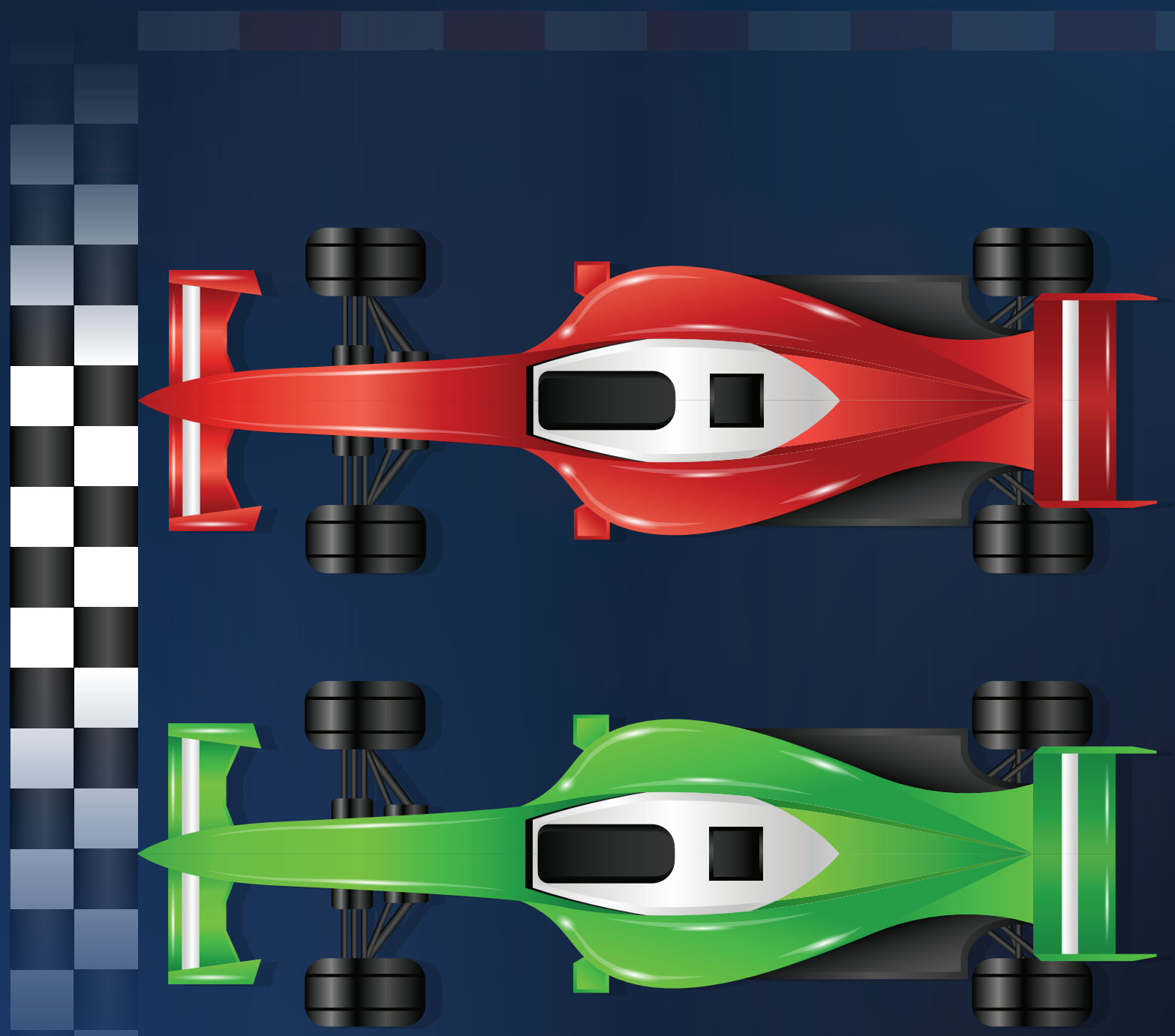
SECURITYBOAT

Frontline Of Your Business

RACE CONDITION HANDBOOK



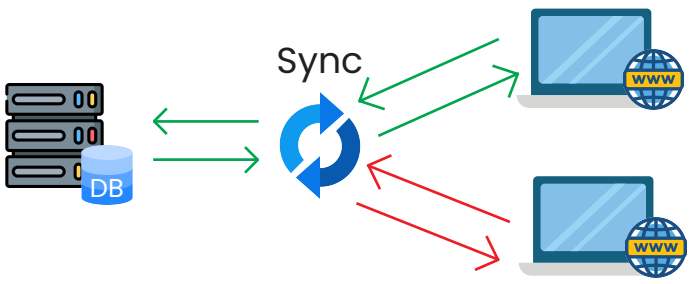
www.securityboat.net

SECURITYBOAT
Frontline Of Your Business

Table of Contents

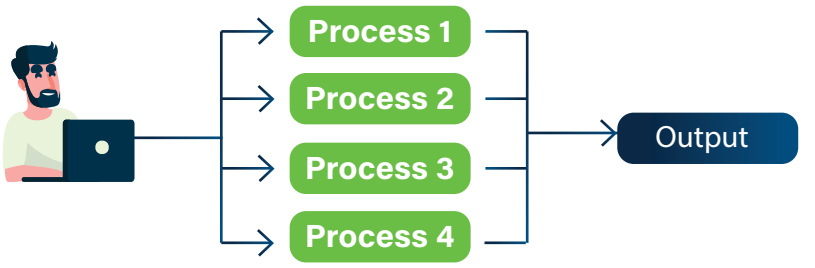
SECURITYBOAT
Frontline Of Your Business

1. Background Information



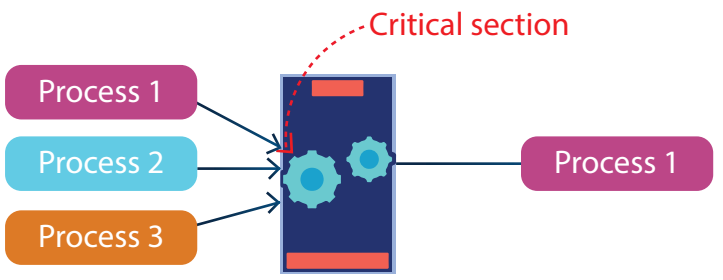
Synchronization

Synchronization mechanisms, such as locks, mutexes, and semaphores, are used to coordinate access to shared resources and prevent concurrent modifications.



