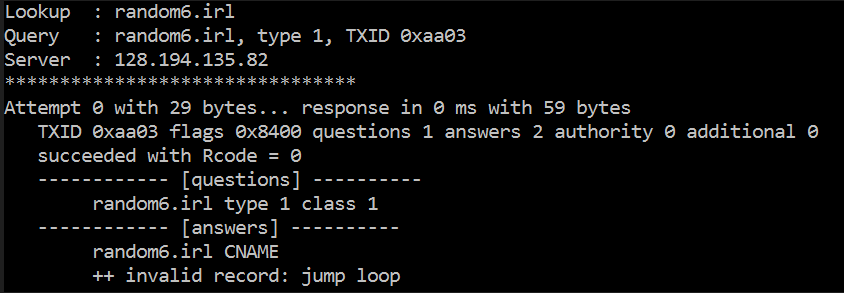
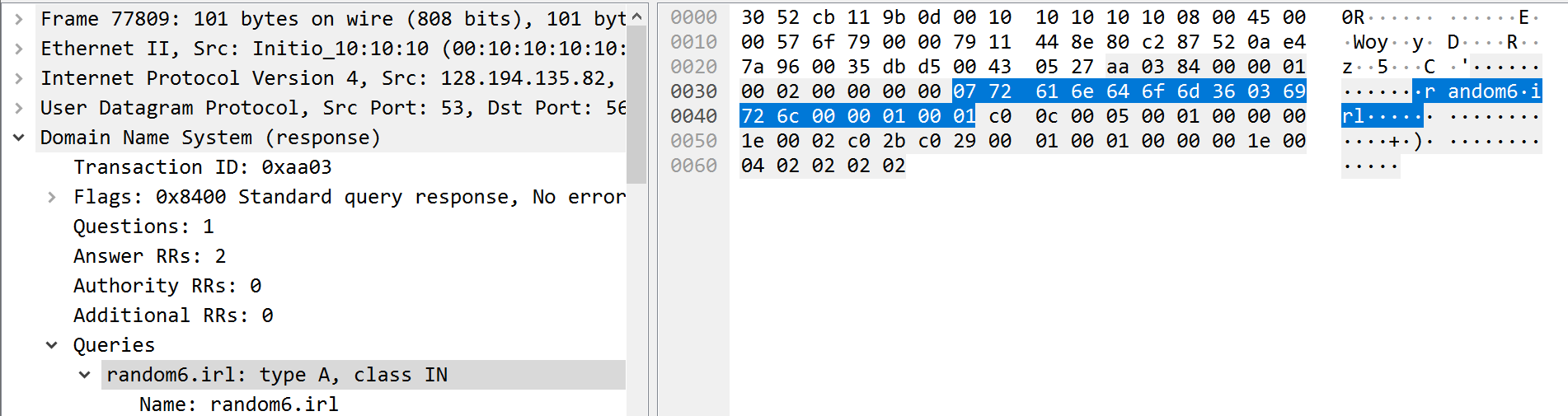
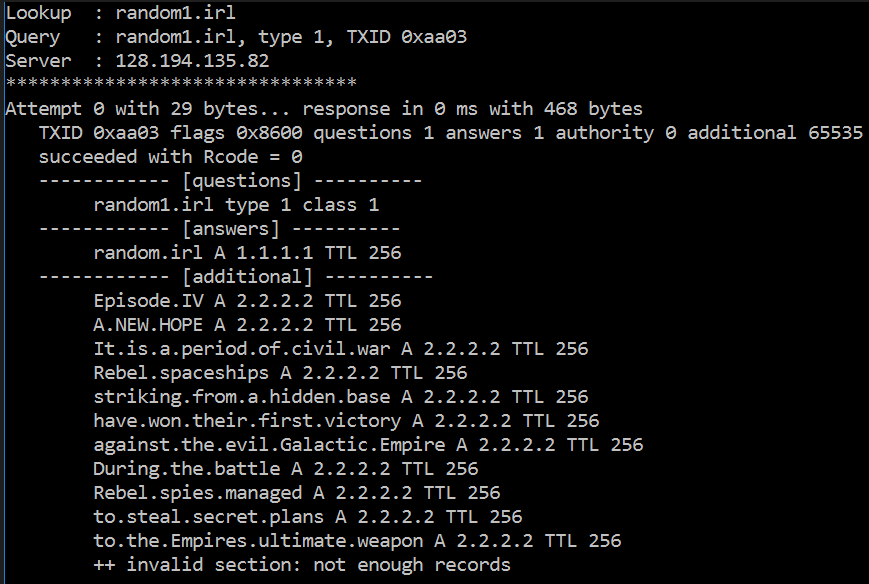
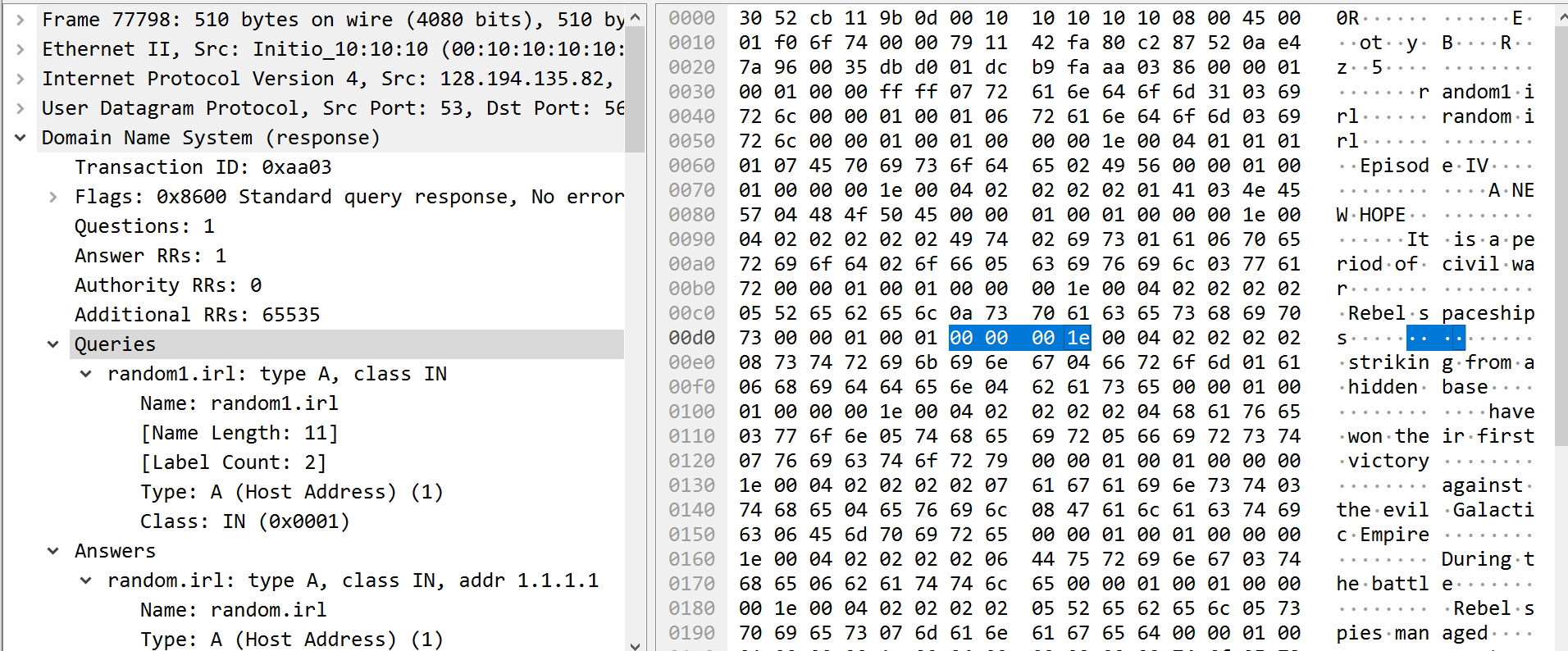
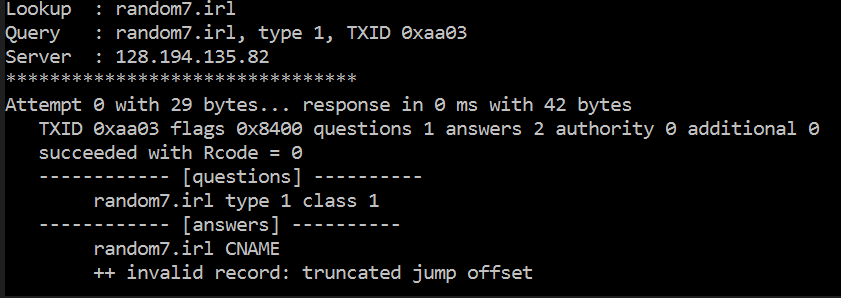
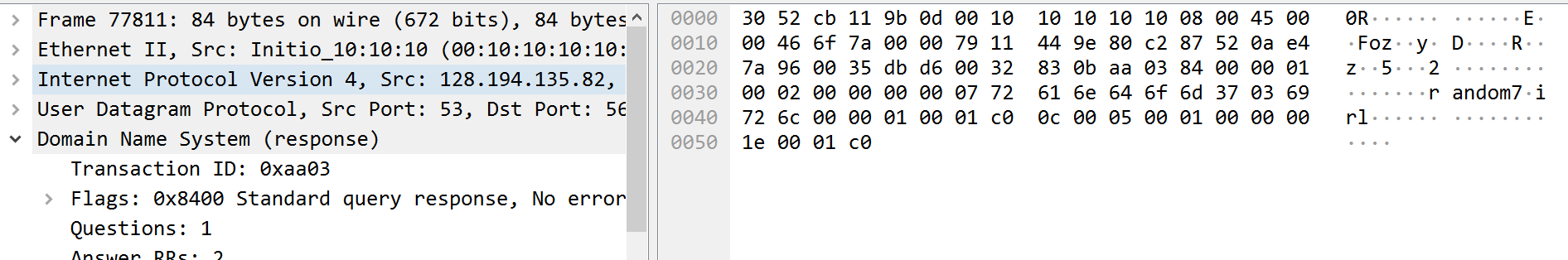
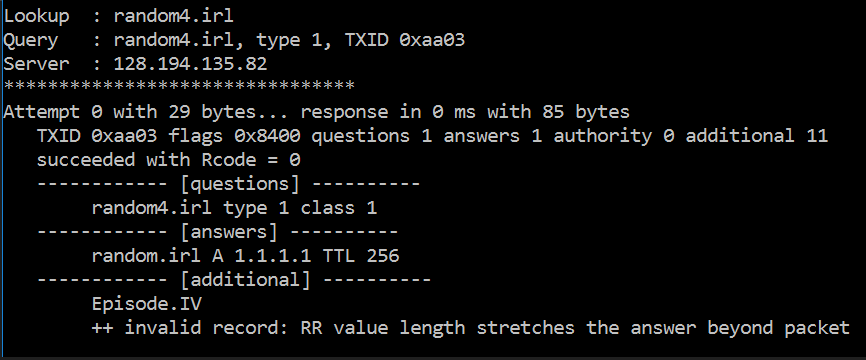
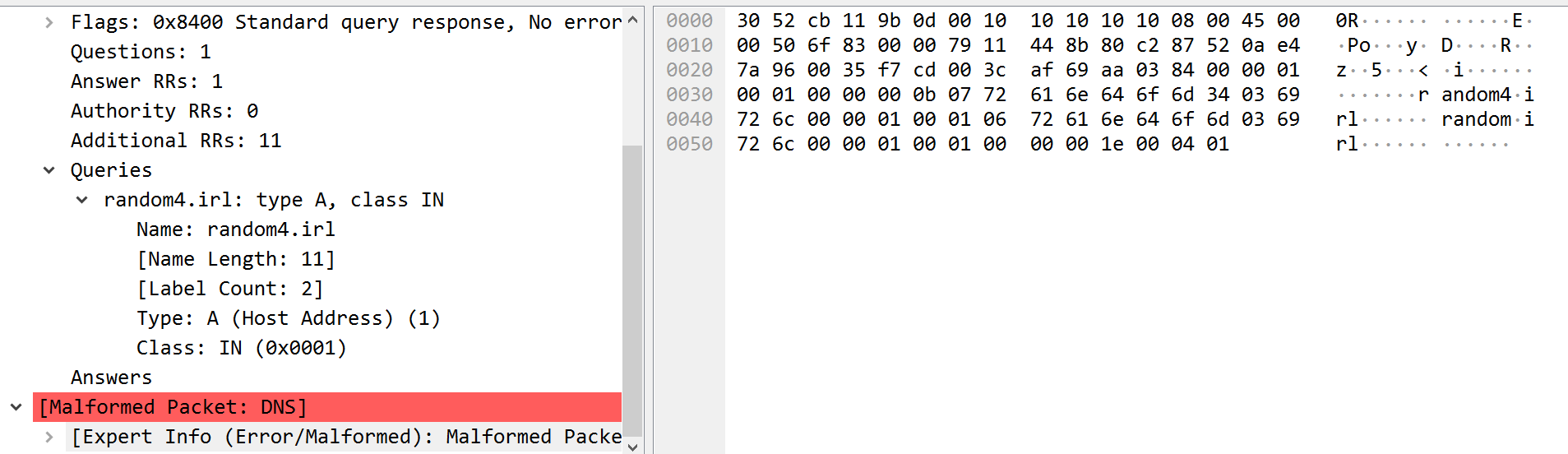
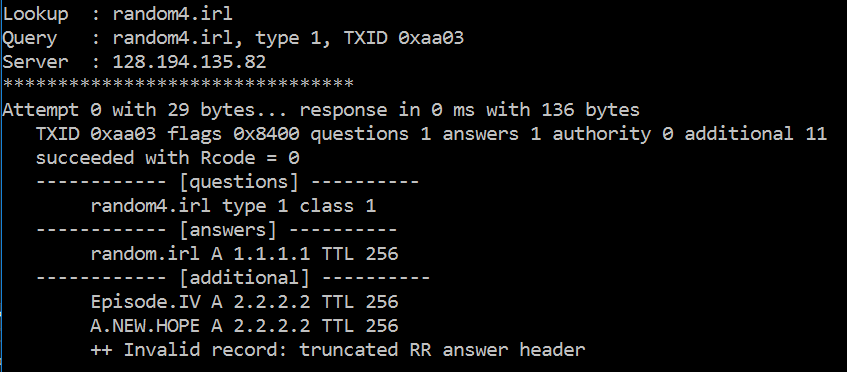
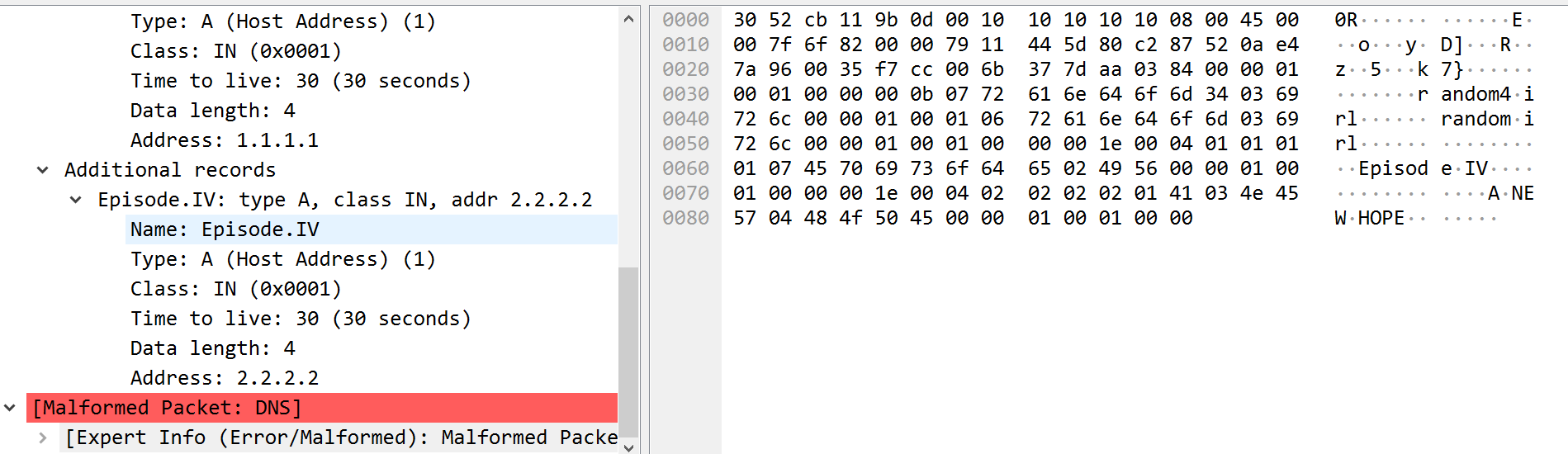
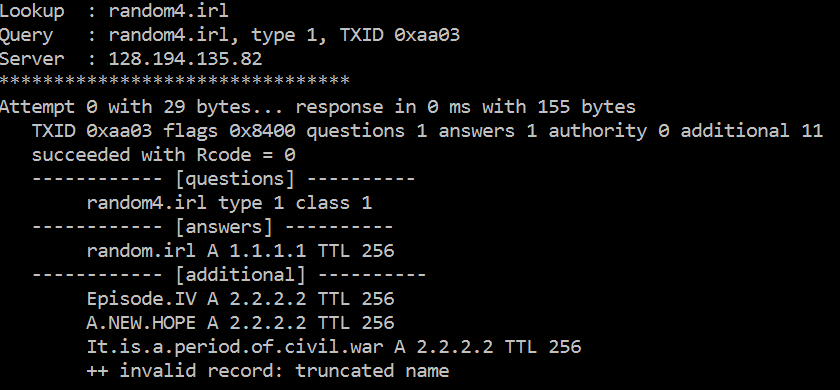
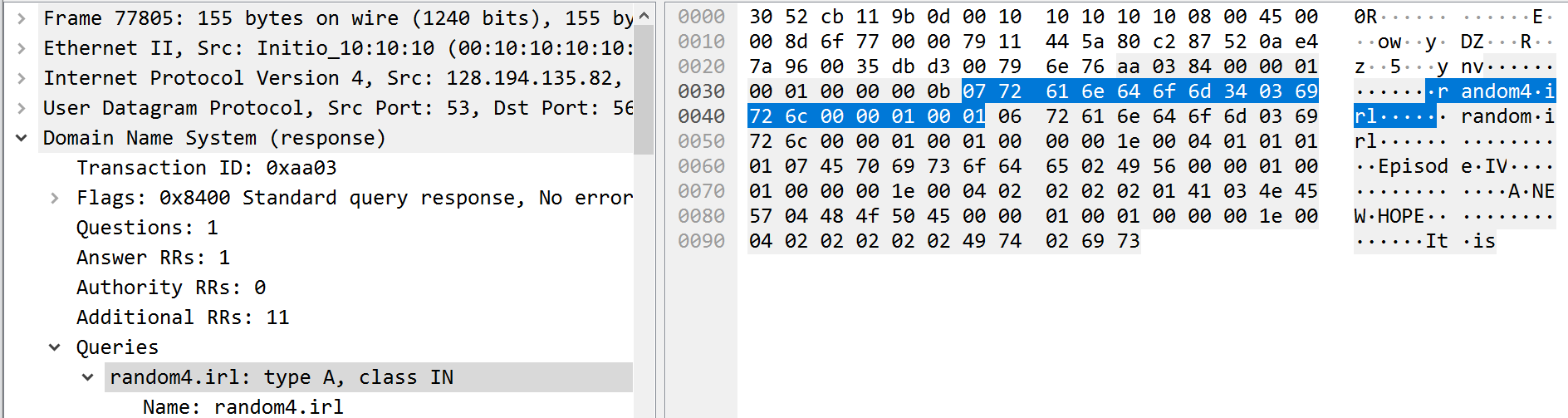
