Shadow Bind – Dynamic Integrity Check Writeup

Challenge Overview

We were provided with a binary named shadow_core and a shared library libshadow.so. The binary performs an integrity check at runtime through the shared library. The objective was to reverse the validation mechanism and trigger a hidden execution path to retrieve the flag.

Initial Analysis

Running the binary normally produced no visible secrets. String analysis also revealed nothing useful. However, using ldd showed that libshadow.so was dynamically linked, indicating important logic resided there.

Reversing the Library

Inside libshadow.so, a function opens /proc/self/exe, seeks to offset 0x3000, reads six bytes, and compares them against the hardcoded string 'CHROMA'.

Executable Inspection

Inspecting shadow_core at offset 0x3000 showed null bytes by default, meaning the hidden condition was not met.

Patching the Binary

We patched the executable by writing the string 'CHROMA' at offset 0x3000 using dd or a hex editor.

Triggering Hidden Execution

Running the patched binary with the shared library loaded unlocked the concealed routine.

Recovered Flag

SECE{sh4d0w_b1nd_dyn4m1c_r3s0lv3}

Conclusion

This challenge demonstrated how shared libraries can perform runtime self-inspection and use offset-based triggers to conceal functionality. Understanding dynamic loading behavior was key to solving it.