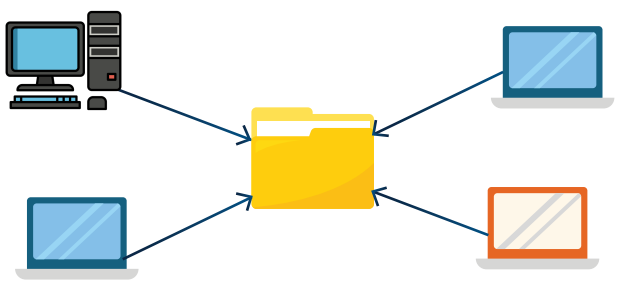
Concurrency

Concurrency involves executing multiple tasks or processes simultaneously, often to improve system performance or responsiveness.



Critical Sections

Critical sections are code segments that access shared resources and must be executed atomically or with proper synchronization to prevent race conditions.

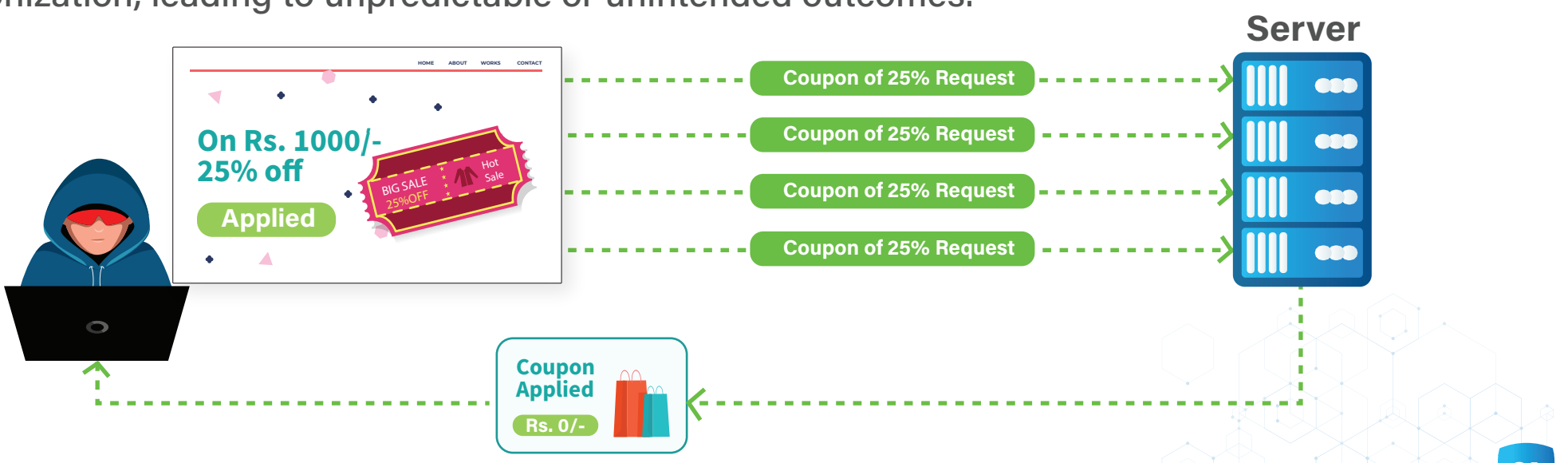


Shared Resources

Shared resources are data structures, files, or other system components accessed by multiple threads or processes concurrently.

