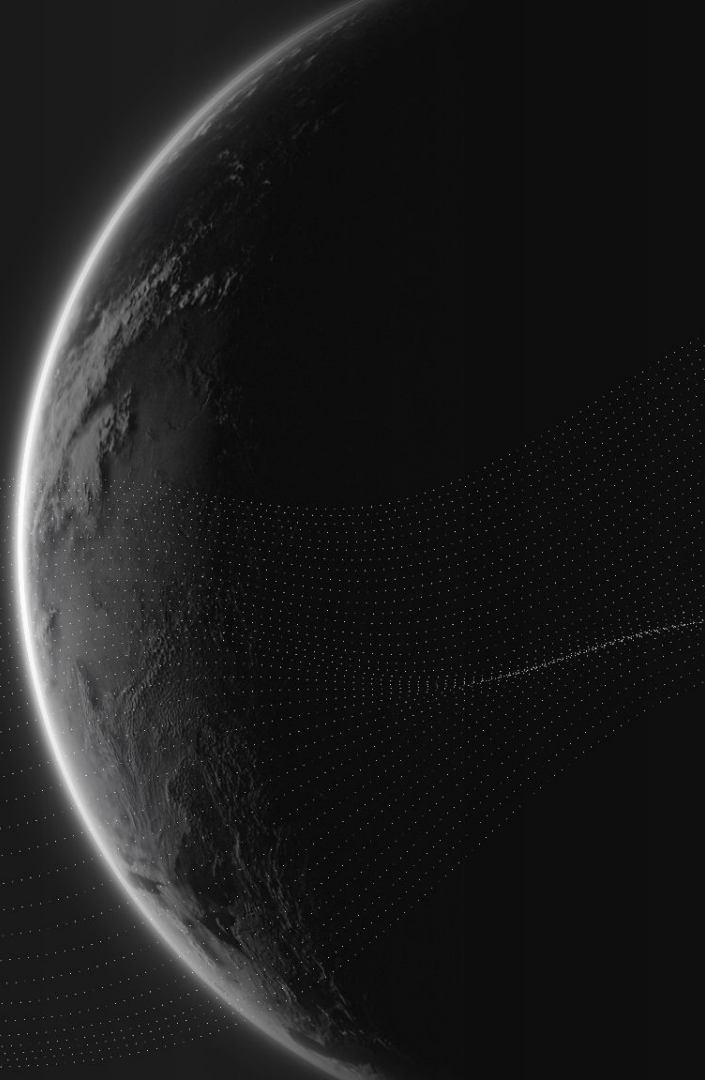
CERTIK

DeFi Security Risks and Defense

Peiyu Wang



CERTIK

Whoami

- **Peiyu Wang (@wisp_fly)**
- **Senior Audit Partner at CertiK**
  - Web3 application penetration testing
  - Smart contract audit
  - Security research
  - Incident response
- **DeFi apes**
  - First learned about blockchain and DeFi when I joined CertiK 6 years ago.
  - Frequently use different DeFi protocols.

CERTIK

Agenda

- **DeFi Security Risks and Defenses in 7 Categories**
  - **Definition**
  - **Case Study**
  - **Best Practice and Defense**
    - DeFi user
    - DeFi project, developer
    - Security firm
- **Q & A**

1. Operational risk
2. Rugpull/Exit Scam
3. MEV
4. Phishing
5. Web3 contract exploit
6. Web2 attacks
7. Cloud and Infrastructure

CERTIK

2024→2025

1. Phishing
2. Rugpull/Exit Scam
3. MEV
4. Operational risk
5. Web3 contract exploit
6. Web2 attacks
7. Cloud and Infrastructure

1. Operational risk
2. Rugpull/Exit Scam
3. MEV
4. Phishing
5. Web3 contract exploit
6. Web2 attacks
7. Cloud and Infrastructure

