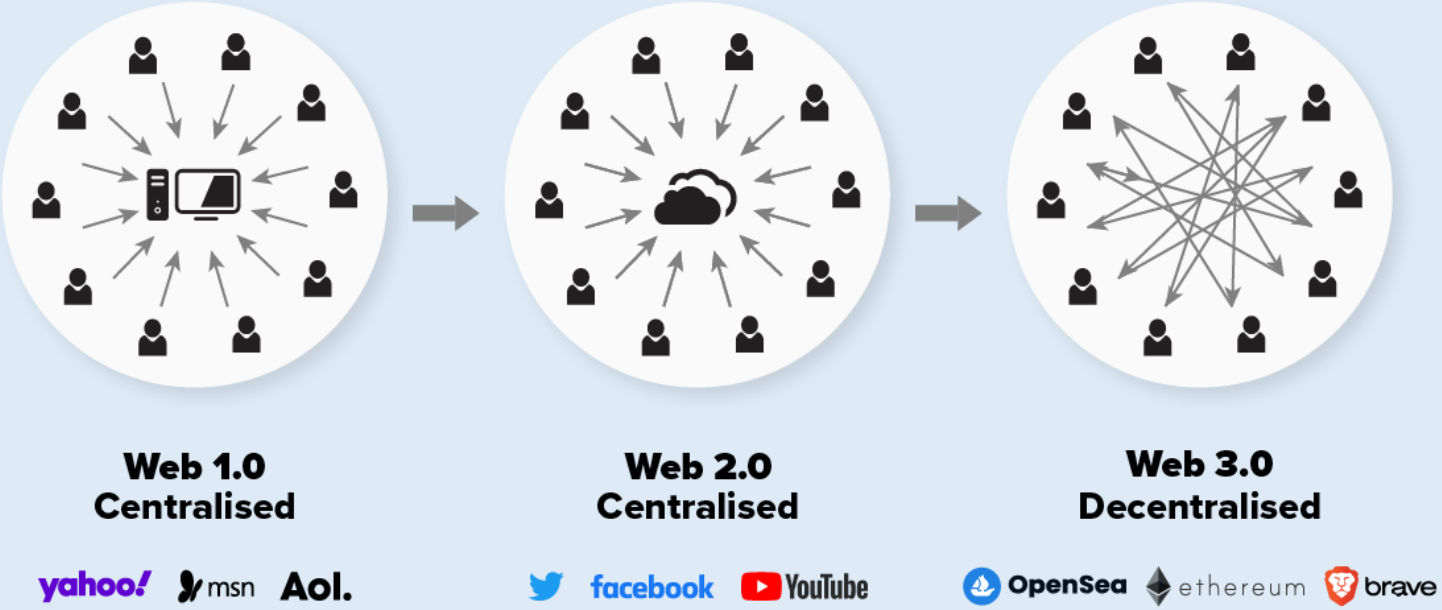
CyRadar

Blockchain Security

Tùng Lê - CyRadar

Web3 Security ?

Web3.0



Article Talk Read Edit View history Search Wikipedia

Web3

From Wikipedia, the free encyclopedia

This article is about the concept of a World Wide Web based on public blockchains. For the concept based around machine-readability, also called Web 3.0, see Semantic Web.

Web3 (also known as **Web 3.0**[1][2][3]) is an idea for a new iteration of the World Wide Web which incorporates concepts such as decentralization, blockchain technologies, and token-based economics.[4] Some technologists and journalists have contrasted it with Web 2.0, wherein they say data and content are centralized in a small group of companies sometimes referred to as "Big Tech".[5] The term "Web3" was coined in 2014 by Ethereum co-founder Gavin Wood, and the idea gained interest in 2021 from cryptocurrency enthusiasts, large technology companies, and venture capital firms.[5][6]

Some commentators argue that Web3 will provide increased data security, scalability, and privacy for users and combat the influence of large technology companies.[7] Others have raised concerns about a decentralized web, citing the potential for low moderation and the proliferation of harmful content,[8] the centralization of wealth to a small group of investors and individuals,[9] or a loss of privacy due to more expansive data collection.[10] Others, such as Elon Musk and Jack Dorsey, have argued that Web3 only serves as a buzzword or marketing term.[11][12][13]

Blockchain Security - Challenges

Top 5 biggest hacks

https://www.cnet.com › Money › Crypto

Axie Infinity's Ronin Network Loses Over $600M in One of the ...

Mar 31, 2022 — The **attack** on the popular NFT video game occurred last week, and the company says it's "committed" to reimbursing players.

TECH | CYBERSECURITY | CRYPTOCURRENCY

Poly Network hacker gave back more than $600 million in stolen crypto

https://fortune.com › 2018/01/31 › coinc... ▾ Dịch trang này

Coincheck Hack: How to Steal $500 Million in Cryptocurrency

31 thg 1, 2018 — Early Friday morning in Tokyo, **hackers** broke into a cryptocurrency exchange called **Coincheck** Inc. and made off with nearly $500 million in ...

The Inside Story of Mt. Gox, Bitcoin's $460 Million Disaster

Tokyo-based bitcoin exchange Mt. Gox filed for bankruptcy last week, saying hackers had stolen the equivalent of $460 million from its online coffers. The news rocked the bitcoin world, and it could even bring down the much-hyped digital currency.

5. **Wormhole – $326 million stolen:** In the first major crypto heist of 2022, Wormhole's crypto platform was exploited to the tune of $326 million. The platform acts as a communication bridge between Solana (an ethereum rival that has recently gained traction) and other decentralized finance networks. On February 2, 2022, hackers were able to exploit a vulnerability, causing Wormhole to shut down its platform while it investigated. It later reported that 120k wrapped Ethereum (wETH) had been stolen.