2. What is Race Condition

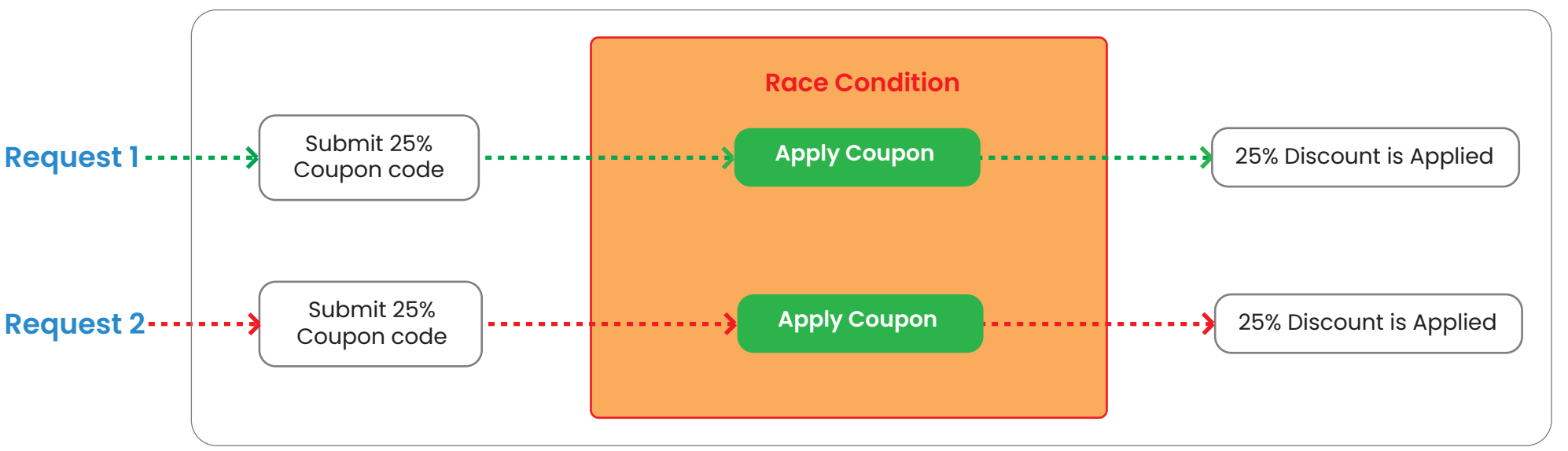
Race conditions is a vulnerability that occur in concurrent systems, where the behaviour of the system depends on the sequence or timing of events. These vulnerabilities arise when multiple threads or processes attempt to modify shared resources concurrently without proper synchronization, leading to unpredictable or unintended outcomes.