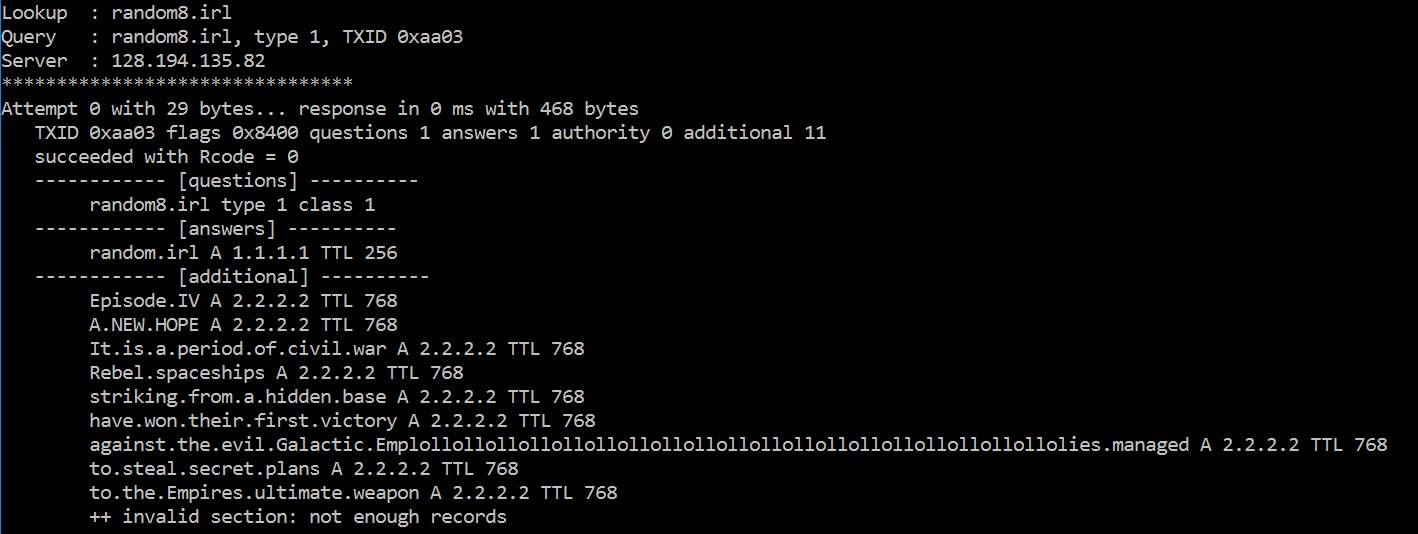
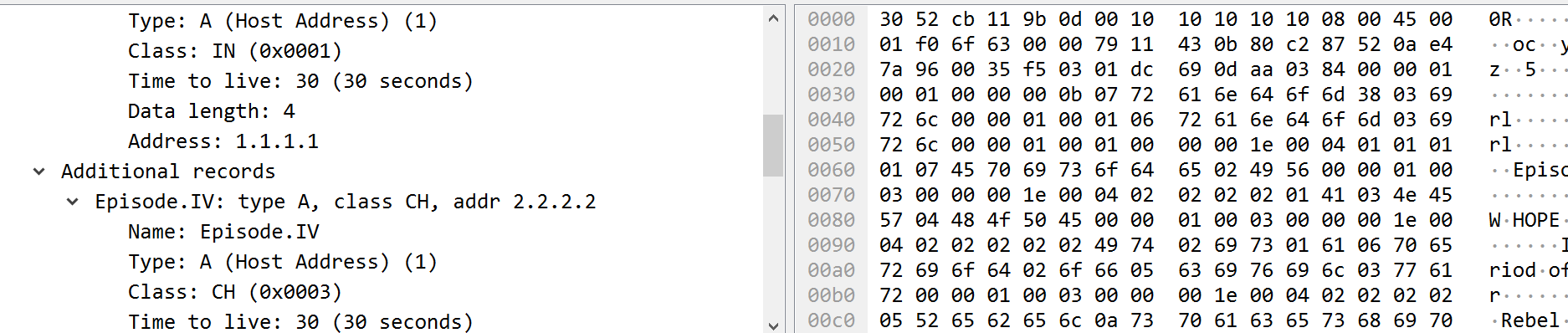
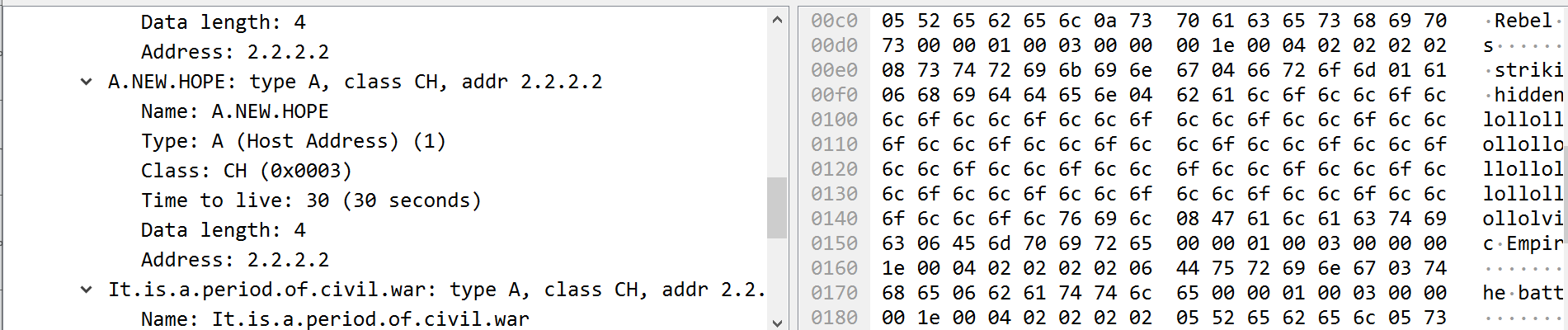
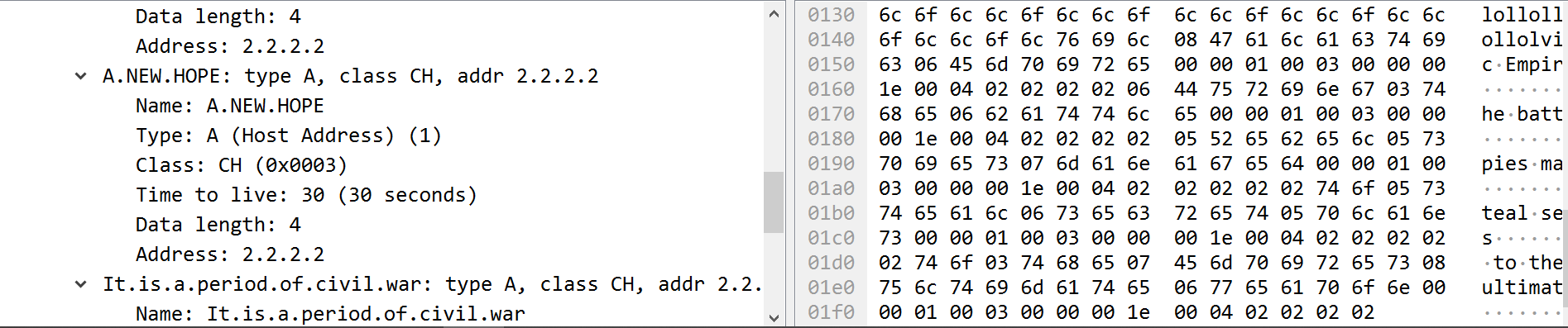
Joshua Clapp

* **Case 1:** 
  + **Random0.irl**
    - 
    - 
      * This error occurs when there is a jump into the leading 12 bytes that make up the header. This can be caught by checking the attempted jump location and throwing.
  + **Random3.irl**
    - 
    - 