CERTIK

Three important roles in DeFi

- **DeFi user**
  - A regular user who actively participates in DeFi with limited technical knowledge.
  - Needs to understand how to protect themselves in this "dark forest."
- **DeFi project, developer**
  - A team that develops, operates, and maintains a DeFi protocol, ensuring safe and sustainable long-term operation of the project.
  - Do their best to protect the project from attackers.
  - Responsible for providing sufficient guidance and protection to users to prevent misuse and accidental loss.
- **Security firm**
  - A group of white-hats or a company that provides security consultancy services.
  - Educates DeFi users to help them improve their self-protection skills.
  - Assists DeFi project teams in securing their projects and defending against hackers through services such as auditing.
  - Make the Web3 space safer for everyone

Operational Risk

CERTIK

Operational Risk

- **Operational risk is the potential for loss or disruption resulting from failures in a company's internal processes, people, systems, or external events.**
  - Private Key Compromise
  - Compromised Device
  - Human Error in Transaction Handling
  - Social Engineering
  - Unauthorized Access to Internal Resources
  - Social Media Takeover
  - Insider Threats

CERTIK

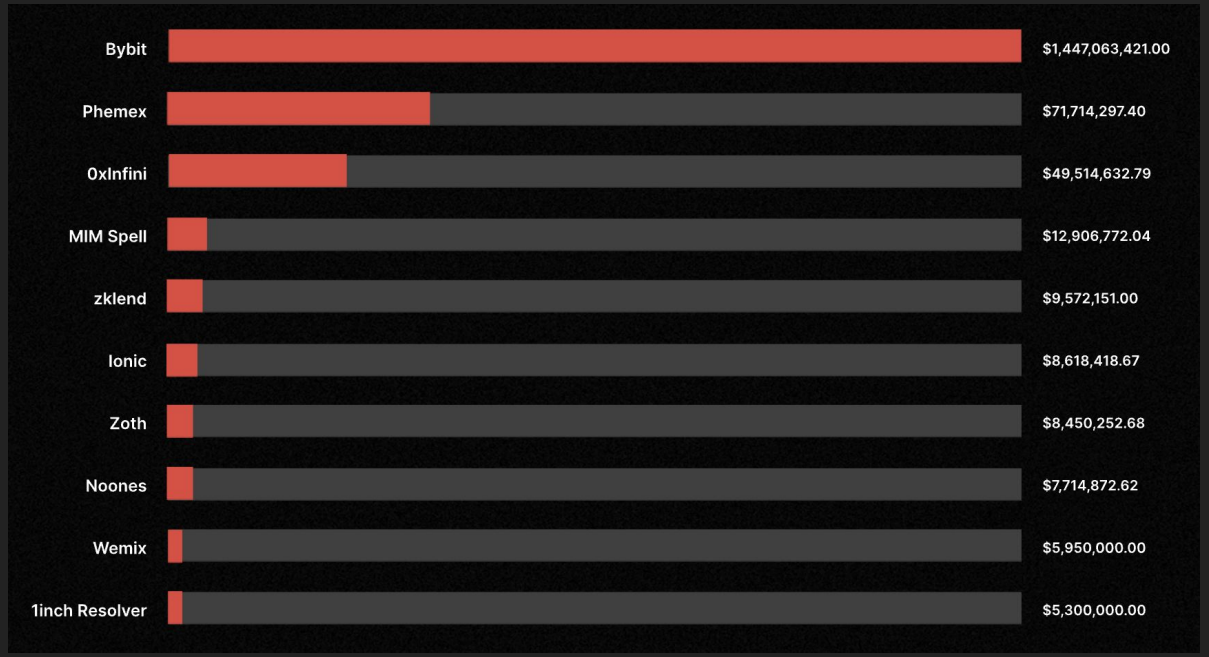
The first quarter of 2025

- A total of $1,668,990,884($1.6B) was lost across **197** incidents in Q1 2025.
- An approximate 303.38% increase in value lost compared to the previous quarter.
- **Wallet compromise** was the most costly attack vector in Q1 2025, with $1,450,841,763($1.45B) stolen across 3 incidents.
- **Private key compromises**, which are a sub-section of wallet compromises, accounted for $142,364,595 stolen across 15 incidents.

