GREETINGS NOBLE ONE... LET'S BEGIN!

PAUL EDWARD

12 YEARS' EXPERIENCE

OPEN SOURCERER

IWRITE- ISPEAK - IBUILD

BOXING ADVOCATE

The race you don’t want to win: **RACE CONDITION**

Edward Paul

What is RACE CONDITION?

“A flaw in system design where multiple processes compete for the same resources, leading to unpredictable outcomes.”

Real-world impact in software:
Simultaneous access to a database record, like wallet balances.
Leads to inconsistencies or vulnerabilities.

The Cost of a Race Condition
Real-World Examples of Costly Mistakes

- Knight Capital Group (2012): Lost $440 million in 45 minutes due to a race condition.
- NASDAQ Facebook IPO (2012): Paid over $40 million in compensation after system failures.
- In October 2024, GTBank's system upgrade caused two weeks of transaction failures and account issues, highlighting the need for robust testing during upgrades.

The smallest oversight can have catastrophic consequences.

Case Study: Incident Overview Firsthand experience with Race Condition

What happened?
Attackers exploited a race condition in transaction processing.

Technical details:
Simultaneous withdrawal requests processed incorrectly.
Database recorded both withdrawals, causing double-spending.

Impact: Financial losses and system downtime.

The Investigation and Forensics “Playing Detective”

Steps taken:

- Reviewing logs to trace activity.
- **Recreating the issue in a controlled environment.**

Tools used:

- **Log analyzers, debugging frameworks, monitoring tools.**

Key finding: Lack of synchronization in database transactions.

Strategies to Avoid Race Conditions
The Fix: Building Resilient Systems

- Use distributed locks to control resource access.
- Implement idempotent operations in APIs.
- Use transaction logs and rollback mechanisms.
- Introduce rigorous testing: simulate high concurrency scenarios.

Mitigation and Countermeasures “Fixing the Flaw”

Immediate actions:

- Disable vulnerable features.
- Deploy patches.

Long-term strategies:

- Introduce locking mechanisms for transaction handling.
- Improve monitoring and alert systems.

Preventive Measures and Best Practices “Avoiding Race Conditions”

Best practices:

- Use locks or transactions in database operations.
- Test concurrency under high loads.
- Conduct thorough code reviews focused on timing issues.
- Develop with security in mind:
- Anticipate edge cases.
- Regularly test for vulnerabilities.

Live Demonstration

- A simulation of the race condition vulnerability.
- Walkthrough of the exploit and the implemented solutions.

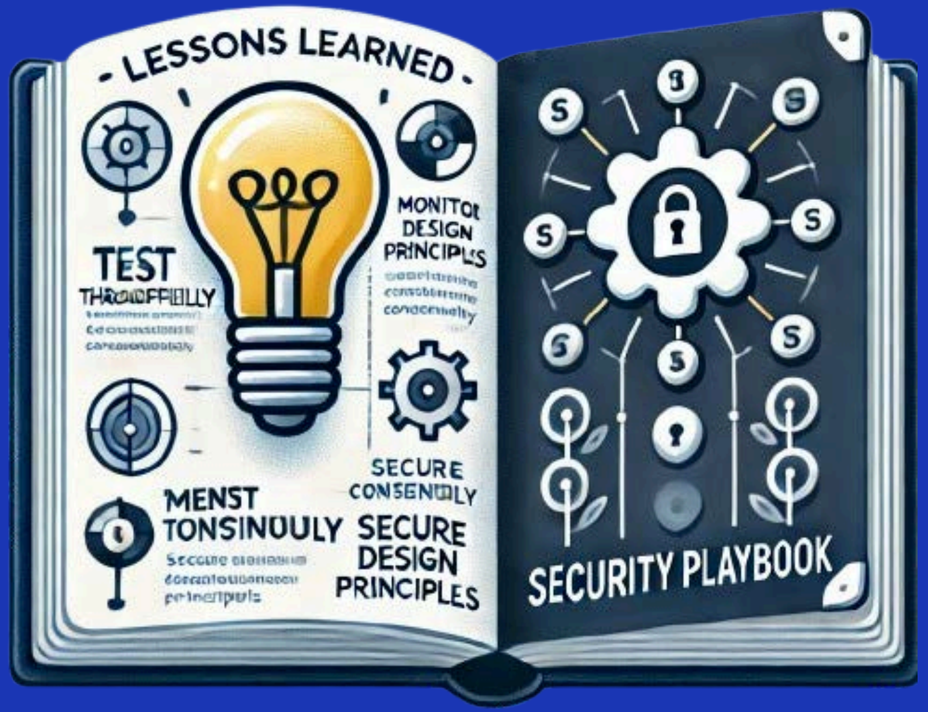
Lessons Learned

Key insights:

- Small flaws can lead to catastrophic outcomes.
- Proactive testing and monitoring can prevent major issues.
- Security requires a team effort.

How to apply these lessons to other systems:

- Emphasize secure design and testing at every stage.



Conclusion:
The Importance of Vigilance

The critical role of security in software development:

- It’s not a one-time effort; it’s continuous.
- Test thoroughly, monitor systems, and never underestimate small flaws.

"Security isn’t a sprint—it’s a marathon.

Let’s Connect

EDWARD PAUL

THEINFINITYPAUL

THEINFINITYPAUL

INFINITY PAUL