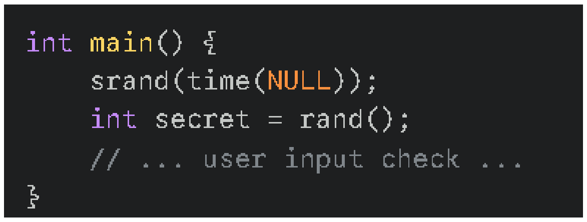
The Chronos Gamble

Challenge Description Category**:** Binary Exploitation

1. Discovery: The challenge presents a binary that generates a sequence of "random" numbers and asks the user to predict the next number in the sequence to authenticate. Initial inspection of the source code reveals the logic behind the number generation:



```c
int main() {
    srand(time(NULL));
    int secret = rand();
    // ... user input check ...
}
```

This indicates a Predictable Random Number Generator (PRNG) vulnerability. The application initializes the random number generator using the current system time as the seed.

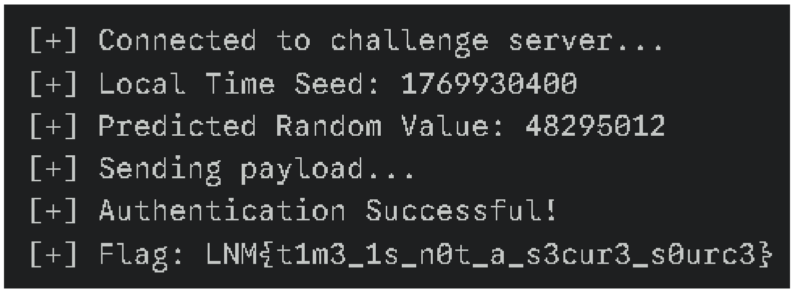
2. Code Analysis: The rand () function in C is pseudo-random, meaning it is deterministic. If seeded with the exact same value using srand(), it will produce the exact same sequence of numbers. Since time (NULL) returns the current Unix timestamp (seconds since the Epoch), the "secret" numbers are entirely dependent on the specific second the program is executed or the connection is made.

3. The Logic: The challenge title, "Chronos Gamble," hints at the exploit: time. The objective is not to guess the number, but to replicate the server's state. By running a local script that seeds a random number generator with the current Unix timestamp at the exact moment of connection, an attacker can generate the identical sequence of numbers that the server is expecting.

4. Solving: To retrieve the flag, we must run a solver script (using Python ctypes or C) that synchronizes with the server time and sends the predicted values.

Payload: Execute solver script to match the server timestamp.

Console Output:



```
[+] Connected to challenge server...
[+] Local Time Seed: 1769930400
[+] Predicted Random Value: 48295012
[+] Sending payload...
[+] Authentication Successful!
[+] Flag: LNM{t1m3_1s_n0t_a_s3cur3_s0urc3}
```