CSED353: Chap. 3 Exercises (due Apr. 26)

손량(20220323)

Last compiled on: Thursday 25th April, 2024, 23:32

1 Problem #1

1.1 Solution for (a)

The hexadecimal representation of the header is `c5e8c351000c5cf8`, and `61626364` for data. We can split the hexadecimal into 16-bit words and obtain `[0xc5e8, 0xc351, 0x000c, 0x5cf8, 0x6162, 0x6364]`. The sum can be calculated using the following code:

```
words = [0xC5E8, 0xC351, 0x000C, 0x5CF8, 0x6162, 0x6364]
checksum = 0
for word in words:
    checksum += word
    carryover = checksum >> 16
    checksum = (checksum & 0xFFFF) + carryover
print(f"checksum: {checksum:04x}")
```

and we obtain `0xab05`.

1.2 Solution for (b)

As the checksum is not `0xffff`, the packet contains errors.

1.3 Solution for (c)

The structure of UDP pseudoheader can be represented as follows:

Octet	0	1	2	3
0	Source IPv4 Address (32 bit)			
4	Destination IPv4 Address (32 bit)			
8	Zeroes (8 bit)	Protocol (8 bit)	UDP Length (16 bit)	

1.4 Solution for (d)

The hexadecimal part for pseudoheader is `c0a801078ddf054e0011000c`, which can be split as `[0xc0a8, 0x0107, 0x8ddf, 0x054e, 0x0011, 0x000c]`. The sum can be calculated using the following code:

```
words = [
    0xC0A8,
    0x0107,
    0x8DDF,
    0x054E,
    0x0011,
    0x000C,
    0xC5E8,
    0xC351,
    0x000C,
    0x5CF8,
    0x6162,
    0x6364,
]
checksum = 0
for word in words:
    checksum += word
    carryover = checksum >> 16
    checksum = (checksum & 0xFFFF) + carryover
print(f"checksum: {checksum:04x}")
```

And we obtain `0xffff`.

1.5 Solution for (e)

Taking IPv4 UDP pseudoheader into account, the packet is error-free in UDP layer as the checksum is valid. However, there might be errors in other layers, such as application layer so we cannot be entirely sure that packet is error-free.

1.6 Solution for (f)

By using UDP pseudoheader, the UDP checksum mechanism can detect errors outside UDP layer, such as incorrect source/destination address and routing errors.

2 Problem #2

Let $L = 1500$ bytes, $R = 10^9$ bits/s, $\mathsf{RTT} = 30$ ms and n as window size, in number of packets. Then the utilization condition can be written as

$$\frac{nL/R}{\mathsf{RTT} + nL/R} > 0.98$$

Solving this inequality, we obtain $n > 122500$[1] so the window size should be bigger than 122500 bits to have utilization greater than 98 percent.

3 Problem #3

3.1 Solution for (a)

As written in the problem statement, the initial sequence number at $t = 1$ is 111.

[1]Of course, we only consider positive values of n.

3.2 Solution for (b)

Since the segment sent at $t = 1$ is the initial segment, the segment has SYN flag set and contains payload of size 926 bytes. So the segment's length in sequence space is 907. Thus, the sequence number of the segment sent at $t = 2$ is $111 + 927 = 1038$.

3.3 Solution for (c)

The length of the segment sent in $t = 2$, in sequence space is 926, as both SYN and FIN flag are not set. Thus, the sequence number of the segment sent in $t = 3$ is $111 + 927 + 926 = 1964$.

3.4 Solution for (d)

The length of the segment sent in $t = 3$, in sequence space is 926, like the segment sent in $t = 2$. Thus, the sequence number of the segment sent in $t = 4$ is $111 + 927 + 926 + 926 = 2890$.

3.5 Solution for (e)

The length of the segment sent in $t = 4$, in sequence space is 926, like the segment sent in $t = 3$. Thus, the sequence number of the segment sent in $t = 5$ is $111 + 927 + 926 + 926 + 926 = 3816$.

3.6 Solution for (f)

Only segment sent in $t = 1$ has arrived in $t = 8$, so ACK number is $111 + 927 = 1038$.

3.7 Solution for (g)

Segment sent in $t = 1$ and $t = 2$ have arrived in $t = 9$, so ACK number is $111+927+926 = 1964$.

3.8 Solution for (h)

The segment is lost.