      * This occurs when the server replies with too few bytes, and can be caught by checking if the minimum 12 bytes are not received from the server.
  + **Random5.irl**
    - 
    - 
      * This occurs when the jump is negative, out of the upper bound, or into the middle of another record in the buffer. For example, a jump into the resource record bytes in buffer. This can be checked with checking the byte bounds, and the offset position, ensuring it is at the leading packet.
  + **Random6.irl**
    - 
    - 
      * This occurs when there are two jumps that point at each other, and can be caught with counting the number of jumps, until it exceeds the size of the packet, then throw.
* **Case 2:** 
  + **Random1.irl**
    - 
    - 
      * This error occurs when the amount of replies in the response record is less than the number of actual replies in the buffer. This can be checked by counting the actual replies to the claimed replies and throwing if they differ.
* **Case 3:** 
  + **Random7.irl**
    - 
    - 
      * This occurs when there is a jump at the last byte of the response buffer (packet) with no location to jump to. With the current position, adding one and seeing if it is the same as the packet size.
* **Case 4:**
  + **Random4.irl**
    - 
    - 
      * This error occurs when there is corruption in the plaintext address record (IP) of the packet. It can be caught by checking the curPos position in the response buffer incrementing by one (resource length received), and if that exceeds the packet size, then throw.
    - 
    - 
      * This occurs when in the response buffer the resource record struct’s bytes have been corrupted and do not exist. It can be caught by checking the curPos in buffer + (8), after the incrementing curPos by name record and checking if the buffer at that pos is unassigned (-52).
    - 
    - 
      * This error occurs when there is an incomplete name record inside the packet. It can be caught by checking all names for empty un-assigned chars in the buffer at curPos, and throwing when found.
* **Case 5:**
  + **Random8.irl**
    - 
    - 
    - 
    - 
      * The server picks a position in the name record response buffer and injects a “lol” until the size is 510 bytes. To recreate this, I would take the sent in packet and at a reply if a condition is true, insert my string and resize the buffer accordingly. Where in the packet depends on a hidden clock that triggers and populated until the size of the whole is 510.