[j00ru] Fastcalc-hardened (PWN 500)

first blood: **217**
solved **3** times

- Original fastcalc task from 2017 teaser:
  - Windows 32-bit, DEP, ASLR, stack cookies enabled.
  - Calculator for arithmetic expressions using SSE2 XMM registers as a stack, and Windows *Fibers* for scheduling.
  - Simple non-linear stack-based buffer overflow skipping the cookie and overflowing return address + 1 dword behind with a controlled 64-bit double.

[j00ru] Fastcalc-hardened (PWN 500)

first blood: **217**
solved **3** times

- Intended ASLR bypass:
  - The XMM regs are not preserved/restored between fibers.
  - `memcpy()` also uses them for big, aligned copies.
  - Trick: trigger `memcpy()` by creating a long expression, then execute ++++++ to leak the XMM6 value.
- Unintended solutions:
  - Overwrite `std::string` on stack to leak data.
  - Brute-force the very weak 8-bit ASLR.

[j00ru] Fastcalc-hardened (PWN 500)

first blood: **217**
solved **3** times

- Hardening applied:
  - Moved the `std::string` object on the stack to prevent leaks.
  - Changed the stack layout to overwrite EBP+RET instead of RET+ARG.
  - Partially fixed the XMM leak by resetting XMM registers at the beginning of the fibers (but not on preemption).
- Solution:
  - Use the XMM leak with a 106+ expression instead of 6+.
  - … or easier: still brute-force the ASLR but with a slightly more difficult gadget / ROP.

[j00ru] Fastcalc-hardened (PWN 500)

first blood: **217**
solved **3** times

```
DrgnS{M4yb3_thi5_tim3_7h3_XMM_m3mcpy_leak_g3t5_s0me_l0v3}
```