3. Attacks

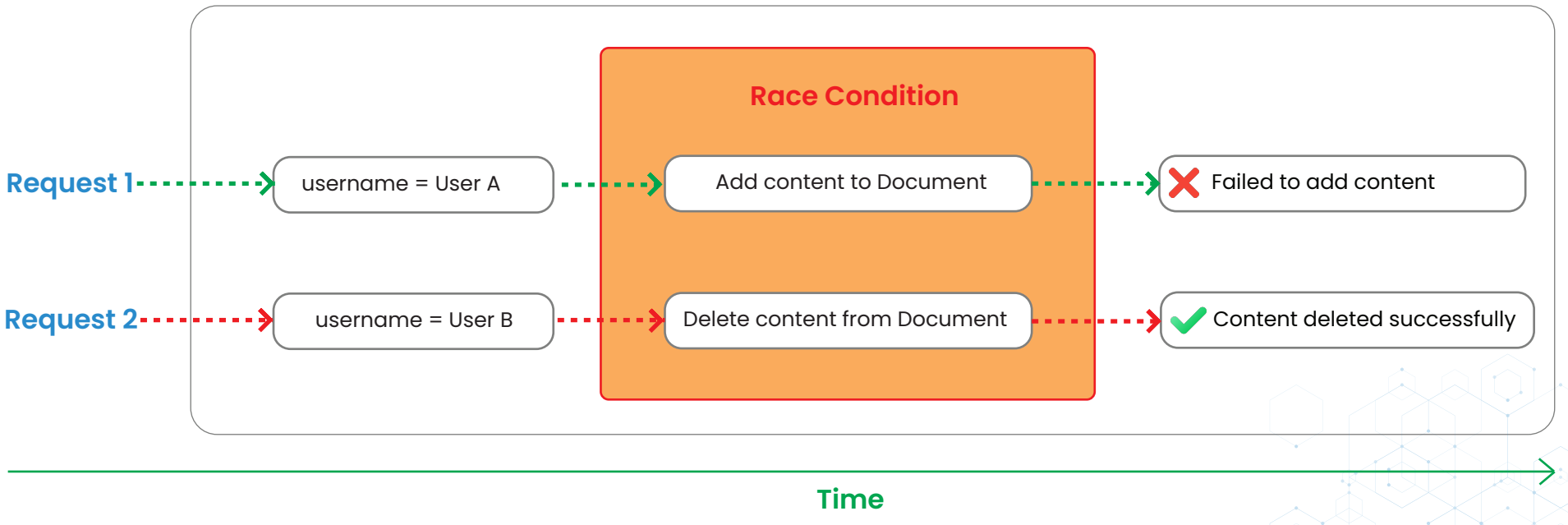
3.1 Boundary Exceeding Race Condition

Boundary exceeding race conditions occur when multiple processes or threads attempt to surpass predefined constraints established by an application's operational rules at the same time. An attacker can send multiple requests concurrently, by aiming to exploit the short time intervals between the validation of rules and their actual application. This allows the attacker to deliberately breach limitations set by the system. Consequently, the attacker can repetitively claim discounts, redeem gift cards multiple times, manipulate ratings, or bypass enforced usage limits.



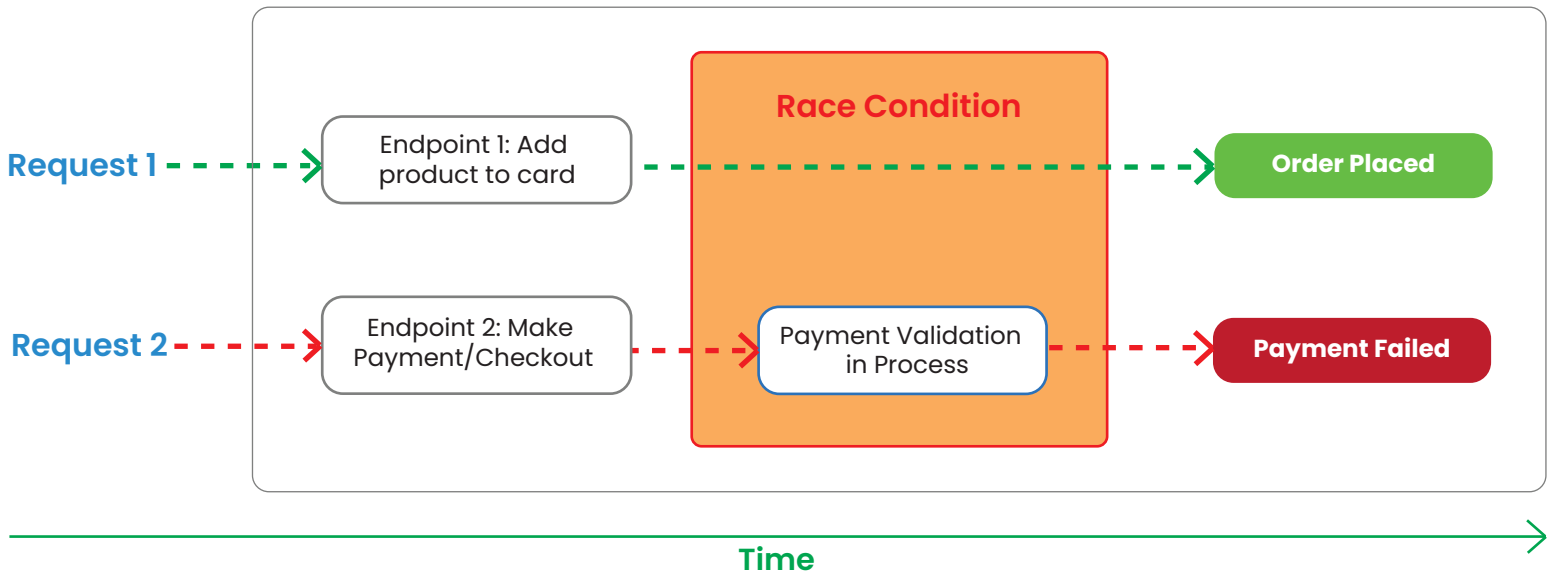
3.2 Single-Endpoint Race Condition

In a single-endpoint race condition, attackers exploit vulnerabilities where multiple processes or threads compete for control over the same endpoint or resource simultaneously. In a collaborative document editing platform where multiple users can simultaneously edit a document, a single endpoint race condition can occur, posing challenges to data integrity and collaborative efforts. For example, when User A and User B edit the same document concurrently, modifications made by one user may conflict with those made by another if the system fails to handle simultaneous edits effectively. If User A deletes a paragraph while User B simultaneously adds content to the same paragraph, the result could lead to data loss or inconsistencies in the document.