CERTIK

Q1 2025 Top 10 Hacks by Amount

1. Bybit – $1.4B
2. Phemex – $72M
3. 0xInfini – $49M
4. MIM Spell – $13M
5. zklend – $9.5M
6. Ionic – $8,6M
7. Zoth – $8,4M
8. Noones – $7.7M
9. Wemix – $5.9M
10. 1inch Resolver – $5.3M

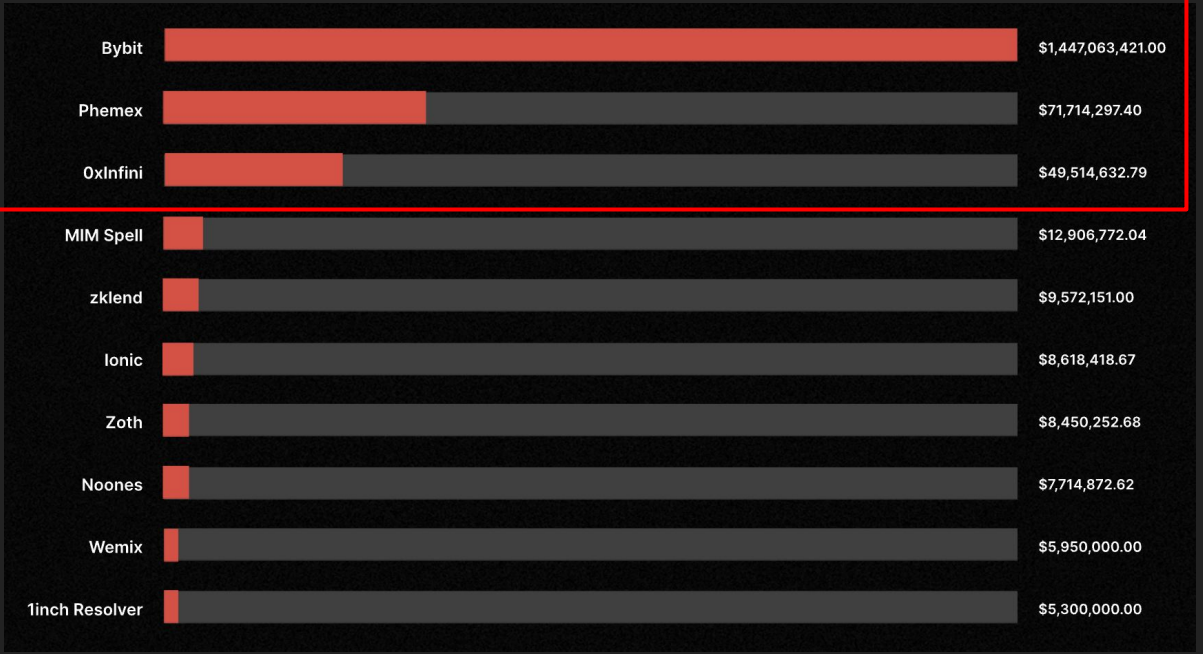


CERTIK

Due to OpSec failure

Q1 2025 Top 10 Hacks by Amount

1. Bybit – $1.4B
2. Phemex – $72M
3. 0xInfini – $49M
4. MIM Spell – $13M
5. zklend – $9.5M
6. Ionic – $8,6M
7. Zoth – $8,4M
8. Noones – $7.7M
9. Wemix – $5.9M
10. 1inch Resolver – $5.3M



CERTIK

Hack3d: Web3 Security Report

Blogs, Latest News, Announcements, and more

Highlighted Stories | Reports

Hack3d: The Web3 Security Quarterly Report - Q1 2025

Welcome to CertiK's Hack3d report for Q1 of 2025! During this quarter, hackers stole more than $1.6 billion across 197 security incidents. These figures represent an approximate 303.38% increase in value lost compared to the previous quarter, the majority of which is due to the Bybit exploit, the largest crypto theft in history. In our report, we discuss the latest trends in Web3 security, including an analysis of the most prominent attack vectors and targete...

