> **Info**
>
> Основано на статье с хабра

> **Important**
>
> **ВАЖНО!**
> Гонка - весьма сложная уяза. Здесь я распишу базовые понятия. Вам придется много практиковаться для ее нахождения/экспуатации

Race Condition

Состояние гонки — ситуация, когда *два и более* потока или процесса о*дновременно читают и записывают общие данные* без должной синхронизации. Это приводит к непредсказуемым и, как правило, нежелательным результатам

> **Info**
>
> Например интернет-магазин позволяет применить купон «best10» дважды, если запросы проходят «параллельно»

Почему возникает гонка?

Чтобы разобраться с этим вопросом, нам важно усвоить несколько терминов:

1. **Критическая секция** – участок кода, обращающийся к общим ресурсам
2. **Отсутствие синхронизации (mutex например)** – несколько потоков могут зайти в критическую секцию одновременно

Так вот, большинство приложений используют *много*(*поточность/процессность*) или *асинхронность*. Есть нет норм синхронизации - гонки не избежать

доп пример

SQL-сервер при обновлении создает временный файл — злоумышленник может подменить его, чтобы получить права администратора

1. **Сервер создаёт временный файл**
   При обновлении администраторской таблицы SQL-сервер сначала проверяет существование temp-файла, затем создаёт его и записывает новые данные (например, с обновлением прав пользователя)
2. **Окно гонки возникает между "check" и "use"**
   В фазе между проверкой и открытием файла присутствует небольшой временной промежуток — именно его используют злоумышленники

   > The primary challenge is timing the requests so that at least two race windows line up, causing a collision. This window is often just milliseconds and can be even shorter.

3. **Подмена файла**
   Во время этого окна атаки злоумышленник быстрее создаёт символическую ссылку или свой собственный файл с тем же именем. При открытии temp-файла сервер, считая его безопасным, записывает туда содержимое уже злоумышленника, либо же записывает важные данные (пароли, роли администратора) в неверный файл

- **Перезапись системного ресурса**
  Запись производится в подставной файл, например `/etc/shadow` в Linux или реальную таблицу SQL. В итоге у атакующего возникают полномочия администратора — он получает доступ к данным или системе целиком

Типы Race Condition

1. Limit-overrun / TOCTOU (Time of Check to Time of Use)

> Race condition, когда выполняется проверка условия (check), а затем — действие (use), но между ними есть временное окно. Злоумышленник может успеть прыгнуть в это окно и обойти проверку

Механика :

1. Поток A читает, например, флаг `used = false`.
2. Поток B делает то же самое одновременно.

3. Оба проходят проверку и вносят изменение — купоны применяются дважды, деньги списываются дважды и тд

Например Коды скидки (купон «использован» проверяется, но не помечается до применения)

2. Hidden substates

> Менее очевидный тип: работает с подсостояниями
> Race condition, спрятанная в сложной, многошаговой логике, где приложение временно входит в промежуточное состояние. Атакующий может использовать именно это состояние

- Нужно:
  1. **Прогнозировать** подсистемы, где возможна «гонка»
  2. **Искать подсказки** — неожиданные ответы
  3. **Доказать концепт** — минимизировать число запросов до двух, оптимизировать тайминг

Пример:

- Первый запрос начинает сброс пароля, но ещё не завершил шаг `mfa_enabled = true`.
- Второй запрос выполняет сброс прямо в промежутке — bypass MFA

3. Time-Sensitive Attacks

- Используется, когда механизм зависит от меток времени.
- Пример: токен сброса пароля генерируется на основе timestamp — два запроса в один момент — один токен Habr

---

Обнаружение

1. Найти конечную точку с ограничением по количеству или скорости.
2. Отправить множество запросов за короткий промежуток.
3. Добиться, чтобы два окна гонки пересеклись — это сложно из-за jitter’а и задержек

Инструменты:

- **Burp Turbo Intruder** — параллельные запросы с питоновским скриптом
- **racepwn** — golang-утилита для конфигурирования гонок

Эксплуатация

- HTTP-запрос можно разорвать на части: отправить заголовки, задержать тело, потом добить.
- Сервер откроет соединение и обработает части «параллельно» — пригодно для атак

Практика:

- https://portswigger.net/web-security/race-conditions/lab-race-conditions-limit-overrun