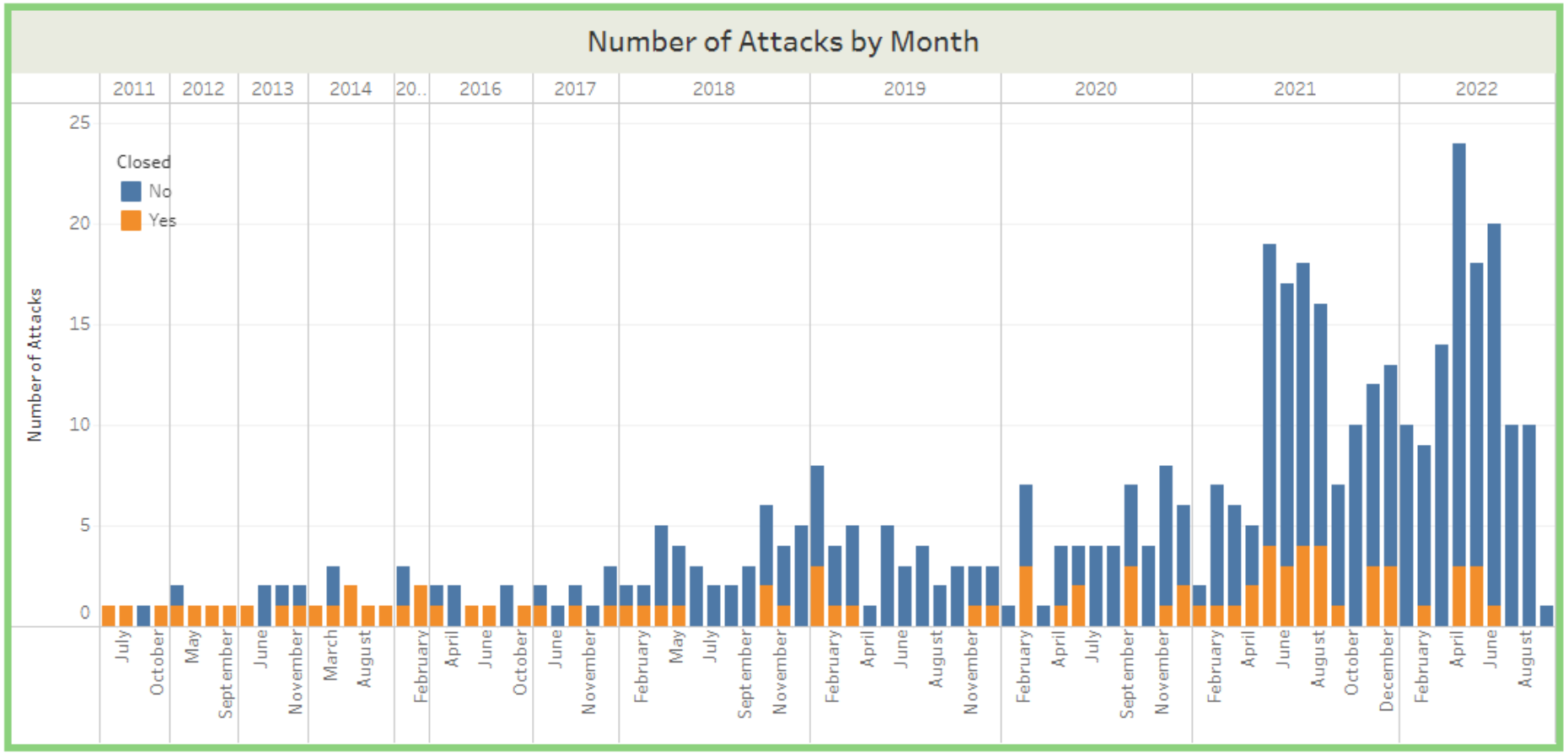
CyRadar

Actual Amount Stolen (USD)

7,440,975,202

Amount Stolen in Today's Value (USD)

45,406,386,654



https://www.comparitech.com/crypto/biggest-cryptocurrency-heists/

Blockchain Security - Challenges

CyRadar

Rise of Attack numbers

Billions of $ lost

Both blockchain and
non-BC Bugs

Basic bugs
(lack of experience in battle
vs hackers

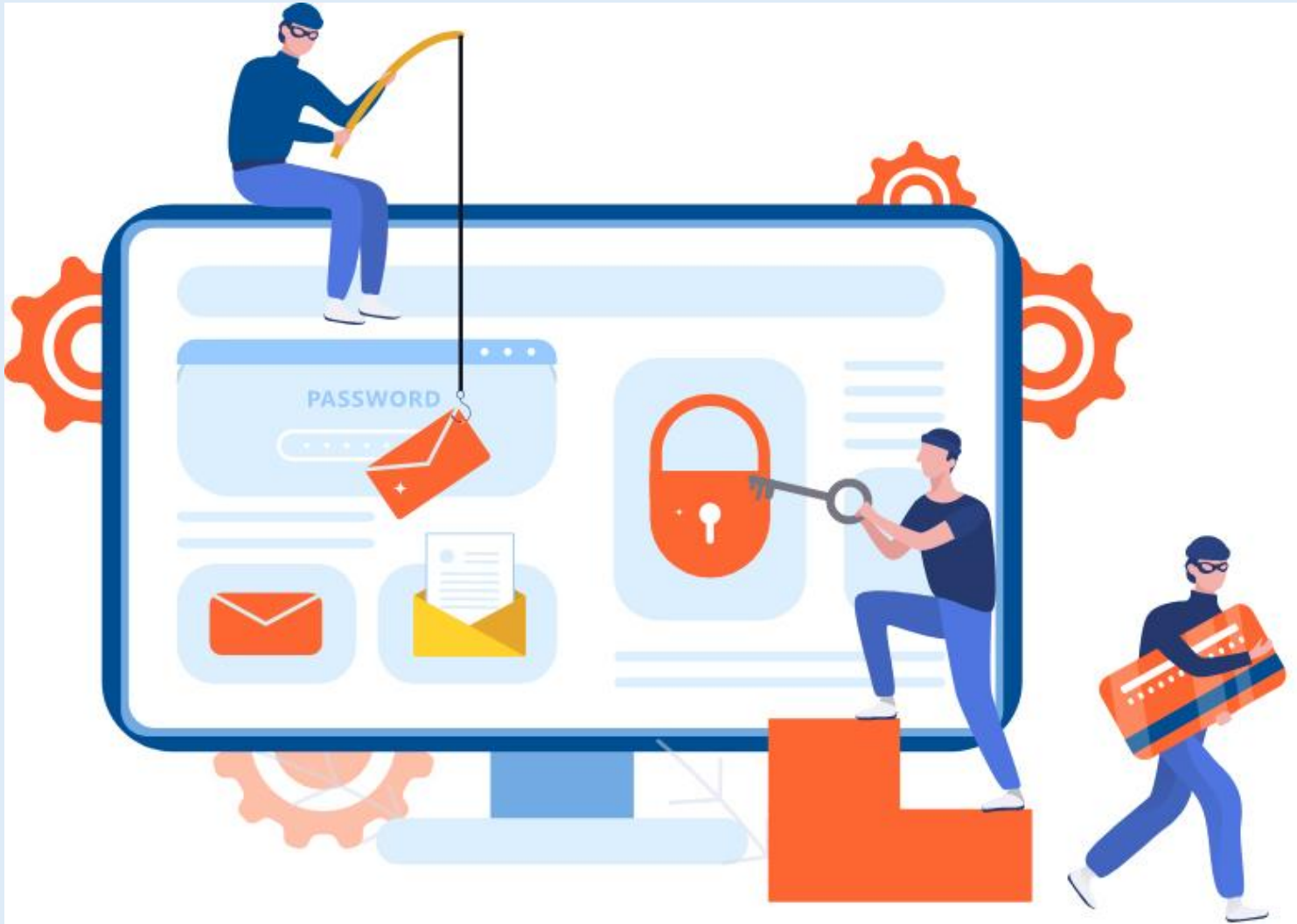
Audit process need improve
(Audit is not enough)

Vulnerabilities in blockchain

- Regarding Traditional Cyber Security
- Network-Level Security
- Smart Contract Vulnerabilities

Regarding Traditional Cyber Security

- User Security
  - Failure to Protect Private Keys
  - Vulnerability to Malware
- Node Security
  - Insecure API Connections
- Network Security
  - Flawed Network Design
  - Poor Network Security