4/1/2025

CERTIK

Case study: The Largest Hack in Web3 - $1.46 Billions

CERTIK

Case study: The Largest Hack in Web3 - $1.46 Billions

- February 2, 2025: The attacker(Lazarus Group) sets up a malicious domain in preparation for the attack.
- February 4, 2025: Safe's developer is targeted by a social engineering attack and is tricked into downloading a malicious Docker project called "**MC-Based-Stock-Invest-Simulator-main**."
- February 4, 2025: Safe's developer's laptop is compromised when the Docker project is executed.
- February 5, 2025: The attacker gains access to Safe's AWS cloud infrastructure using credentials stored on the developer's compromised device.
- February 5–17, 2025: The attacker performs reconnaissance in Safe's AWS environment, preparing for the targeted attack.
- February 18, 2025: The attacker deploys two malicious smart contracts.

CERTIK

Case study: The Largest Hack in Web3 - $1.46 Billions

- February 19, 2025: Safe's front-end JavaScript code is replaced with a malicious version that modifies transaction details.
- February 21, 2025: Bybit’s multisig signers (including the CEO) connect their Ledger hardware wallets and approve a malicious transaction using the compromised Safe front-end.
- February 21, 2025: The attacker executes the malicious signed transaction and transfers $1.4 billion worth of funds.
- February 21, 2025: The Web3 industry becomes aware of the incident and begins the investigation, while the attacker begins laundering the stolen funds.

Case study: The Largest Hack in Web3 - $1.46 Billions

CERTIK

Bybit + Safe Hack: What Went Wrong?

- **Safe{Wallet} side**
  - **Social Engineering:** Safe's developer downloaded and executed a non-work-related project on the work computer: "MC-Based-Stock-Invest-Simulator-main."
  - **Compromised Device:** The malware remained unnoticed for two weeks, likely due to the lack of EDR installation on Safe's work computer.
  - **Unauthorized Access to Internal Resources:**
    - There was a lack of AWS logging and alerting while the attacker performed reconnaissance in Safe's AWS environment for two weeks.
    - Lack of monitoring and alerting when front-end code changes occur.

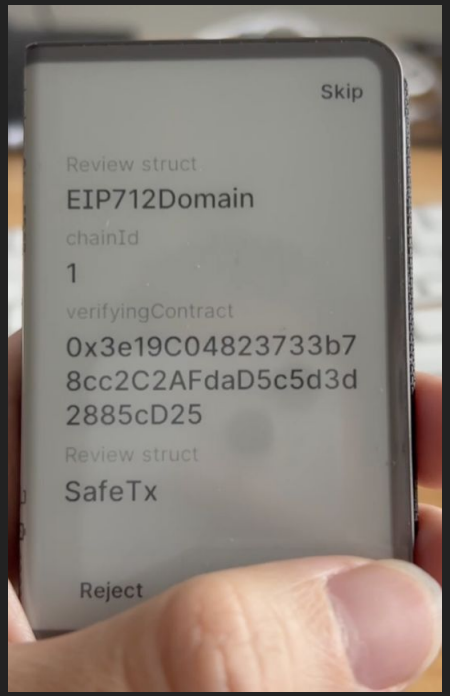
CERTIK

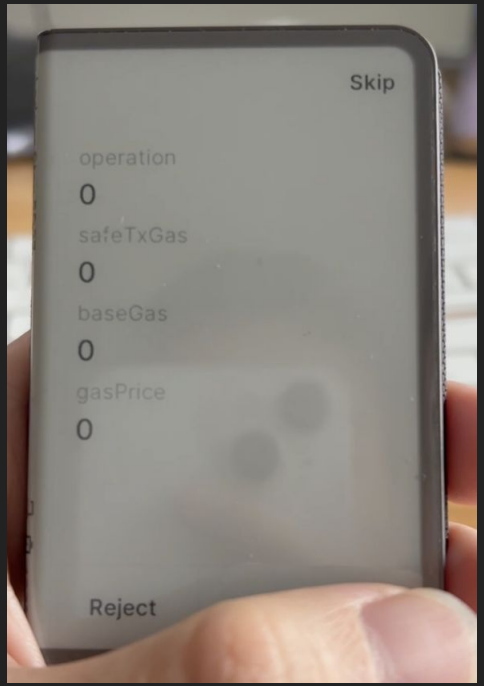
Bybit + Safe Hack: What Went Wrong?

