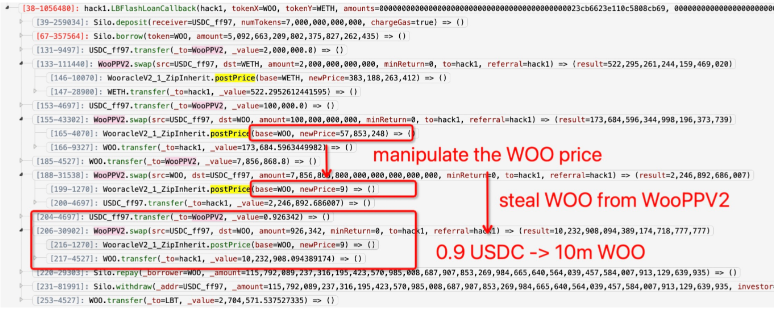
Making another sense of Trust Experience (TX)

Blockchain networks are designed, and supposed to be, trustless. This means that any third-party can independently verify and execute the proofs and calculations executed by other parties, without having to trust them in the first place. We can also see it as if we can *trust everyone in the network* because no one knows one another, and every one just work for themselves.

The entire network works as a very big justice court in which all the judges work together to take independent decisions that have very little ambiguity or not at all. Citing Josh Stark, it's the strongest type of hardness that we know at this point.

Hardness is the actual application of rules. Blockchain networks, and even more smart contracts-based ones, auto-enforce rules written in snippets of code, and there's no way to proactively fix some error, except for 'opening' a new network (a hard fork). When a new application is released on a blockhain network, the smart contracts it's composed of act as independent police for its users.

Just like police in real life, if some rules are written poorly and someone takes advantage of the way they're written, they can steal goods (read: money) and get away with it (mostly). Many of these thefts happened over the last few years, one of those merely a few days ago.



```
[38-1056480]: hack1.LBFlashLoanCallback(hack1, tokenX=WOO, tokenY=WETH, amounts=000000000000000000000000000000000000000000023cb6623e110c5808cb69, 0000000000000000000000
  [39-259034]: Silo.deposit(receiver=USDC_ff97, numTokens=7,000,000,000,000, chargeGas=true) => ()
  [67-357564]: Silo.borrow(token=WOO, amount=5,092,663,209,802,375,827,262,435) => ()
  [131-9497]: USDC_ff97.transfer(_to=WooPPV2, _value=2,000,000.0) => ()
  [133-111440]: WooPPV2.swap(src=USDC_ff97, dst=WETH, amount=2,000,000,000,000, minReturn=0, to=hack1, referral=hack1) => (result=522,295,261,244,159,469,020)
    [146-10070]: WooracleV2_1_ZipInherit.postPrice(base=WETH, newPrice=383,188,263,412) => ()
    [147-28900]: WETH.transfer(_to=hack1, _value=522.2952612441595) => ()
  [153-4697]: USDC_ff97.transfer(_to=WooPPV2, _value=100,000.0) => ()
  [155-43302]: WooPPV2.swap(src=USDC_ff97, dst=WOO, amount=100,000,000,000, minReturn=0, to=hack1, referral=hack1) => (result=173,684,596,344,998,196,373,739)
    [165-4070]: WooracleV2_1_ZipInherit.postPrice(base=WOO, newPrice=57,853,248) => ()
    [166-9327]: WOO.transfer(_to=hack1, _value=173,684.5963449982) => ()
  [185-4527]: WOO.transfer(_to=WooPPV2, _value=7,856,868.8) => ()
  [188-31538]: WooPPV2.swap(src=WOO, dst=USDC_ff97, amount=7,856,868,800,000,000,000,000,000, minReturn=0, to=hack1, referral=hack1) => (result=2,246,892,686,007)
    [199-1270]: WooracleV2_1_ZipInherit.postPrice(base=WOO, newPrice=9) => ()
    [200-4697]: USDC_ff97.transfer(_to=hack1, _value=2,246,892.686007) => ()
  [204-4697]: USDC_ff97.transfer(_to=WooPPV2, _value=0.926342) => ()
  [206-30902]: WooPPV2.swap(src=USDC_ff97, dst=WOO, amount=926,342, minReturn=0, to=hack1, referral=hack1) => (result=10,232,908,094,389,174,718,777,777)
    [216-1270]: WooracleV2_1_ZipInherit.postPrice(base=WOO, newPrice=9) => ()
    [217-4527]: WOO.transfer(_to=hack1, _value=10,232,908.094389174) => ()
  [220-29303]: Silo.repay(_borrower=WOO, _amount=115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,935) => ()
  [231-81991]: Silo.withdraw(_addr=USDC_ff97, _amount=115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,935, investor
  [253-4527]: WOO.transfer(_to=LBT, _value=2,704,571.537527335) => ()
```

If the rules in a smart contract are written poorly, a thief can exploit them and likely get away with it

But unlike police in real life, there's really no way to fix what has gone wrong and just eliminate the past. Sure, the contract deployer can release a new contract and link the new one on their website, but the history can't be erased, and in many cases the money can't be recovered easily.

So, blockchain networks are trustless systems, and when smart contracts are involved, rules are auto-enforced. But contracts are written by people, and people have biases and make mistakes, just like when they train AI systems.