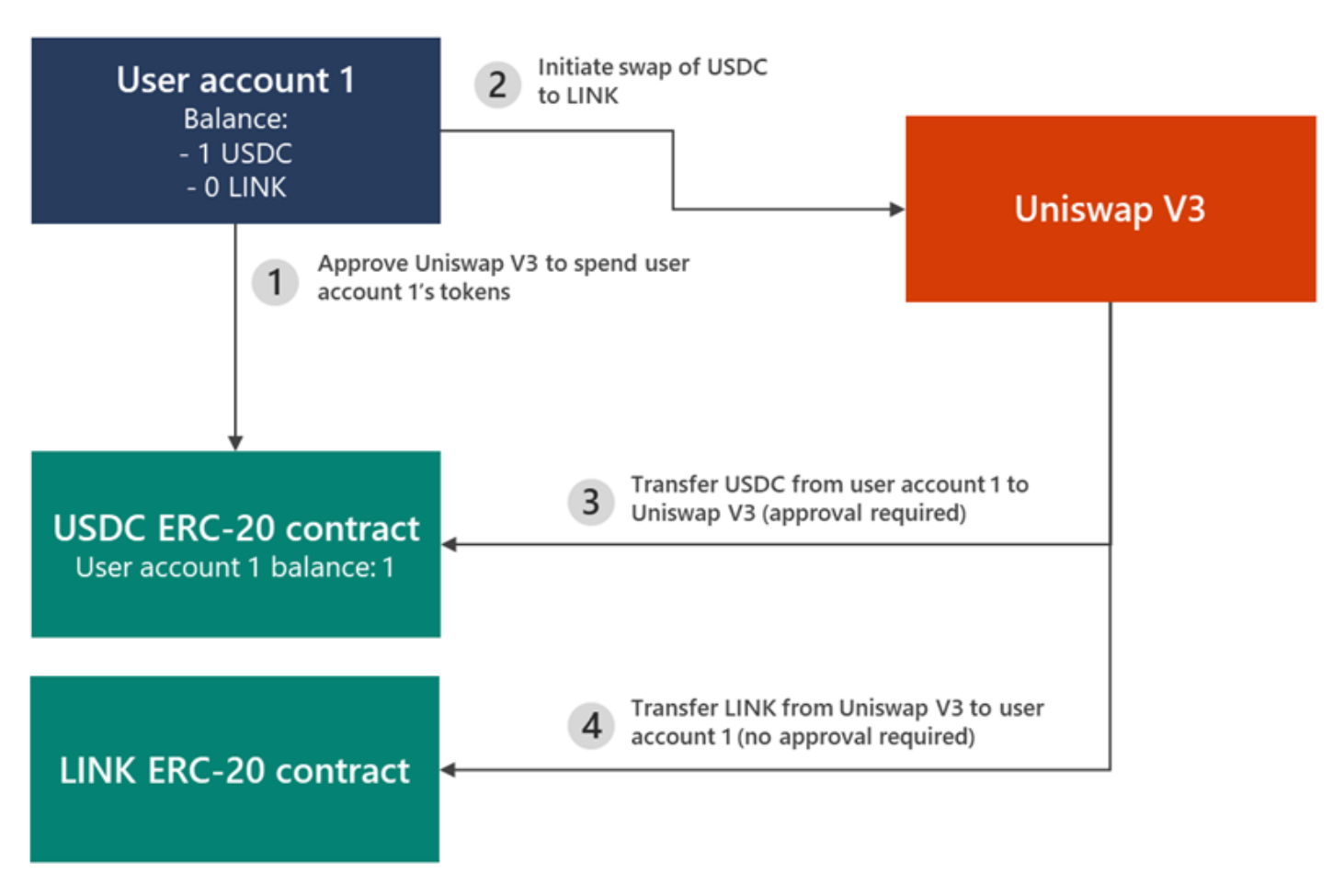


Regarding Traditional Cyber Security – Case Study

- Ice Phishing

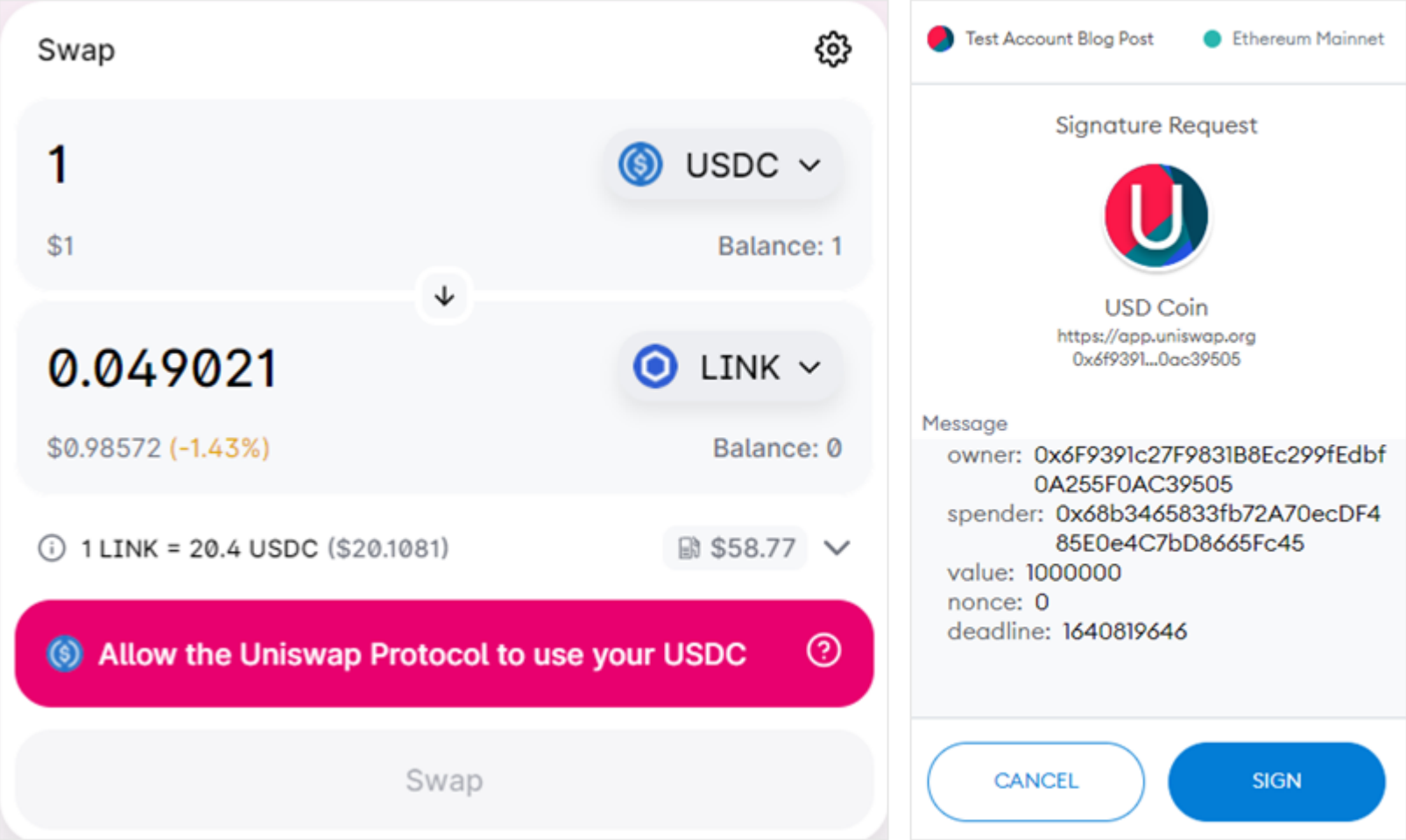
(Internal Communication Emulation Phishing)

- https://www.microsoft.com/security/blog/2022/02/16/ice-phishing-on-the-blockchain/



“Ice Phishing”

- https://www.microsoft.com/security/blog/2022/02/16/ice-phishing-on-the-blockchain/



