Friend.tech App UX lack of synchronization + excessive ETH sent to the contract not returned / locked

1. Issue description

There are two issues here resulting in users losing money when interacting with the Friend.tech application. First problem is related to the lack of UX synchronization with the current blockchain status resulting in using outdated prices to fill out the transaction data. Secondly when such a situation happens (usually in high volume keys, but it's also possible when a user has bad luck and someone sells a key just before he hits buy) the excessive ETH is locked forever in the FT contract (0xcf205808ed36593aa40a44f10c7f7c2f67d4a4d4) as there's no refund or withdraw functions implemented.

As of this writing, 113,924 transactions have been identified with this issue, leading to an excess of 442 ETH locked in the smart contract (2.64% of the Total Value Locked).
SQL: TVL according to smart contract logs

2. Impact

Who is affected:

- All users using the Friend.tech application to buy keys

Concerning the affected funds:

Filtering for transactions executed through the UI and where the difference between sent ETH and "should have sent" ETH is explainable through differences in supply (i.e. smart contract is saying that a trader bought key number 10, whereas ETH sent to the contract corresponds to what should have been paid to acquire a key when supply is 12), we get that 43,054 transactions have resulted in locking 187 excess ETH and affecting 14,390 traders.

SQL: Filtered transactions where excess ETH is sent

About these transactions: There are still anomalous cases (for example supply differences of 10+) that are probably not derived from natural app usage. However, most supply differences (+90%) are 1,2 or 3 as seen in the image below:

supply_difference	transactions
1	32583
2	5069
3	2087
4	920
5	662
6	367
7	741
8	160

SQL: Transactions by supply difference

Average difference per transaction: 0.0035119224186117423ETH ($8)
Median difference per transaction: 0.000832562481446122 4ETH ($1,96)

There are no transactions where the amount of ETH sent to buy keys is lower than what it should have been. In the same way, there are no transactions where the smart contract is sending unexpected ETH amounts to the trader, subject or the protocol

Below is a link that aims to show the issue addressed in this report. The highly anticipated arrival of CBBank to Friend.Tech resulted in many transactions where users paid excess ETH to acquire this key.

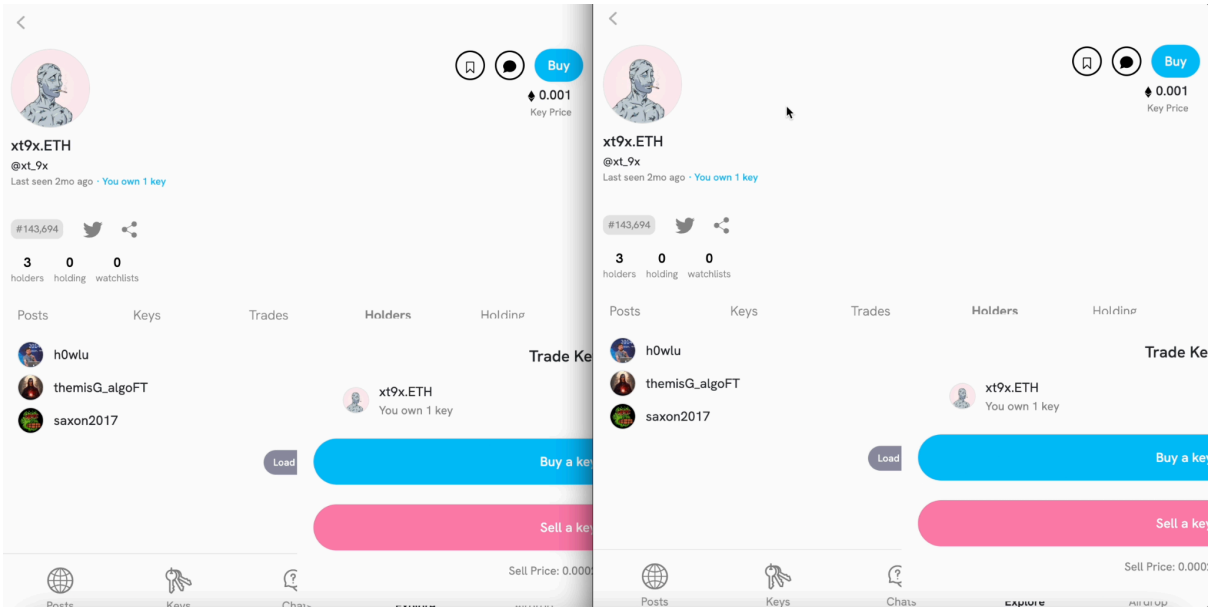
This link shows every transaction ordered by block number (from earliest to latest). SC_supply means what was the supply when the user bought according to the smart contract and bt_supply means what was the supply according to the amount of ETH sent. In the transactions where both values are equal, as expected, the value in bt_supply is null.