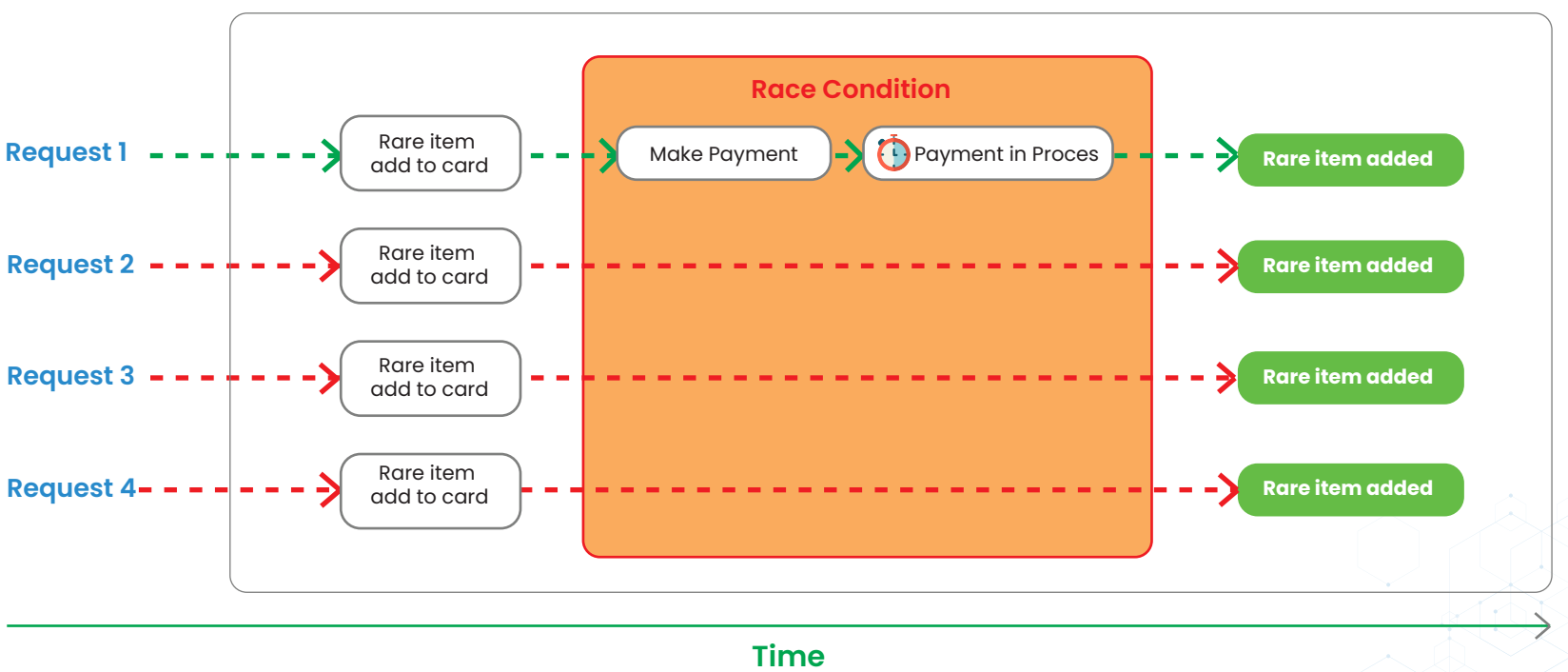
3.3 Multi-endpoint Race Condition

In a multi-endpoint race condition scenario, attackers exploit vulnerabilities across various components or resources within a system to orchestrate sophisticated attacks. For instance, consider an e-commerce platform where users can add items to their shopping carts and proceed to checkout. Attackers may exploit the multi-endpoint race condition by initiating simultaneous actions across different endpoints, such as adding items to the cart while manipulating payment details during the checkout process. By doing so, attackers aim to disrupt the flow of transactions and exploit inconsistencies between endpoints to gain unauthorized access or execute fraudulent activities. For instance, they might manipulate the cart contents to inflate the total price while simultaneously attempting to bypass payment verification mechanisms.