“Ice Phishing” - Case Study - Badger DAO

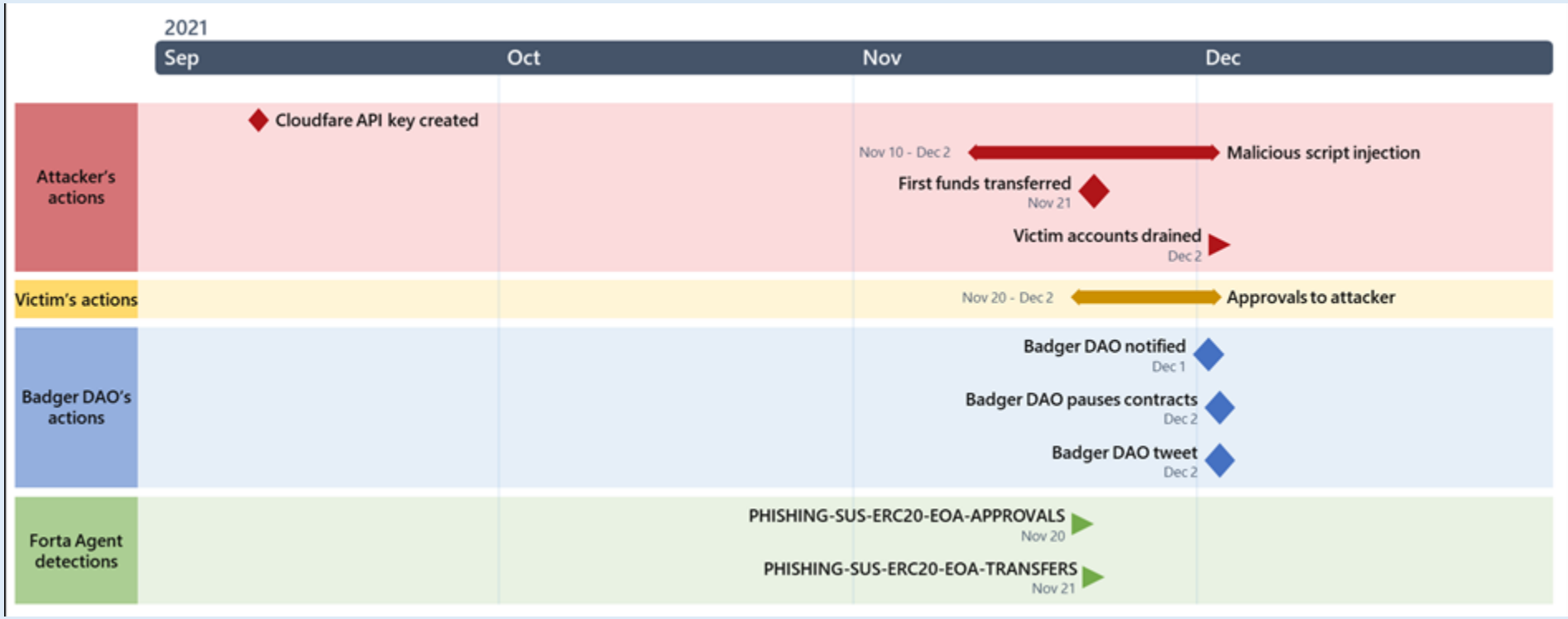
Badger

After extensive internal security reviews, upgrades, and the implementation of new safeguards with the support of Mandiant, Badger smart contracts have been safely reactivated. Read more about our security upgrades.

Web Infrastructure Security

Since the recent exploit originated through unauthorized access to our Content Delivery Network (CDN) infrastructure, Badger is relaunching the website with additional security safeguards suggested by Mandiant.

“Ice Phishing” - Case Study - Badger DAO

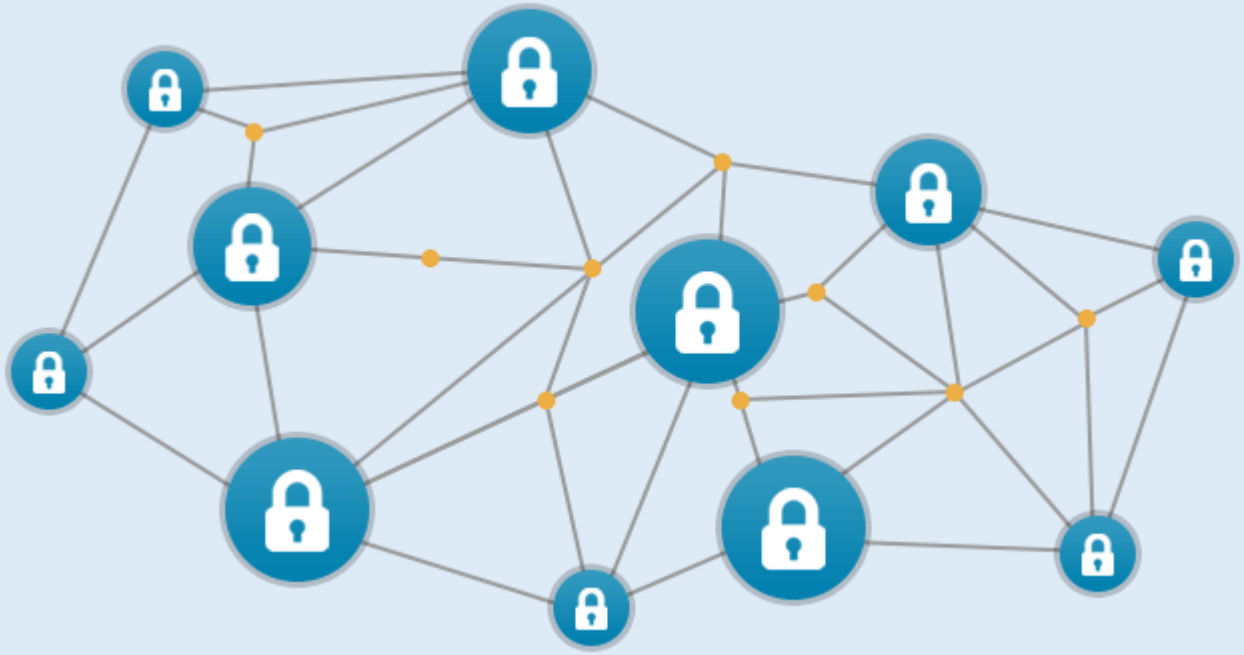


> Here is the current whereabouts as well as the total loss: $120.3M (with ~2.1k BTC + 151 ETH) @BadgerDAO pic.twitter.com/fJ4hJcMWTq
>
> — PeckShield Inc. (@peckshield) December 2, 2021

Insecure API

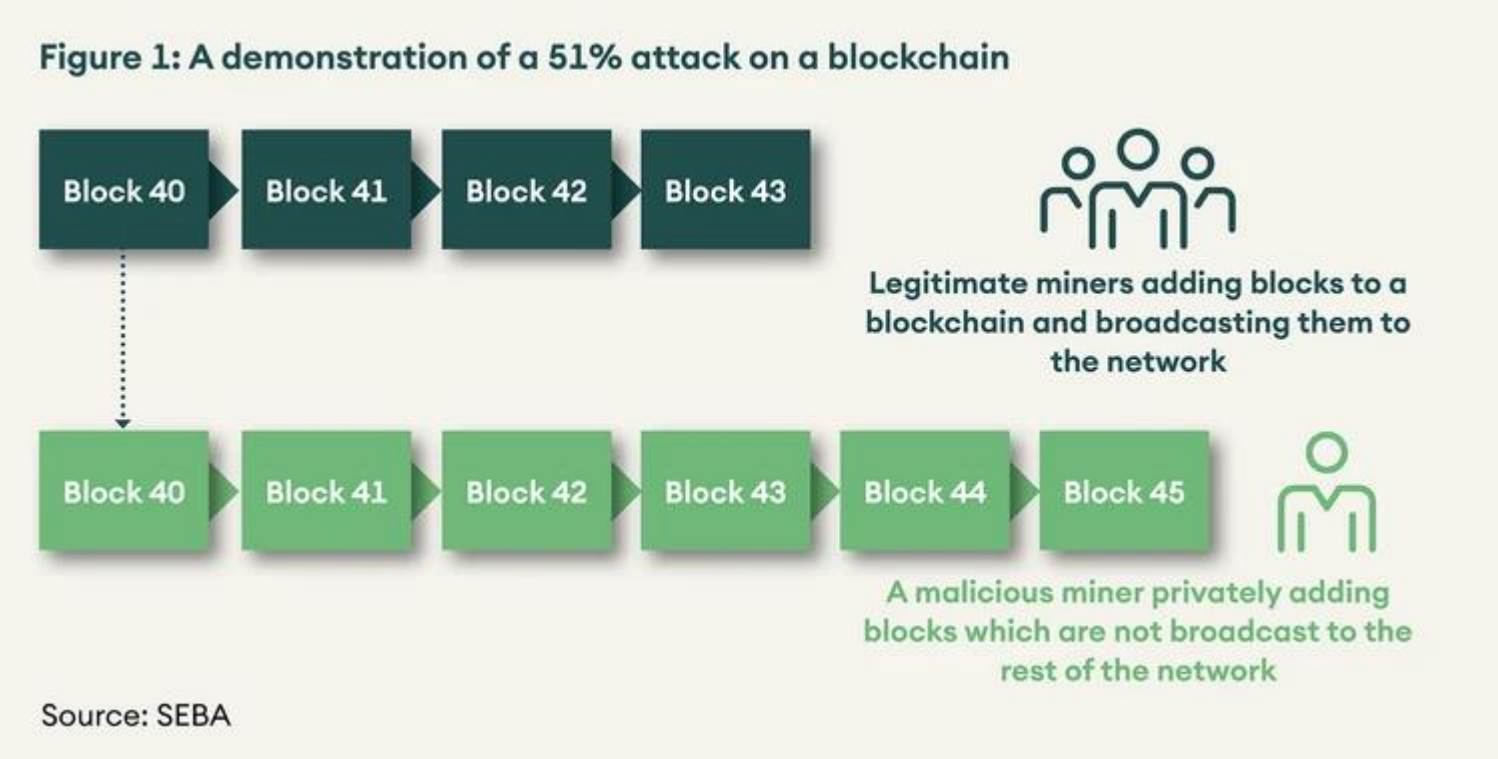
Network-Level Security

- 51% Attacks
- Denial of Service Attacks
- Eclipse Attacks
- Replay Attacks
- Routing Attacks
- Sybil Attacks