CBBank's key launch as an example

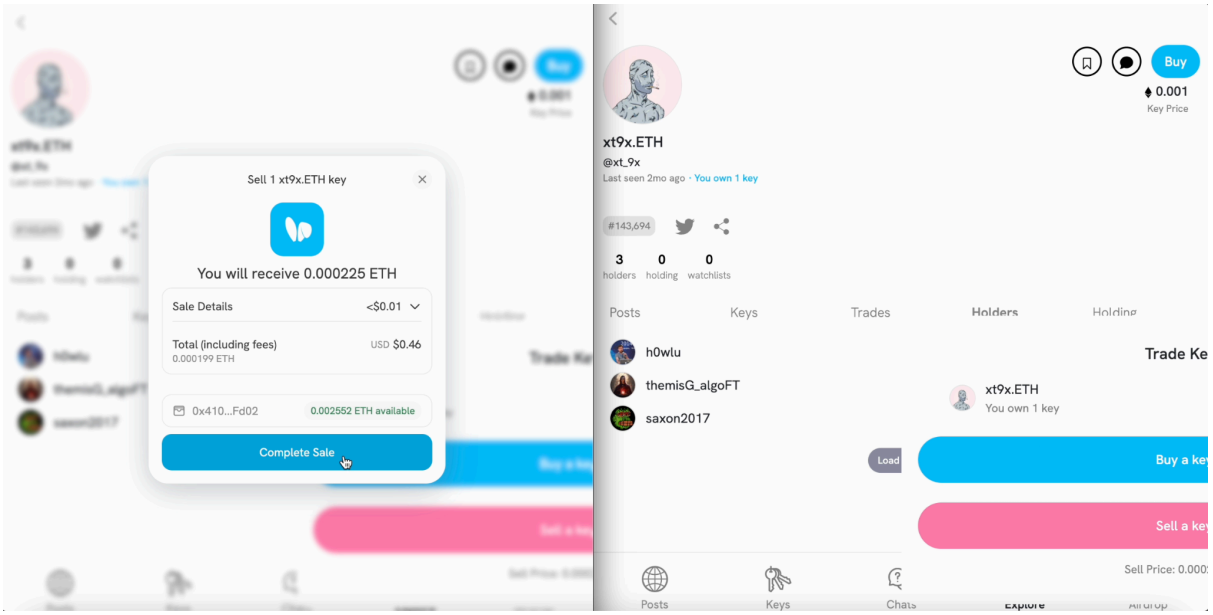
3. Proof of Concept

To reproduce the issue follow the steps below - it can be done using the same account so its easier:

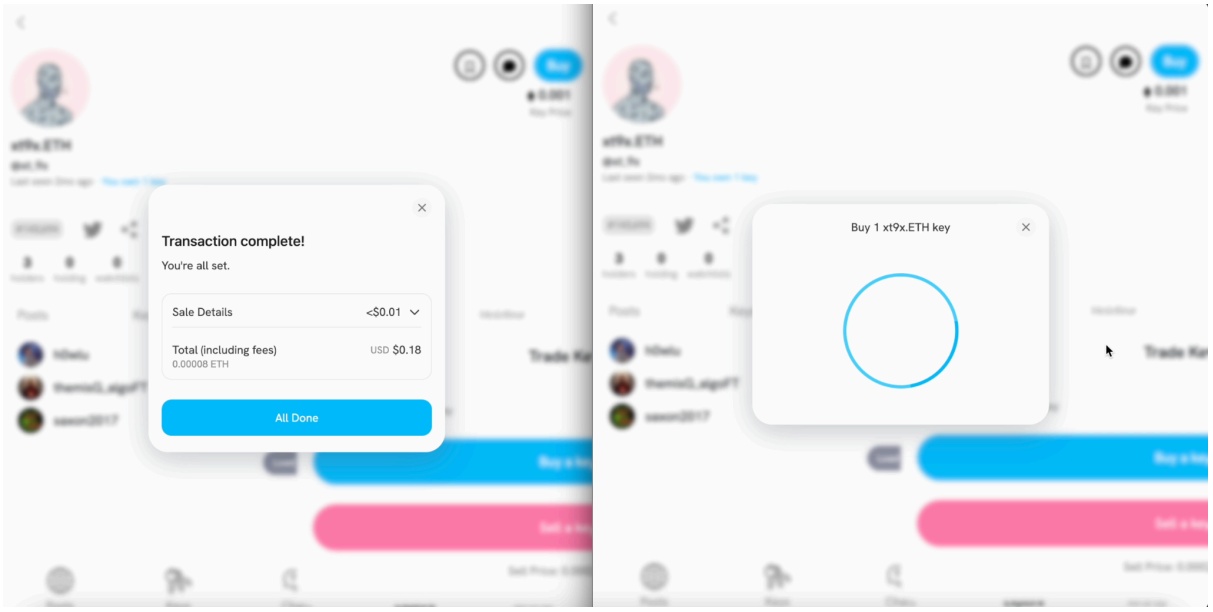
1. Open two browser windows on a profile you own a key and click ‘buy’ on both