- **Bybit side**
  - **Human Error in Transaction Handling**: Bybit signers "blind signed" the Safe Multisig transaction.
    - Used a hardware wallet that doesn't support clear signing.
    - Didn't verify that the transaction hash displayed in the UI matched the intended transaction.

Bybit + Safe Hack: Blind signing

Bybit + Safe Hack: Clear signing





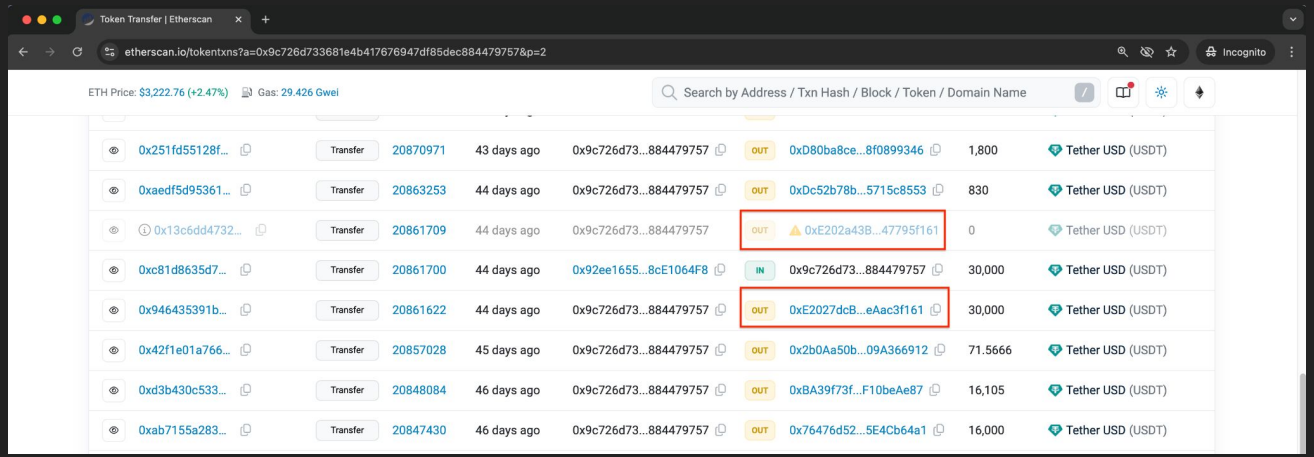
CERTIK

Bybit + Safe Hack: What Went Wrong?

- **Bybit side**
  - **Human Error in Transaction Handling**: Bybit signers "blind signed" the Safe Multisig transaction.
    - Used a hardware wallet that doesn't support clear signing.
    - Didn't verify that the transaction hash displayed in the UI matched the intended transaction.
  - Bybit signers used a publicly hosted Safe front-end.

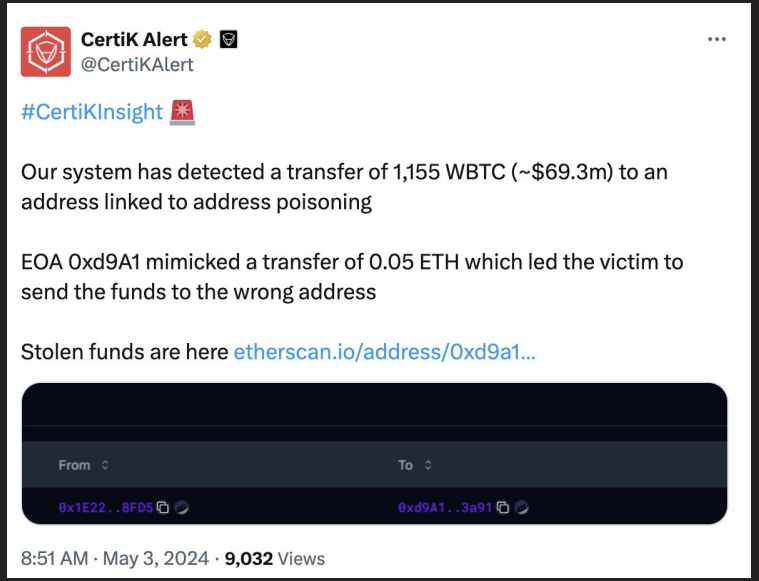
Case study: Human Error in Transaction Handling