3.9 Solution for (i)

Since the segment sent in $t = 2$ is the last in-order segment in $t = 11$, the ACK number is $111 + 927 + 926 = 1964$.

3.10 Solution for (j)

The segment is lost.

3.11 Solution for (k)

The segment sent in $t = 1$ is acknowledged, so the sixth segment is sent in order to fill the window, whose sequence number is $111 + 927 + 926 + 926 + 926 + 926 = 4742$.

3.12 Solution for (l)

The segment sent in $t = 2$ is yet to be acknowledged, so the sender does not send any segment. None.

3.13 Solution for (m)

The segment sent in $t = 2$ is yet to be acknowledged, so the sender does not send any segment. None.

3.14 Solution for (n)

The segment sent in $t = 2$ is acknowledged, so the seventh segment is sent in order to fill the window, whose sequence number is $111 + 927 + 926 + 926 + 926 + 926 + 926 = 5668$.

3.15 Solution for (o)

The segment sent in $t = 3$ is yet to be acknowledged, so the sender does not send any segment. None.

4 Problem #4

For convenience, we denote values of $\mathsf{EstimatedRTT}$, DevRTT and $\mathsf{TimeoutInterval}$ after n-th RTT as $\mathsf{EstimatedRTT}(n)$, $\mathsf{DevRTT}(n)$ and $\mathsf{TimeoutInterval}$, respectively. Naturally, $\mathsf{EstimatedRTT}(0) = 320$ and $\mathsf{DevRTT}(0) = 39$. In addition, let n-th measured RTT as $\mathsf{SampleRTT}(n)$.

4.1 Solution for (a)

We can calculate

$$\begin{aligned}\mathsf{EstimatedRTT}(1) &= (1-\alpha) \times \mathsf{EstimatedRTT}(0) + \alpha \times \mathsf{SampleRTT}(1)\\ &= (1-0.125) \times 320 + 0.125 \times 390 = 328.75\end{aligned}$$

4.2 Solution for (b)

We can calculate

$$\begin{aligned}\mathsf{DevRTT}(1) &= (1-\beta) \times \mathsf{DevRTT}(0) + \beta \times |\mathsf{SampleRTT}(1) - \mathsf{EstimatedRTT}(0)|\\ &= (1-0.25) \times 39 + 0.25 \times |390 - 320| = 46.75\end{aligned}$$

4.3 Solution for (c)

We can calculate

$$\mathsf{TimeoutInterval}(1) = \mathsf{EstimatedRTT}(1) + 4 \times \mathsf{DevRTT}(1) = 515.75$$

4.4 Solution for (d)

We can calculate

$$\begin{aligned}\mathsf{EstimatedRTT}(2) &= (1-\alpha) \times \mathsf{EstimatedRTT}(1) + \alpha \times \mathsf{SampleRTT}(2)\\ &= (1-0.125) \times 328.75 + 0.125 \times 270 = 321.41\end{aligned}$$

4.5 Solution for (e)

We can calculate

$$\begin{aligned}\mathsf{DevRTT}(2) &= (1-\beta)\times \mathsf{DevRTT}(1) + \beta \times |\mathsf{SampleRTT}(2) - \mathsf{EstimatedRTT}(1)| \\ &= (1-0.25)\times 46.75 + 0.25 \times |270 - 328.75| = 49.75\end{aligned}$$

4.6 Solution for (f)

We can calculate

$$\mathsf{TimeoutInterval}(2) = \mathsf{EstimatedRTT}(2) + 4 \times \mathsf{DevRTT}(2) = 520.41$$

5 Problem #5

5.1 Solution for (a)

Since the server does not keep track of SYN segments, the server has to store the information elsewhere. SYN cookie utilizes the sequence number as the alternative storage.

5.2 Solution for (b)

The attacker has to guess the server's initial sequence number, and that involves guessing the server's secret number which is hashed alongside other data. The attacker won't be able to create connection as long as the server's secret number stays secret.

5.3 Solution for (c)

If all the data that is hashed alongside the secret value can be controlled by the attacker, it is possible for the attacker to create fully open connections. Since the server does not keep track of informations about SYN, the attacker can simply send ACK segment with appropriate fields (which depends on how SYN cookies are constructed in the server), and with sequence number of $\mathsf{CollectedISN} + 1$ and perform SYN flooding attack. The server won't be able to determine whether those ACK segments are legitimate, as they have no information about SYN segments received in the past.