3.4 Time Sensitive Race Condition

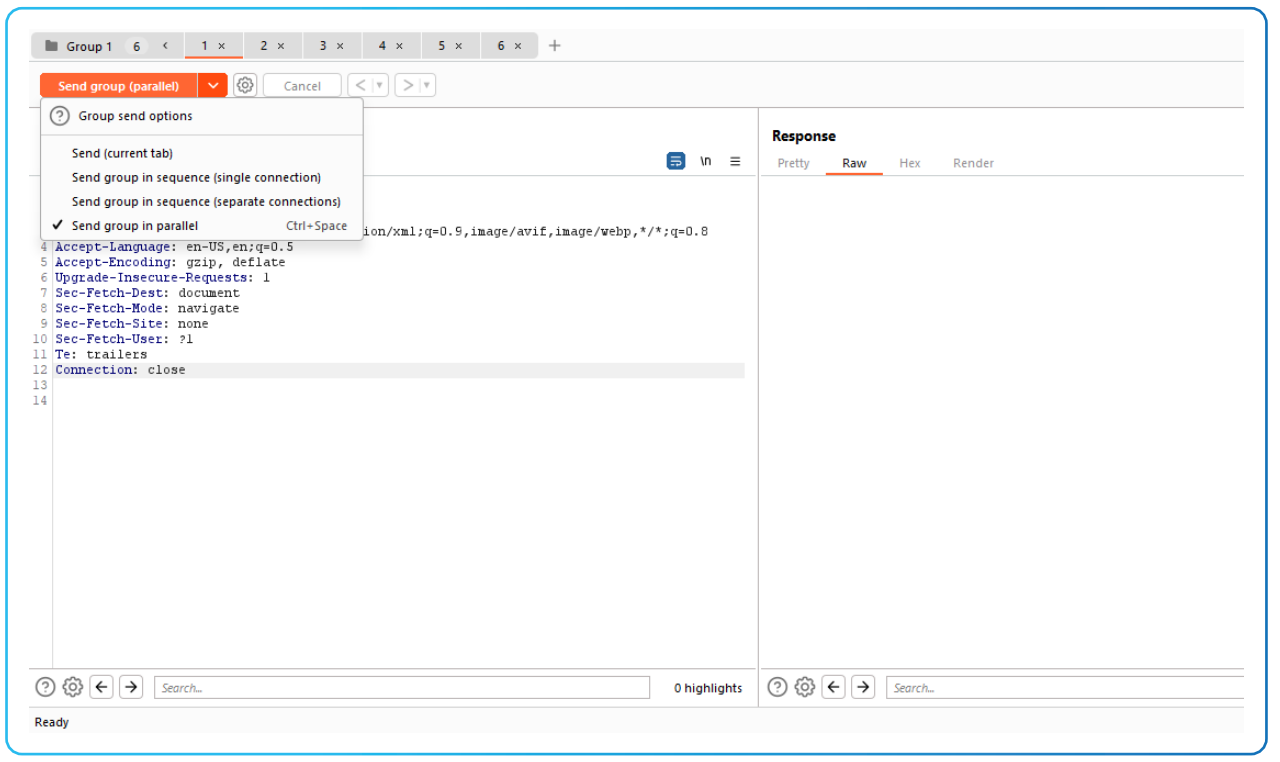
Time-based race condition is a security vulnerability where the outcome of a race condition depends on the timing of events or processes. Attackers exploit this timing dependency to manipulate or disrupt normal system behaviour, often leading to unauthorized access, data corruption, or other security issues. For example, in online gaming, race conditions occur when simultaneous transactions, like buying or selling items, disrupt the system's ability to handle data modifications accurately. Attackers exploit this vulnerability by initiating transactions in rapid succession, aiming to duplicate rare items or gain unfair advantages. For instance, an attacker may submit a purchase request for a rare item while the system is still processing a legitimate player's request, resulting in unintended duplicate purchases. This manipulation disrupts the game's economy and fairness, as attackers gain valuable items without proper effort.