51% Attack – Case Study

- **51% Attack - Double Spends**



51% Attack – Case Study - Bitcoin Gold

NEWS
by **C. Edward Kelso**

May 24, 2018

23073

Bitcoin Gold Hacked for $18 Million

- Thu, 23 Jan 2020 18:01:32 - 14 blocks removed, 13 blocks added
  - 1,900 BTG double-spent (~$19,000).
  - 1,900 BTG originally sent to `GgmzUSgXrXpDxiY34bG6SxaDVi2rQ1zU8Q` in TXID 3a17157994502a749a1827883a670d822f8ee95dae94064631770faeec1e8443 was redirected to `GNH5cUEg5LZZP5HfLgaLvTE9ApKAf76aBf` in TXID 6e05e8253b2ce7f1acf6f0684898e13141c0e9b893e1a5e44d215d8ebe4d28b4.
  - The majority of the coins were sent from an output owned by the address `GK6HuN964f3XFScY5CPGg1oZ1gFRq52nf5` .
- Fri, 24 Jan 2020 00:24:08 - 15 blocks removed, 16 blocks added
  - ~5,267 BTG double-spent (~$53,000).
  - ~1,947 BTG originally sent to `Gg4YDMrMuqit6eJAYKaBxmK17zPFnpLt5w` , 1,850 BTG to `GfRdNzHJan8sfW9wxozAYhRPL9fFLD9A9m` in TXID 481d608591f4d6a7013ac1b879c2caf1e2c0a2bb30b5346b2c876deb43873b2b and 1,470 BTG to `GfWUNAdW3aEXfQWshApFLf2ZNtMV9MC6VQ` in TXID

The Github gist that explains the recent Bitcoin Gold 51% attack on Thursday and Friday.

NEWS
by **Jamie Redman**

Jan 26, 2020

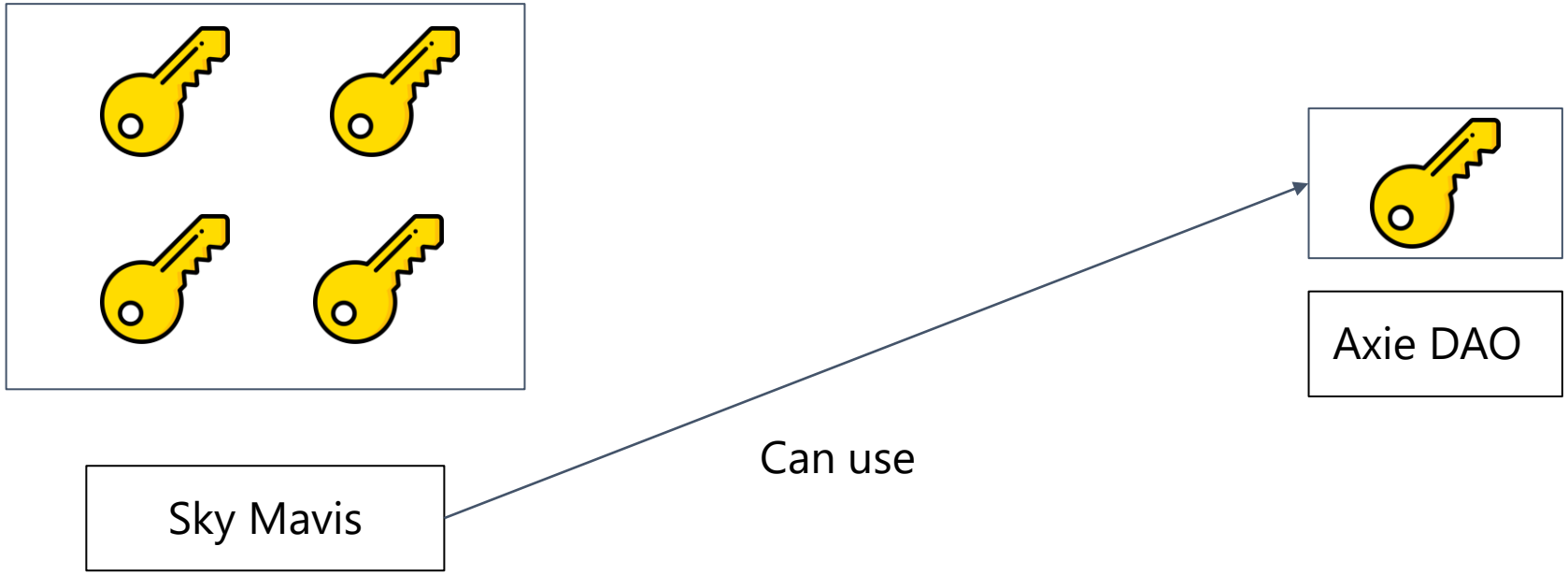
15999

Bitcoin Gold 51% Attacked - Network Loses $70,000 in Double Spends

The Bitcoin Gold (BTG) network suffered another set of 51% attacks on January 23-24, as roughly 29 blocks were removed in two deep blockchain reorganizations (reorgs). Reports indicate that more than 7,000 BTG was double spent ($70,000) in two days.

Axie Hack - What went wrong?