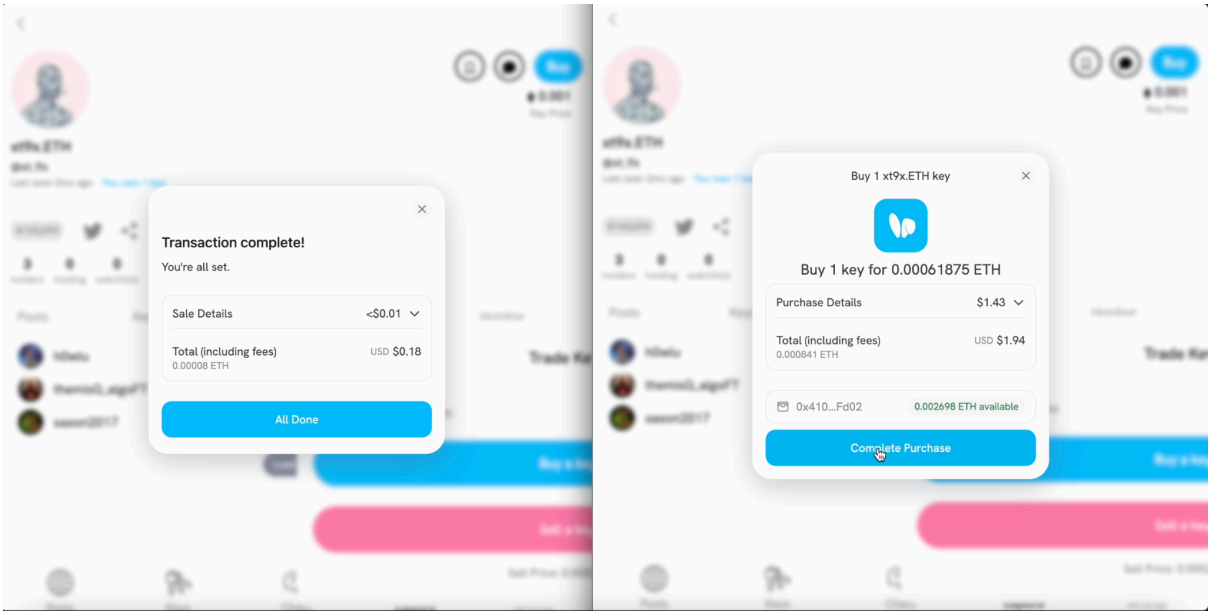
2. On the left window click ‘sell’ -> ‘Complete Sale’ and wait for it to be completed



