With Vitalik Buterin et al.'s recent paper on finding Web3's Soul [1] and my work on "Account-bound tokens" [2], I've come across an authenticity issue that plagues the Ethereum community.

[

2744×1520 551 KB

](https://ethresear.ch/uploads/default/original/2X/2/29f9bacd60fddc276fe5022ab93cfc56aaace544.jpeg)

Its most prominent symptom has been the infamous "sleepminting" attacks that took place just a year ago [3] when someone manipulated the parameters of EIP-721's event Transfer(from, to, id)

such that platforms like Etherscan and rarible showed an arbitrary NFT's minter as "beeple," when, in fact, an attacker was behind these transactions.

[

2714×1496 355 KB

](https://ethresear.ch/uploads/default/original/2X/1/148e53380dd3f5fd20b7bb903a2909ce94208075.png)

Identifying the "sleepminting attack" problem

The outcome of a blog post I then wrote to identify the attack's vector was that any platform displaying NFTs shouldn't trust a contract implementation's "honest" usage of event Transfer

. After all, it isn't and can't be mandated to "just emit a single event Transfer

" upon transfer that perfectly reflects the actual transfer. Or rather, I then started understanding that the problem isn't one of narrowly defining standards: But rather one of social scalability [4].

Namely, despite the possibility of every human being theoretically capable of inspecting an EIP-721-compliant contract's use of event Transfer

, that assumption not being deliberately socially scalable.

Social Scalability, hence, refers to what Szabo outlines as being conscious of the human mind's limitations and designing systems to limit the risk and harm of users while still possibly scaling to large populations. A different place, I've come across this principle too is in Steve Krug's book "Don't make me think."

So really, I think it is too much to ask users to review every EIP-721-compliant contract they want to interact with. In fact, it is a good moment to be grateful that the Ethereum community was able to find consensus on using EIP-20 and EIP-721's interfaces.

Still, I see there being issues with event Transfer

.

Improving Authenticity

In a perfect world, some event signatures would be known to developers as authentic. E.g., as Solidity developers, we already know that msg.sender

within Solidity is immutable and non-controllable by an implementer. So, auditing individual contracts by checking if event Transfer

sets from=msg.sender

would most all doubts about event Transfer

's signature being non-authentic while being (at least to some highly technically sophisticated users) non-overwhelming on a mental capacity-level.

Indexing today's EIP-721 contracts, event Transfer

's 'from

may be used to understand which events originated from an address. However, for authenticity, it MUST be cross-checked with the actual transaction's from

field to ensure authenticity as these two fields can diverge in case of malicious implementations. Here's "fake-beeple"'s

minting transaction on Etherscan:

[

2714×1496 355 KB

](https://ethresear.ch/uploads/default/original/2X/1/148e53380dd3f5fd20b7bb903a2909ce94208075.png)

And to quote Szabo on Social Scalability, however:

The more an institution depends on local laws, customs, or language, the less socially scalable it is.

Which is why I want to identify educating users to explicitly check the outer transaction's from

field constitutes a "local custom" within the Ethereum community and, hence, lessens its social scalability.

Proposal

Rather, I think machines should do the job of authenticating an implementer's correct use of, e.g., an event Transfer(from, to, id)

. E.g., an ideal system would include an identification component similar to EIP-165 in a contract's interface. Similarly, an interface mandate as EIP-721 would then enforce from=msg.sender

in event Transfer(from, to, id)

, and it would be identifiable through checking, e.g., hashes with a supportsInterface

method.

So imagine we introduced a new type of event. Actually, we may need to include event signatures in EIP-165 first. Then secondly, already in an interface's definition, an implementer shall be allowed to define a transfer event as event Transfer(address msg.sender, address to, uint256 id)

, where then the EIP-165's hashing ensures that its identifier differs from, e.g. event Transfer(address from, address to, uint256 id)

(where from

isn't guaranteed to be authentic.

Finally, when indexing such contracts, an indexer would then be able to socially-scalable determine EIP-721 contracts that implement from=msg.sender

.

Further notes

- I've spoken about this to Chris from the Solidity team, and they told me that the Solidity compiler could not enforce such guarantees. I was suggested I talk to the EVM team. I've not been in touch yet. But I'm happy to start discussing!

References

- 1: Decentralized Society: Finding Web3's Soul by E. Glen Weyl, Puja Ohlhaver, Vitalik Buterin :: SSRN
- 2: Submit EIP-4973 - Account-bound tokens by TimDaub · Pull Request #4973 · ethereum/EIPs · GitHub
- 3: What Is Sleepminting And Will It Ruin NFT Provenance?
- 4: Unenumerated: Money, blockchains, and social scalability
- 5: EIPs/eip-165.md at 1dc1929379b46b8e5a787d2a19bbee2a71a850b0 · ethereum/EIPs · GitHub