- Keys are **not independently protected**

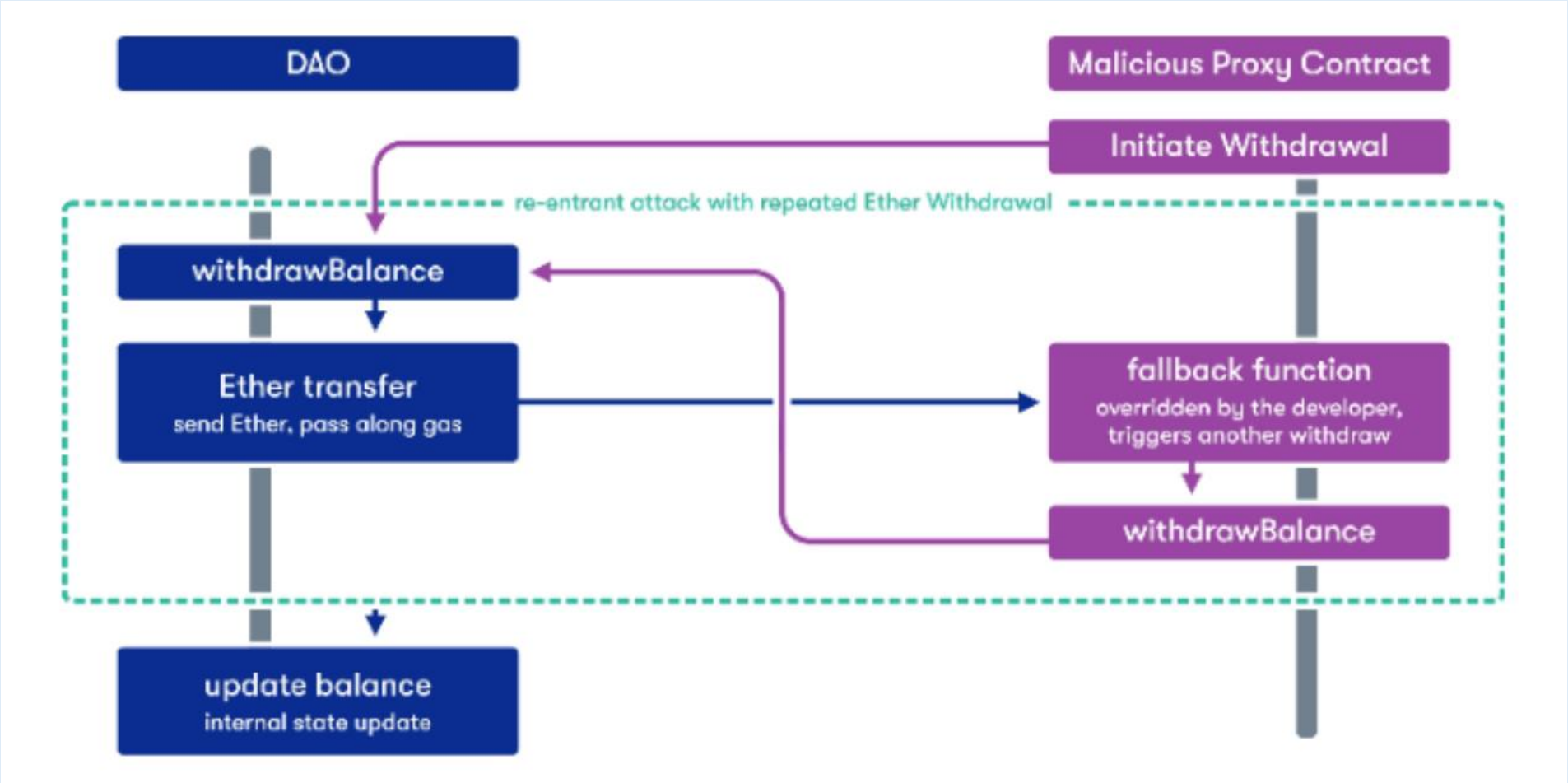


Smart Contract Vulnerabilities

- Reentrancy
- Access Control
- Arithmetic
- Unchecked Return Value
- Denial of Service
- Bad Randomness
- Race Conditions
- Timestamp Dependence
- Short Addresses
- Logic bugs

Smart Contract Vulnerabilities – Case Study

- Reentrancy



Reentrancy – Case Study - Lendf.me

total amount of loss caused by the attack for Lendf.Me was about **$24,696,616**, which includes the following cryptocurrencies:

> *WETH: 55159.02134,*
> *WBTC: 9.01152,*
> *CHAI: 77930.93433,*
> *HBTC: 320.27714,*
> *HUSD: 432162.90569,*
> *BUSD: 480787.88767,*
> *PAX: 587014.60367,*
> *TUSD: 459794.38763,*
> *USDC: 698916.40348,*
> *USDT: 7180525.08156,*
> *USDx: 510868.16067,*
> *imBTC: 291.3471*

Reentrancy – Case Study - Lendf.me

Transaction Details

Buy | Exchange | Derivatives | Crypto Credit

Feature Tip: Track historical data points of any address with the analytics module !

Overview | Internal Transactions | Event Logs (13) | State Changes | Comments

Transaction Hash:	0xae7d664bdfcc54220df4f18d339005c6faf6e62c9ca79c56387bc0389274363b
Status:	Success
Block:	9899738 1183 Block Confirmations
Timestamp:	4 hrs 31 mins ago (Apr-19-2020 12:58:55 AM +UTC)
From:	0xa9bf70a420d364e923c74448d9d817d3f2a77822
To:	Contract 0x538359785a8d5ab1a741a0ba94f26a800759d91d
Tokens Transferred: 3	From 0x538359785a8d5a... To Lendf.Me For 0.00021593 ($1.51) The Tokenize... (imBTC) From Lendf.Me To 0x538359785a8d5a... For 0.00043188 ($3.03) The Tokenize... (imBTC) From 0x538359785a8d5a... To Lendf.Me For 0.00000001 ($0.00) The Tokenize... (imBTC)
Transaction Action	Supply 0.00021593 imBTC To Lendf.Me Withdraw 0.00043188 imBTC From Lendf.Me Supply 0.00000001 imBTC To Lendf.Me
Value:	0 Ether ($0.00)
Transaction Fee:	0.0026283666 Ether ($0.49)

Click to see More

Private Note:	To access the Private Note feature, you must be Logged In

Reentrancy – Case Study - Lendf.me

```
function supply(address asset, uint amount) public returns (uint) {
    // Fail gracefully if asset is not approved or has insufficient balance
    err = checkTransferIn(asset, msg.sender, amou
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_TRANS
    }

    // We calculate the newSupplyIndex, user's su
    (err, localResults.newSupplyIndex) = calculat
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_NEW_S
    }

    (err, localResults.userSupplyCurrent) = calcu
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_ACCUM
    }

    (err, localResults.userSupplyUpdated) = add(l
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_NEW_T
    }

    // We calculate the protocol's totalSupply by
    (err, localResults.newTotalSupply) = addThenS
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_NEW_T
    }

    // We need to calculate what the updated cash
    localResults.currentCash = getCash(asset);

    (err, localResults.updatedCash) = add(localRe
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_NEW_T
    }

    // The utilization rate has changed! We calcu
    (rateCalculationResultCode, localResults.newS
    if (rateCalculationResultCode != 0) {
        return failOpaque(FailureInfo.SUPPLY_NEW_SUPPLY_RATE_CALCULATION_FAILED, rateCalculationResultCode);
    }

    // We calculate the newBorrowIndex (we already had newSupplyIndex)
    (err, localResults.newBorrowIndex) = calculateInterestIndex(market.borrowIndex, market.borrowRateMantissa
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.SUPPLY_NEW_BORROW_INDEX_CALCULATION_FAILED);
```

```
        /////////////////////////
        // EFFECTS & INTERACTIONS
        // (No safe failures beyond this point)

        // We ERC-20 transfer the asset into the protocol (note: pre-conditions already checked above)
        err = doTransferIn(asset, msg.sender, amount);
        if (err != Error.NO_ERROR) {
            // This is safe since it's our first interaction and it didn't do anything if it failed
            return fail(err, FailureInfo.SUPPLY_TRANSFER_IN_FAILED);
        }

        // Save market updates
        market.blockNumber = getBlockNumber();
        market.totalSupply =  localResults.newTotalSupply;
        market.supplyRateMantissa = localResults.newSupplyRateMantissa;
        market.supplyIndex = localResults.newSupplyIndex;
        market.borrowRateMantissa = localResults.newBorrowRateMantissa;
        market.borrowIndex = localResults.newBorrowIndex;

        // Save user updates
        localResults.startingBalance = balance.principal; // save for use in `SupplyReceived` event
        balance.principal = localResults.userSupplyUpdated;
        balance.interestIndex = localResults.newSupplyIndex;

        emit SupplyReceived(msg.sender, asset, amount, localResults.startingBalance, localResults.userSupplyUpdated);

        return uint(Error.NO_ERROR)
    }
```

```
function doTransferIn(address asset, address from, uint amount) internal returns (Error) {
    EIP20NonStandardInterface token = EIP20NonStandardInterface(asset);

    bool result;

    token.transferFrom(from, address(this), amount);
```

Reentrancy – Case Study - Lendf.me

```
function withdraw(address asset, uint requestedAmount) public returns (uint) {
    if (paused) {
        return fail(Error.CONTRACT_PAUSED, FailureInfo.WITHDRAW_CONTRACT_PAUSED);
    }

    Market storage market = markets[asset];
    Balance storage supplyBalance = supplyBalances[msg.sender][asset];

    WithdrawLocalVars memory localResults; // Holds
    Error err; // Re-used for every function call t
    uint rateCalculationResultCode; // Used for 2 i

    // We calculate the user's accountLiquidity and
    (err, localResults.accountLiquidity, localResul
    if (err != Error.NO_ERROR) {
        return fail(err, FailureInfo.WITHDRAW_ACCOU
    }

    // We calculate the newSupplyIndex, user's supp
```