4. Tools

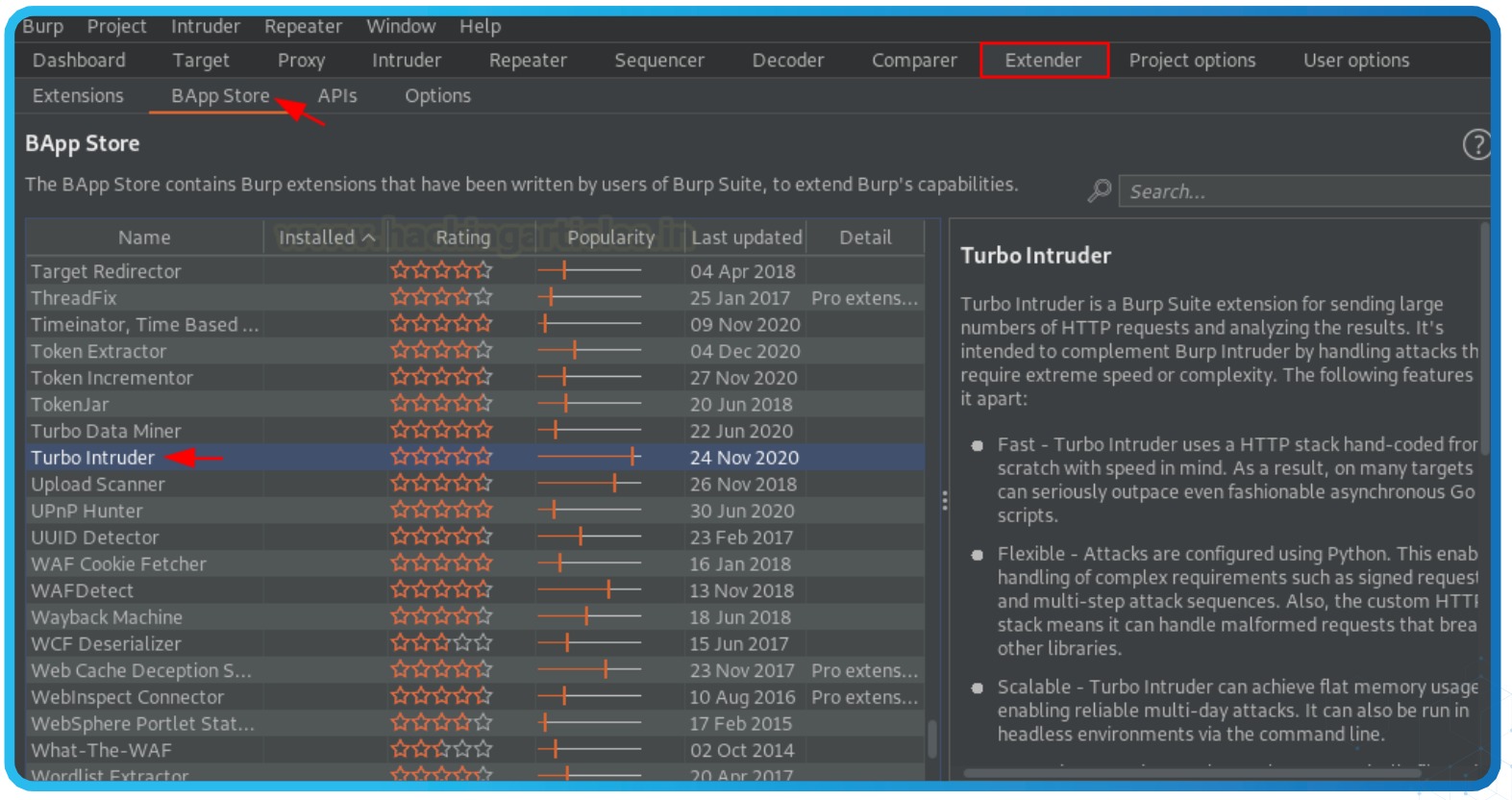
4.1 Burp Suite - Repeater

The "Group send options" feature in Burp Repeater streamlines the process of sending multiple HTTP requests by allowing them to be grouped and sent with a single click. Users have the flexibility to send requests either simultaneously (in parallel) or sequentially (one after the other), enhancing efficiency and convenience during testing and analysis.



4.2 Burp Suite – Turbo Intruder

Turbo Intruder works best for complicated attacks, especially ones that need many tries, requests with different timings, or a lot of requests altogether.



5. Impact

5.1 Financial Losses

In systems handling financial transactions or critical operations, race condition vulnerabilities can result in financial losses, fraudulent activities, or transactional errors. For example, race conditions in financial trading platforms may enable attackers to exploit price differentials, manipulate trades, or execute unauthorized transactions.

5.2 Data Corruption

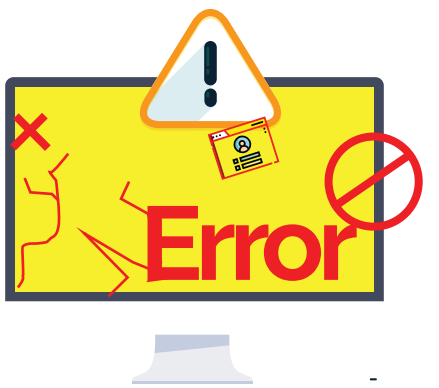
Concurrent access to shared resources without proper synchronization can result in data corruption or inconsistency. Race conditions may cause unexpected changes or overwrites to critical data, leading to inaccurate information or system malfunction.

5.3 Security Breaches

Exploitation of race conditions can enable attackers to manipulate system behaviour, bypass security controls, or gain unauthorized access to sensitive data or resources. This could result in data breaches, leakage of confidential information, or unauthorized privilege escalation.

