Mission 2: Shhh... It's a Secret (in the Cache)

"Your mission, should you choose to accept it... actually, you have to."

Top Secret Briefing

This message is classified. You're receiving it because the President trusts no one else for this operation. An encrypted message needs to be delivered to HQ. Traditional communication channels are compromised. Your job is to use the cache based covert channel to send the message. Learn more about cache covert channels here: Click Here

Operation Overview

Your mission is to transmit a secret message (in the form of a `msg.txt` file) from one agent (sender) to another agent (receiver) using a **Flush+Reload covert channel**. Your only communication medium is the processor's cache.

Use the covert channel to leak the message while satisfying these constraints:

- **Bandwidth Threshold:** Ensure the transfer rate is high enough for timely delivery.
- **Accuracy Tolerance:** Keep the error rate low, for example, "Mission Go" must not become "Mission No".
- You may use shared memory only to map a memory region between sender and receiver, but direct read/write on shared location is strictly forbidden, You have to load something to shared location and infer bits using access latencies. Read more about shared memory here: Click here
- Since sender and receiver are independent processes, they must implement their own synchronization protocol. This can include sending a known start-bit pattern at the beginning of transmission or periodically realigning after fixed intervals to prevent drift and ensure correct decoding.
- Direct communication through sockets, pipes, shared memory reads/writes, signals, or files is strictly prohibited. The only permitted communication channel is via cache access timing using shared memory and measuring latency through instructions like `clflush` and `rdtsc`.
- To maintain timing consistency between sender and receiver, implement clock signal using timer interrupts. This timing reference ensures that both processes are aligned during transmission and sampling.
- You must run both the `sender` and `receiver` programs simultaneously on different CPU cores. This is crucial for timing-based side-channel communication. You can achieve this using the `taskset` command. For example:

```
taskset -c 0 ./receiver &
taskset -c 1 ./sender
```

Files Provided:

- `Makefile` – Automates the compilation of the sender and receiver programs.
- `msg.txt` – Contains the message to be transmitted via the covert channel.
- `Readme.md` – Provides usage instructions and project overview.
- `receiver.c` – Code for the receiver that listens to and decodes the message using cache timings.
- `sender.c` – Code for the sender that encodes and transmits the message via cache access patterns.
- `utils.c` – Implements helper functions such as timing and cache flush operations.
- `utils.h` – Header file declaring utility functions and constants shared by sender and receiver.

Deliverables:

- A completed implementation using Flush+Reload.
- A directory named `mission2/` in zip format containing:
  - All the files including your modified `sender.c` and `receiver.c`
  - `msg.txt`
  - A PDF report detailing:
    - Your code's design and logic
    - Achieved bandwidth and accuracy
    - Link to a short demo video showing the working system

Deadline: Wednesday, 25nd June, EOD.

This message will not self-destruct, but ignoring it may sabotage your mission (and course completion).
Good luck, Agent. Operate silently, think fast, and leave no trace.