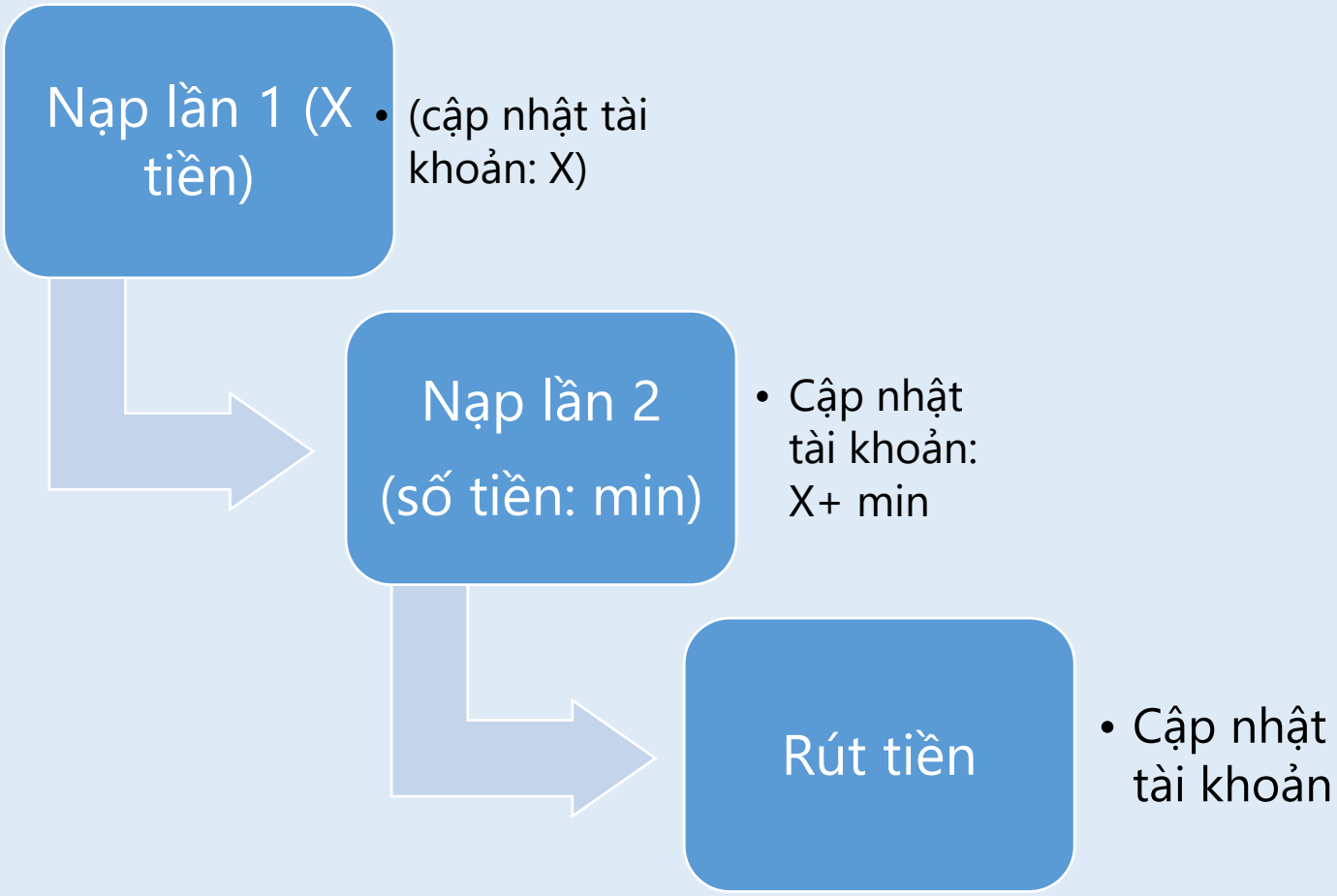
```
    if (err != Error.NO_ERROR) {
        // This is safe since it's our first interaction and it didn't do anything if it failed
        return fail(err, FailureInfo.WITHDRAW_TRANSFER_OUT_FAILED);
    }

    // Save market updates
    market.blockNumber = getBlockNumber();
    market.totalSupply =  localResults.newTotalSupply;
    market.supplyRateMantissa = localResults.newSupplyRateMantissa;
    market.supplyIndex = localResults.newSupplyIndex;
    market.borrowRateMantissa = localResults.newBorrowRateMantissa;
    market.borrowIndex = localResults.newBorrowIndex;

    // Save user updates
    localResults.startingBalance = supplyBalance.principal; // save for use in `SupplyWithdrawn` event
    supplyBalance.principal = localResults.userSupplyUpdated;
    supplyBalance.interestIndex = localResults.newSupplyIndex;

    emit SupplyWithdrawn(msg.sender, asset, localResults.withdrawAmount, localResults.startingBalance, localResults.userSupplyUpdated);

    return uint(Error.NO_ERROR); // success
}
```

Reentrancy – Case Study - Lendf.me

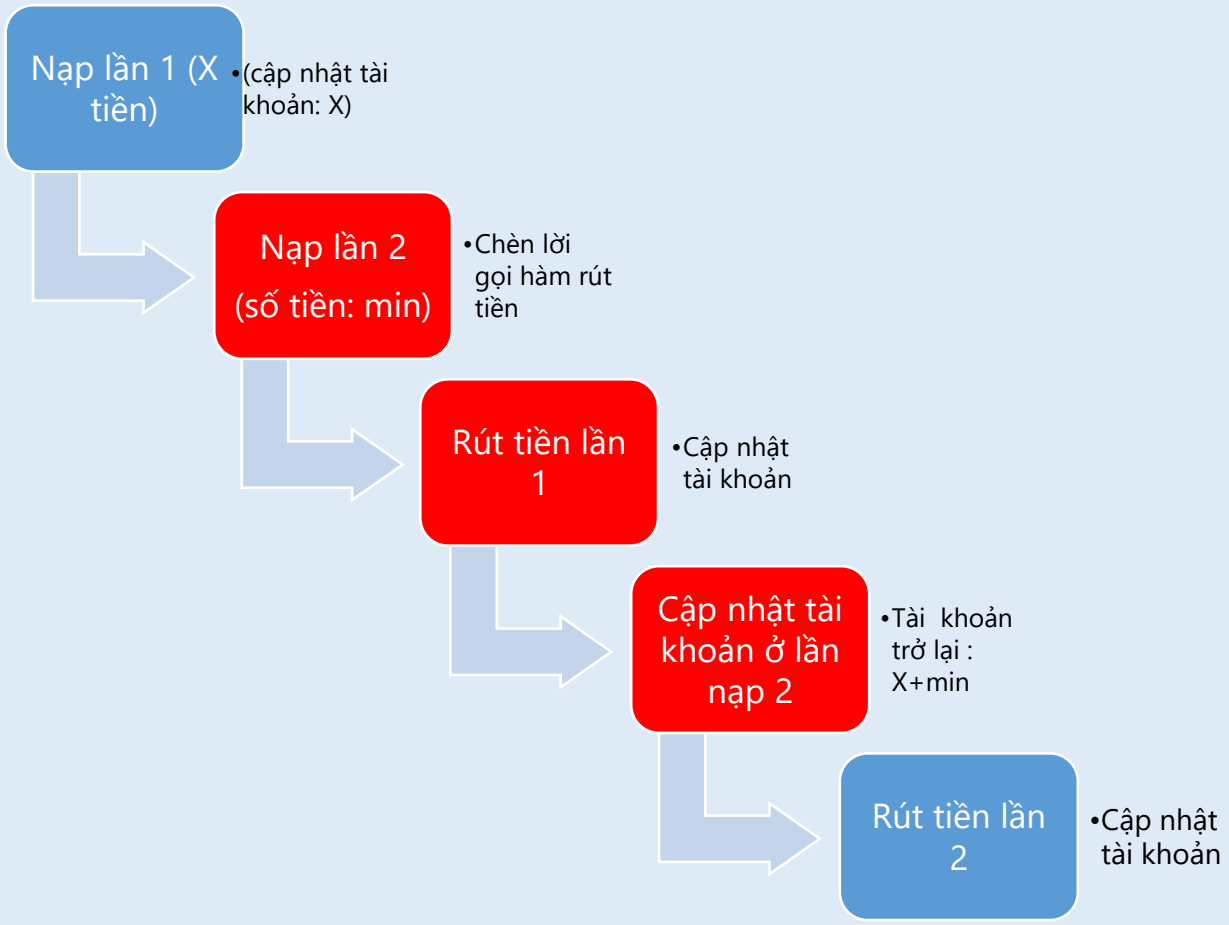
```
function _transferFrom(address holder, address recipient, uint256 amount) internal returns (bool) {
    require(recipient != address(0), "ERC777: transfer to the zero address");
    require(holder != address(0), "ERC777: transfer from the zero address");

    address spender = msg.sender;

    _callTokensToSend(spender, holder, recipient, amount, "", "");

    _move(spender, holder, recipient, amount, "", "");

    _approve(holde

    _callTokensRec

    return true;
}
```

```
function _callTokensToSend(
    address operator,
    address from,
    address to,
    uint256 amount,
    bytes memory userData,
    bytes memory operatorData
)
    internal
{
    address implementer =  erc1820.getInterfaceImplementer(from, TOKENS_SENDER_INTERFACE_HASH);
    if (implementer != address(0)) {
        IERC777Sender(implementer).tokensToSend(operator, from, to, amount, userData, operatorData);
    }
}
```