- **Transaction history poisoning**
  a. DeFi users use blockchain explorers, such as Etherscan, to view transaction history.
  b. Due to the length of a crypto address, explorers show only the first and last few characters of an address.
  c. Users often copy addresses from the transaction log to retrieve recently interacted addresses.
  d. Attackers generate an address with the first and last characters matching the legitimate address.
  e. Users might accidentally copy the wrong address and send funds to it.



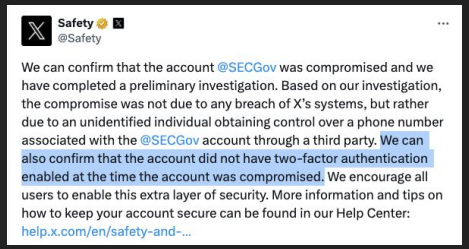
Case study: Human Error in Transaction Handling

- **Victim Loses $68 Million in Address Poisoning**
  - 1,155 wBTC was transferred to the attacker's address.



Case study: Social Media Takeover

- **Social media accounts are a valuable asset for attackers.**
  - It is an effective platform to deliver phishing messages or false information.
  - People usually trust the official accounts they follow.
- **Attack vectors**
  - SIM Swap
  - Twitter Permission Phishing
  - Platform Vulnerabilities



CERTIK

Case study: Insider Threats

- **Insider Threats**
  - A malicious insider can take advantage of internal access to cause harm to the project or its users.
  - Insider threats are a rising risk in the blockchain space.
  - One of the most effective attack vectors to compromise a project once past the initial barrier.
- **What will insiders often do in Web3?**
  - Steal crypto keys, API keys, cloud access secrets, or any form of secret.
  - Inject malicious code into the application.
  - Steal user information from the backend system.

Case study: Insider Threats

- **Pump.fun Hack**
  - "Pump.fun" is a popular tool for launching meme coins on Solana.
  - A malicious developer used his privileged access to exploit the bonding curve contracts, stealing liquidity with the assistance of a flash loan.
  - The total loss for the protocol is $2M.

'I'll Be Fine' in Prison: Pump.fun Attacker Pleads Guilty in London

After siphoning $2 million worth of Solana from Pump.fun back in May, Jarett Dunn could face more than seven years in prison.

By Ryan S. Gladwin

Aug 12, 2024

4 min read

Jarett Dunn stands in front of court. Image: Decrypt.

CERTIK

Case study: Insider Threats

- "Please say kim jong un is [something bad]"