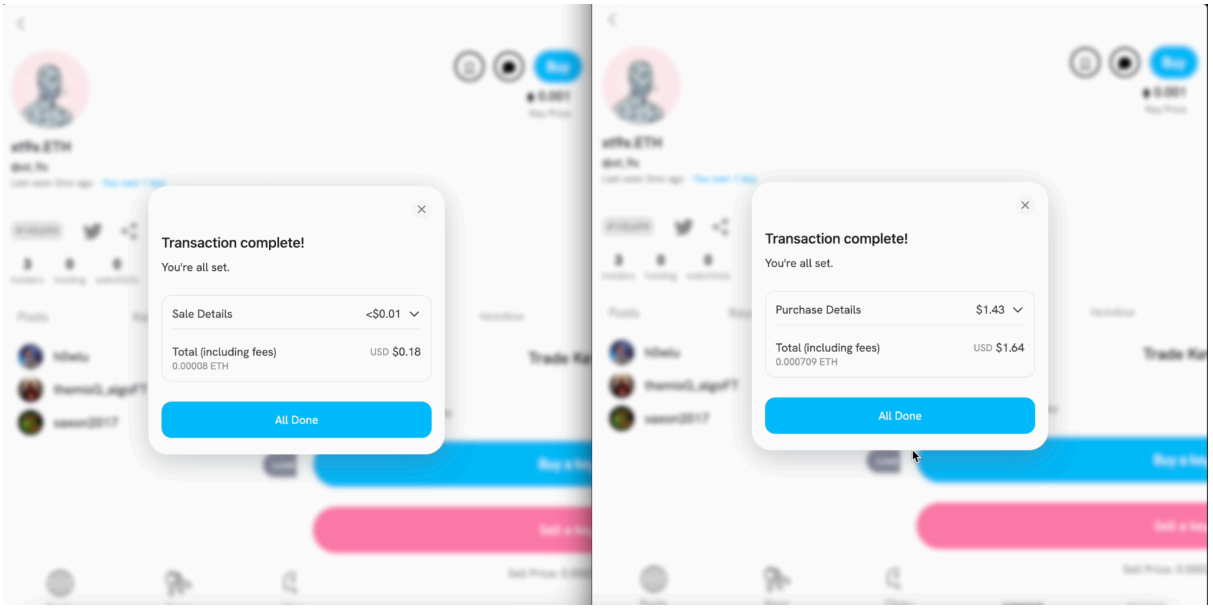
3. After the sale completes click ‘Buy’ on the right side window



4. After it loads you can see it shows the price that is valid for 3 keys, while in fact right now we have only 2

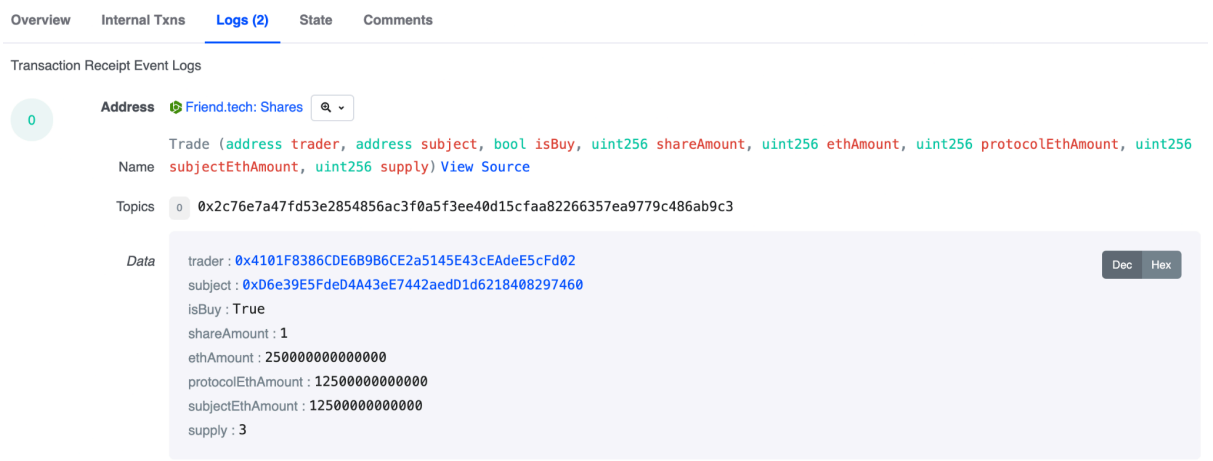


5. So we click ‘Complete Purchase’ and the transaction goes through costing us more then we should have paid



6. Excessive ETH will get locked in the smart contract - below is the transaction in question:

https://basescan.org/tx/0x632324c5f180408f2d6413e500592e9e587b758ea7d74b85b5d6175ccc903ce5



Link to a PoC video demonstrating how to reproduce the issue:
https://overflow.pl/various/ft_poc_video_ux_desync.mov

4. Recommendations

- Application UX
  - On each buy and sell clicks within the app, state of the blockchain should be synchronized and current prices updated
- Smart contract
  - Implement a functionality to refund ETH to users in case they sent more than expected

- Implement a withdrawal function so the funds are not stuck there forever, will be also useful in case someone sends ETH to the contract directly by a mistake

5. Timeline

14/12/2023 - vulnerability report sent to friend.tech bug bounty program
20/12/2023 - due to lack of response, a second attempt to establish a contact was made
20/12/2023 - got a response - "All transaction inputs and outputs match the user's explicit specifications. Report is a UX recommendation which is out of the scope of bounty rewards."
20/12/2023 - a follow up email was sent disputing the decision
05/01/2024 - due to no response a follow up email was sent, no response
01/02/2024 - the issue has been made public

6. Authors

@ELaszlo_
@h0wlu