Reentrancy – Case Study - Lendf.me



Reentrancy – Case Study - Lendf.me



Reentrancy – Case Study - X

Đỗ Quang Thắng 5/18 9:34 PM

1 demo nhỏ [redacted] bị reentrancy

https://testnet.bscscan.com/tx/0xaa67f71c495bd96b9a26434a17d7f8dbdbe2f61b7b708e0cb1809[redacted]

```
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080...
└ TRANSFER 0.5 BNB From 0xc7ce79103ea72ef38527aa9d... To → 0xef393bf117edba7773dbe080
```

Reentrancy – Case Study - X

```
function cancelBid(uint256 _bidId) external whenNotPaused {
    Bid memory bid = bids[_bidId];
    require(
        bid.owner == msg.sender && bid.status,
        "Wrong-order-owner-or-cancelled"
    );
    if (bid.quantity > 0) {
        _paid(bid.paymentToken, msg.sender, bid.price, 0, 0);
    }

    // IERC20Upgradeable(bid.paymentToken).safeTransfer(
    //     msg.sender,
    //     bid.price
    // );
    bid.quantity = 0;
    bid.status = false;
    bids[_bidId] = bid;
    emit BidCancelled(_bidId);
}
```

DoS with (unexpected) revert

```
function createBid(uint256 auctionId, uint256 price) external payable nonReentrant auctionExists(auctionId) {
    require(marketPlaceItems[auctionId].endTime > block.timestamp, "Auction expired");
    require(marketPlaceItems[auctionId].marketItemType == MarketItemsType.AUCTION, "Can not bid on listed Item");
    bool firstBidder = true;
    require(marketPlaceItems[auctionId].reservePrice < price, "Bid price have to greater than reservePrice");
    require(price > marketPlaceItems[auctionId].price + (marketPlaceItems[auctionId].price * minBidIncrementPercentage /100),
        "Bid price have to greater than last bid by minBidIncrementPercentage");


    if(marketPlaceItems[auctionId].bidder != address(0)) {
        firstBidder = false;
        _handleOutgoingFund(marketPlaceItems[auctionId].bidder,marketPlaceItems[auctionId].price,
        marketPlaceItems[auctionId].currency);
        emit AuctionBidCanceled(auctionId, marketPlaceItems[auctionId].tokenId,
        marketPlaceItems[auctionId].tokenAddress, marketPlaceItems[auctionId].bidder, marketPlaceItems[auctionId].price);
    }
    _handleIncomingFund(price, marketPlaceItems[auctionId].currency);
    marketPlaceItems[auctionId].bidder = msg.sender;
    marketPlaceItems[auctionId].price = price;
    emit AuctionBid(auctionId, marketPlaceItems[auctionId].tokenId,
    marketPlaceItems[auctionId].tokenAddress, msg.sender, price, firstBidder);
}
```

Logic Bug

- Fantom – Brigde

Three chains, five assets: Fantasm Finance drained for $2.6 million

According to the official announcement shared by Fantasm Finance DeFi protocol, its mechanism was drained of $2,600,000 on March 9, 2022.

> *Dear Community, we have published a Post Mortem for the Fantasm Finance Exploit on 09 March 2022.Please read the article below covering:- What happened- Forensic Analysis- Repayment Plan and Proposed Steps- Next Tasks$FSM $XFTM— Fantasm Finance (@fantasm_finance) March 10, 2022*

Logic Bug

- Users normally would need to supply $0.981 $FTM + $0.019 $FSM to mint $1 $XFTM
- But…

```
function mint(uint256 _fantasmIn, uint256 _minXftmOut) external payable nonReentrant {
    require(!mintPaused, "Pool::mint: Minting is paused");
    uint256 _ftmIn = msg.value;
    address _minter = msg.sender;

    (uint256 _xftmOut, , uint256 _minFantasmIn, uint256 _ftmFee) = calcMint(_ftmIn, _fantasmIn);
    require(_minXftmOut <= _xftmOut, "Pool::mint: slippage");
    require(_minFantasmIn <= _fantasmIn, "Pool::mint: Not enough Fantasm input");
    require(maxXftmSupply >= xftm.totalSupply() + _xftmOut, "Pool::mint: > Xftm supply limit");

    WethUtils.wrap(_ftmIn);
    userInfo[_minter].lastAction = block.number;

    if (_xftmOut > 0) {
        userInfo[_minter].xftmBalance = userInfo[_minter].xftmBalance + _xftmOut;
        unclaimedXftm = unclaimedXftm + _xftmOut;
    }

    if (_minFantasmIn > 0) {
        fantasm.safeTransferFrom(_minter, address(this), _minFantasmIn);
        fantasm.burn(_minFantasmIn);
    }

    if (_ftmFee > 0) {
        WethUtils.transfer(feeReserve, _ftmFee);
    }

    emit Mint(_minter, _xftmOut, _ftmIn, _fantasmIn, _ftmFee);
}
```

Unchecked

uint256 _minFtmIn

Missing require(_minFtmIn <= _ftmIn, “”);

- the attacker can just supply $0.019 $FSM to mint $1

Flash Loan Attack

https://bean.money/blog/beanstalk-governance-exploit

Search

Beanstalk Farms · April 19th, 2022

Beanstalk Governance Exploit

Beanstalk was attacked on April 17, resulting in a theft of ~$76M in non-Beanstalk user assets.

Beanstalk, a decentralized credit based stablecoin protocol, was attacked at roughly 12:24pm UTC on April 17, resulting in a theft of ~$77M in non-Beanstalk user assets. The perpetrator used a flash loan to exploit the protocol's governance mechanism and send the funds to a wallet they controlled. Beanstalk Farms, the decentralized development team working on Beanstalk, is preparing a strategy to safely re-launch a more secure Beanstalk with a path forward.

Flash Loan Attack

1. Flash Loan — was crucial to the attacker obtaining majority control of Beanstalk's on-chain governance.
2. Governance Mechanism — the emergencyCommit() function that permitted the attacker to immediately execute the proposal on-chain. Normally, the execution of a BIP on-chain requires a minimum of 7 days but leveraging the emergencyCommit function helped the attacker bypass this (a 2/3rd share was necessary for execution)

6.5.4 Beanstalk Improvement Proposals

Beanstalk implements EIP-2535.[19] Beanstalk is a diamond with multiple facets. Beanstalk supports multiple simultaneous *BIPs* with independent *Voting Periods*.

A BIP has four inputs: (1) Beanstalk should *Pause* or *Unpause*, (2) a list of facets and functions to add or remove upon commit, (3) a function to run upon commit, and (4) the Ethereum address of the contract holding the function to run upon commit.

If inputs 2, 3 and 4 are empty, a *BIP* can pass with a two-thirds supermajority vote at any time before the end of the *Voting Period*.

In a gist, the attacker utilized the flash loan to garner just enough collateral to become a supermajority, pass a BIP and swindle $182 m from Beanstalk farms.

Thank you for your attention!

manhtung@cyradar.com