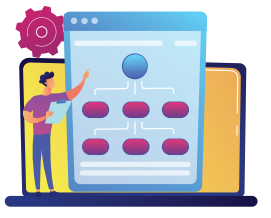
5.4 Application Crashes

Unhandled race conditions can cause software instability and lead to application crashes or unexpected termination. In scenarios where critical operations are interrupted or corrupted due to concurrent access, the application may become unresponsive or unusable, impacting user experience and productivity.



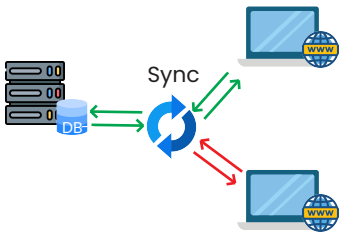
6. Prevention

6.1 Implement Order Matching Algorithms



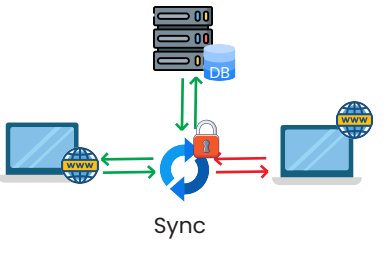
Implement advanced order matching algorithms that assess parameters such as order type, price, and time priority, ensuring fair execution by processing orders sequentially, one after the other.

6.2 Transaction Serialization



Implement transaction serialization mechanisms to ensure that orders are processed in a sequential and deterministic manner, reducing the risk of race conditions.

6.3 Transaction Locking



Utilize transaction locking mechanisms to prevent simultaneous access to critical resources during order execution, thereby reducing the likelihood of race conditions and unauthorized access.

6.4 Concurrency Testing

Conduct rigorous testing, including stress testing and concurrency testing, to identify and address potential race condition vulnerabilities. Simulate real-world scenarios with high levels of concurrent access to assess the system's resilience and identify weak points.

6.5 Rate Limiting and Throttling

Enforce rate limiting and throttling mechanisms to control the rate of incoming orders and prevent rapid, high-volume trading activities that could exploit timing vulnerabilities.

6.6 Source Code Review

Code reviews address race conditions by examining concurrent access, applying safeguards like locking mechanism and transactional processing thus preventing unintended outcomes.

6.7 Regulatory Compliance

Ensure compliance with regulatory requirements and industry standards governing financial trading activities, including transparency, fairness, and market integrity regulations.

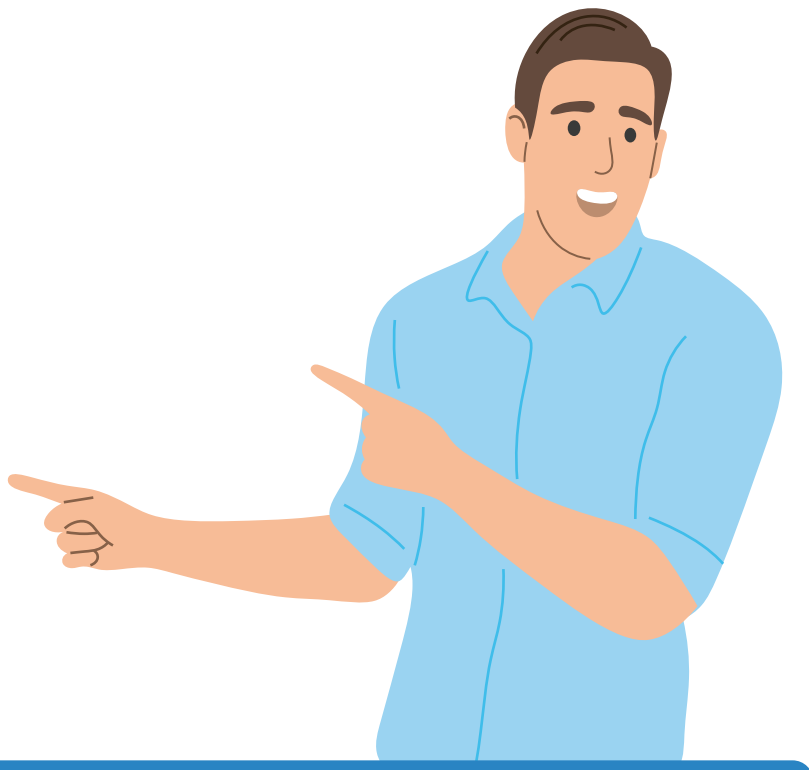
7. Additional Resources and References

- https://portswigger.net/web-security/race-conditions
- https://workbook.securityboat.net/resources/web-app-pentest/race-conditions
- https://book.hacktricks.xyz/pentesting-web/race-condition
- https://cwe.mitre.org/data/definitions/367.html

SECURITYBOAT

Frontline Of Your Business

RACE CONDITION HANDBOOK



Scan QR Code to Download Handbook

www.securityboat.net