So how can we make sure to remove, as much as possible, any trust assumption in systems that are supposed to be completely trustless?

Turns out, we can't.

We can't for many reasons:

1. **Contracts will always be developed by humans**, and even if we delegate smart contract development to AI completely, but even then some AI, or all of them, can have biases prompted by their creators. And even if it's AGI, we're not sure it can act in our best interest (it could think that trustless systems are not good for whatever reason we can't yet comprehend)
2. **Trustlesness, lilke security, is not a definitive statement, but a**

spectrum. And like security, it has compromises. The less trust you need to put into the system, the more trust you need to put in yourself (see custodial wallets, for example, and recovery accounts, that have less trustlessness because you need to trust that the accounts you allow the recovery of your account won't collude against you)

3. **Chicken and egg problem for the 'veil of abstraction' thesis**: If we abstract the 'blockchain' away from the users, it will most likely lead to more centralized (thus trusted) intermediate systems, but if we don't do that, it will most likelty lead to crypto not reaching mainstream ever, or only when the global population is versed enough in computing (still many decades, I suppose)

These are just three reasons that come to mind after a few minutes of thinking it through, but there could be (are for sure!) so many others.

What we can do is making sense of what trust really is at all the different levels. I called it Level of Trust (LoT), but Trust Experience (TX) sounds way better and it really gives a better idea of what that's really about. I think TX encapsulates LoT, but they're not exactly the same thing.

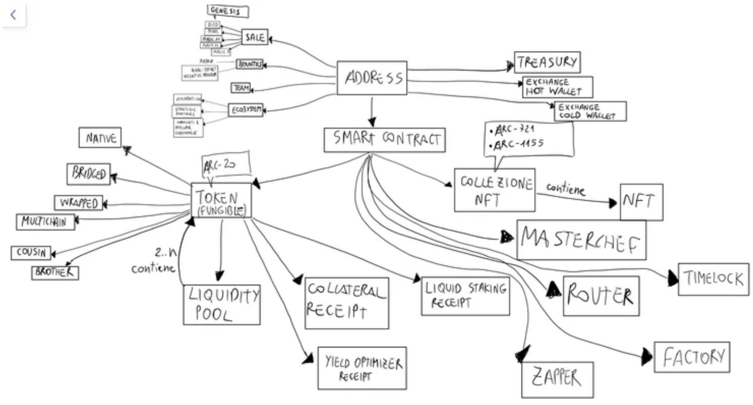
And this post is named 'Making **more** sense of Trust Experience (TX)' to *expand* on the ideas described by Josh Stark in its post that outlines what the Trust Experience in the first place.

Quoting:

> TX is the set of experiences that shape and inform our expectations about how a blockchain (or other system) will behave in the future. It is the sum of all external inputs which leads us to believe that it will function in a certain way in the future - to trust it, or distrust it.
>
> If User Experience (UX) is about how a person interacts with and experiences a technology, Trust Experience (TX) is about how a person interacts with and experiences *forming expectations about the future behaviour* of that technology.

While Josh seems to only apply TX to L1s, I think TX is way broader: it can be applied to all kinds of object on the blockchain, whether it be a token, NFT or account.

I described some of these ideas back in 2022 when talking about Avalanche subnets during a workshop at the first Avalanche Summit.



A very first draft of a schema to define what the types of addresses were and how they 'worked' with one another - the first step in understanding how much trust to put in different types of addresses and why

Describing the Trust Level applied to tokens:

> […] a Level of Trust (or Trust Level) represents the number of software and/or entities involved in the life cycle of a token. This involvement can influence the trust, of course, but it can also influence its name (both symbol and contract name), price and liquidity on the market. Tokens with the same name but different Trust Levels are actually different tokens. The Trust Level we imagined ranks token from 0 to *n*, where *n* is a finite number that we haven't yet discovered. The highest level of trust is 0 (because you need to have 0 or little trust), while the lowest is *n.*

For example, a native token on a blockchain network has the lower (base) Level of Trust, and we can easily say that it's Trust Experience (TX) is at the highest. LoT and TX and inversely correlated: the lowest the LoT, the highest the TX. We can also say that TX is the result of 'abstracting away LoT'. Today, we assume that if we have the highest LoT, meaning that an object or system is completely centralized, we will have the best User Experience (UX), but we'll also have a very minimal Trust Experience (TX), because we're delegating all our trust to a third-party.

The goal is to find a way to inversely align LoT with TX, and to directly align TX with UX, so that for a low Level of Trust, there's a high Trust Experience and a high User Experience, meaning:

> A user needs to have the lowest trust assumptions but also have a user experience as if the trustless system is a trusted one.

I believe there are two steps in ensuring this takes place: **building a trust score system** first, and then **remove or heavily reduce the objects / entities / systems with the highest LoT**, so that the remaining ones have lowest or sufficiently low LoT to make for a high TX. When working on this, the UX will improve greatly as a natural consequence.

I have some ideas about the trust score system, a sort of L2Beat meets Coinmarketcap, but it's still a long way from making some sense. But I believe this is the way to go forward. As soon as I'll do more research, I'll post more about it.

Date: 2024-03-11
Words: 1274
Time to read: 6 mins

Newer Older

15th March 2024
On a sustainable model for bloc…

8th March 2024
About L3s and the Superchain, …