Here is my draft proposal for simple phishing protection to protect DApp users against some of the common phishing techniques malicious actors are using today. This is by no means a complete proposal, but just a request for comments opening.

Goals

Wallets and Dapps can communicate about the transfer of value in a way that makes it easier for users to reason about it and detect phishing.

The goal is to give protection against

- Email phishing
- Messenger phishing (Discord, Telegram)
- Allow wallets and users to understand the counterparty of value transfers, both direct (explicit) and indirect (implicated) like
- Approvals on wrong contracts
- Transferring value to a wrong address
- Trading tokens in a complex DEX composited of multiple pool smart contracts
- Approvals on wrong contracts
- Transferring value to a wrong address
- Trading tokens in a complex DEX composited of multiple pool smart contracts

The goals further include

- To be decentralised (no central party whitelisting or registry, outside maybe ENS name indications)
- Have a simple spec any wallet can easily implement
- Do not try to cover all possible attack vectors, but keep the spec simple enough that it is realistic to get adoption from Dapps and wallets (e.g. do not die in the same pit as ERC-777)

Use cases

- The user knows they are interacting with the smart contracts from the same source that they have interacted before (curve.fi

, uniswap.eth

, opensea.eth

) and are warned if they are about to interact with somebody new or somebody who pretends to be one of these parties.

- Indicate the user that they are about to approve()

or do a similar implicit transfer-of-value to a certain party, through one or multiple smart contracts.

- Indicate to the user they are about to transfer value to a certain centralised exchange. The exchange [hot wallet forwarded contract] can implement the interface, thus also protect against phishing where someone pretends to be Binance, Coinbase or similar.

Overview

Currently, wallets like MetaMask warn you if you are transferring ETH or tokens to a new address and ask you to confirm it when you add it to an address book.

[

image

928×636 97.3 KB

](https://ethresear.ch/uploads/default/original/2X/f/ff180bd6e9d08bc18579702bfa9a7986518e9f3f.png)

The purpose of Simple Phishing Protection proposal is to increase the scope of this functionality, so that we capture more transfer-of-value scenarios that might put the users to risk.

- Approving value or other implicit transfer of value where a user might not understand they are risking vhe value
- Dealing with new organisation (smart contracts associated with the same group, like Uniswap pair contracts) vs. a new address

Simple Phishing Protection does not try to protect against malicious new parties. It focuses to protects you against threats where someone pretends they are a party you have interacted before. This is because most of the scams and phishers ride on the name of a legitimate business.

Method

- Smart contracts self-describe that they offer Simple Phishing Protection
- When a user interacts with a smart contract, or with its allowances and delegations, for the first time wallet warns the user that they are interacting a party that has not been seen before
- Smart contracts refer to on-chain identity, or simply an address, which presents a the party the user is interacting with. This address should resolve to ENS name or other human readable label that phishers cannot copy.

An example of how this could be implemented. For the feedback, I suggest you propose implementation methods that improve this.

- Smart contracts indicate Simple Phishing Protection viaEIP-1820 interface registration

The interface looks like this

interface ISimplePhishingProtection {

```
struct Purpose {
    address party;   // Address of a counterparty e.g. uniswap.eth
    address asset; // Address of an ERC-20 token, NFT, can be also left zero if the scope of allowance cannot be described
    uint256 amount;  // Amount of the assets potentially in risk. Set max unit to unlimited
    string reason;   // Human readable reason why this allowance etc. is required
}

// Decodes a data payload of a transaction that wallet is about to send to the smart contract.
// The first 4 bytes are function selector like in Solidity and the
// implementor will detect what this transaction is about. E.g. it can differentiate
// between different function calls and their arguments to the target contract.
//
// target is the address of the smart contract the user is about to interact
// tx is the data input the user is about to send in
// returns the list of addresses of parties the user is about to interact with
function getCounterparties(address target, bytes tx) public returns(Purpose[]);

}
```

Wallet phishing check workflow

For wallets

1. They check if the user is about to send a transaction to a smart contract with Simple Phishing Protection
2. The wallet checks the on-chain identity of Simple Phishing Protection implementation (TBD
3. prevent phishers to register implementers in the name of someone else, check if the ENS is the best option here)
4. Then wallet calls the ISimplePhishingProtection

implementer to get the list of the addresses and purposes the user is about to interact with. Then the wallet checks if it recognises these addresses and if the user has interacted with them before. If the user has not interacted with before (fake OpenSea, fake Uniswap) the wallet gives something along with a red warning and 10-second countdown before letting the user to press ok.

For example in the case of Uniswap approval this would be the following:

- You are allowing uniswap.eth

to control 10 ETH

for liquidity provision

.

For the phising it could look like this

- You are allowing uniswåp.eth

to control 10 ETH

for liquidity provision

. YOU HAVE NOT INTERACTED WITH UNISWÅP.ETH BEFORE

.

For example in the case of OpenSea this would be

- You are allowing opeansea.eth

to control all of your NFT collection

Deploying Simple Phishing Protection for smart contracts

Smart contract developers must start provide Simple Phishing Protection interface as a part of their main interaction contract (Uniswap router, Aave core).

1. The developers will opt in for the scheme
2. The deployment of a smart contract that supports the scheme is done by EIP-1820interface registration in the smart contract constructor (can be done elsewhere?)
3. After the registration, the deployer generates a proof that they are legitimate owner of uniswap.eth

or similar address (TBD how

)

Note that because how EIP-1820 is structured, only one Simple Phishing Protection implementation deployment is needed for multiple contracts. Thus, adopting Simple Phishing Protection can be done as one go per protocol.

Weaknesses of the scheme

Of course, the weaknesses of this, and all other simile anti-phishing schemes is that phishers won't implement or support this scheme. The only way to overcome this problem is with

1. pushing implementation through advocacy
2. user education
3. time

If we are in the situation that 50% of wallets (by usage) and 50% of new smart contract protocols (by usage) are covered by Simple Phishing Protection, then the users will become more careful when interacting with something that is not covered by the scheme.

Other indicators

Based on my earlier anti-phishing and anti-scam work, there are decentralised and hard-to-fake indicators that can be further introduced to protect users when they are interacting with smart contract systems they have not interacted before (e.g. interacting with OpenSea for the first time and understand that it is likely to be the real OpenSea)

- How old is the on-chain identity (scams have a limited shell life, because they end up on the wallet blacklists. Any large

scam is taken down within a week, so anything longer than a week is safer)

- How much activity other users have had with the on-chain identity. Real Uniswap, real Curve or any popular service the phishers mimic has 10,000,000x activity compared to a scam address.