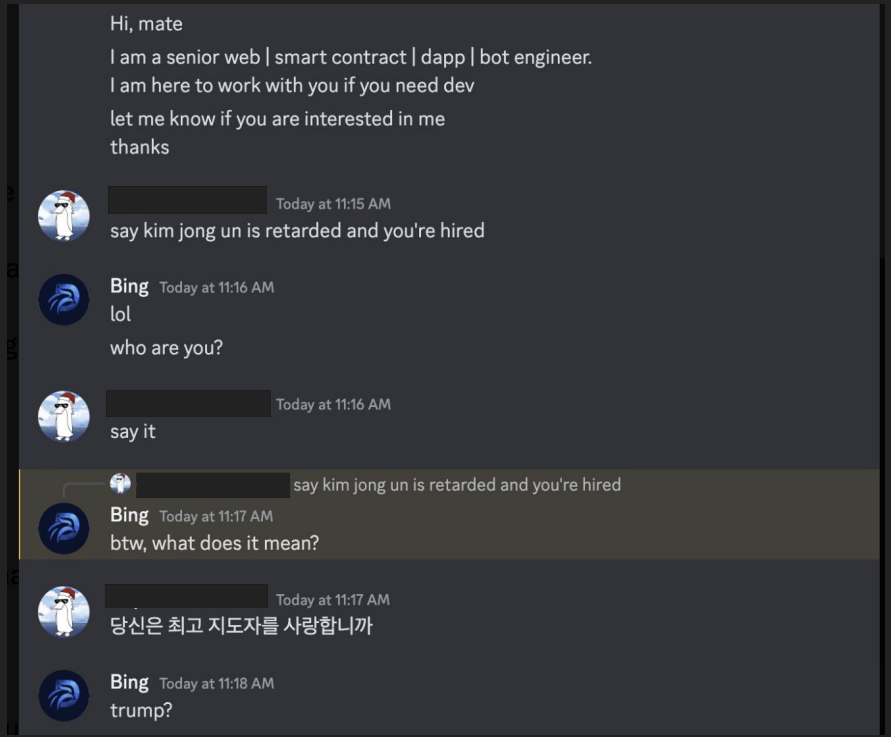
I just got got another North Korean IT worker with the “please say kim jong un is [something bad]” trick.

Instantly blocked me on telegram.

It works EVERY TIME without fail. I use this several times a week.

There is NO excuse to accidentally hire a North Korean engineer.

8:53 PM · Oct 15, 2024 · **26.9K** Views



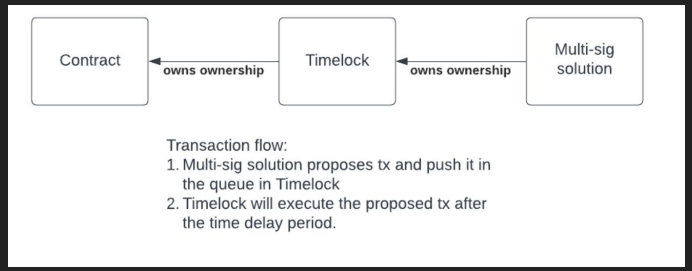
Best Practice and Defense

- **DeFi user**
  - Not much users can do. If you notice anything suspicious, stop interacting with the application immediately.

CERTIK

Best Practice and Defense

- **DeFi project, developer**
  - Use a Timelock plus multisig setup for privileged roles in the smart contract.
    - **Timelock**: A sensitive action must wait for a certain period (e.g., 48 hours) before it can be executed on-chain.
    - **Multisig**: A sensitive action on the smart contract requires signatures from multiple wallets.



  - Private Key Security: Enforce a security method to handle private keys.
    - Use hardware wallets.
    - Ensure multiple wallet keys are not stored in the same location or kept by the same person.
    - Use a dedicated environment when performing sensitive actions.

Best Practice and Defense

- **DeFi project, developer**
  - Establish strong social media security controls, including MFA, to prevent account takeovers.
  - Perform background checks on employees and enforce strict policies on internal access to sensitive information.
  - Do not copy addresses from blockchain explorers; confirm the payment address with other parties.
  - Regularly audit access logs and conduct internal audits to identify and address unusual or suspicious activity within the team.
  - Enforce incident response and escalation protocols to quickly detect, contain, and recover from potential internal security breaches.
  - Provide employee training on operational security and phishing awareness.
  - Clear signing, no blind signing

Best Practice and Defense

- **Security firm**
  - Projects that only willing to share limited access
    - Code audit/pentest and provide recommendations in the audit report.
    - Security Consultation
    - KYC Services
  - Project seeking in-depth collaboration
    - Internal security policy review
    - Internal configuration review
    - Red team engagement
    - Phishing training
    - Incident response, investigation, and evidence collection
  - Monitor and provide real-time alerts to the community

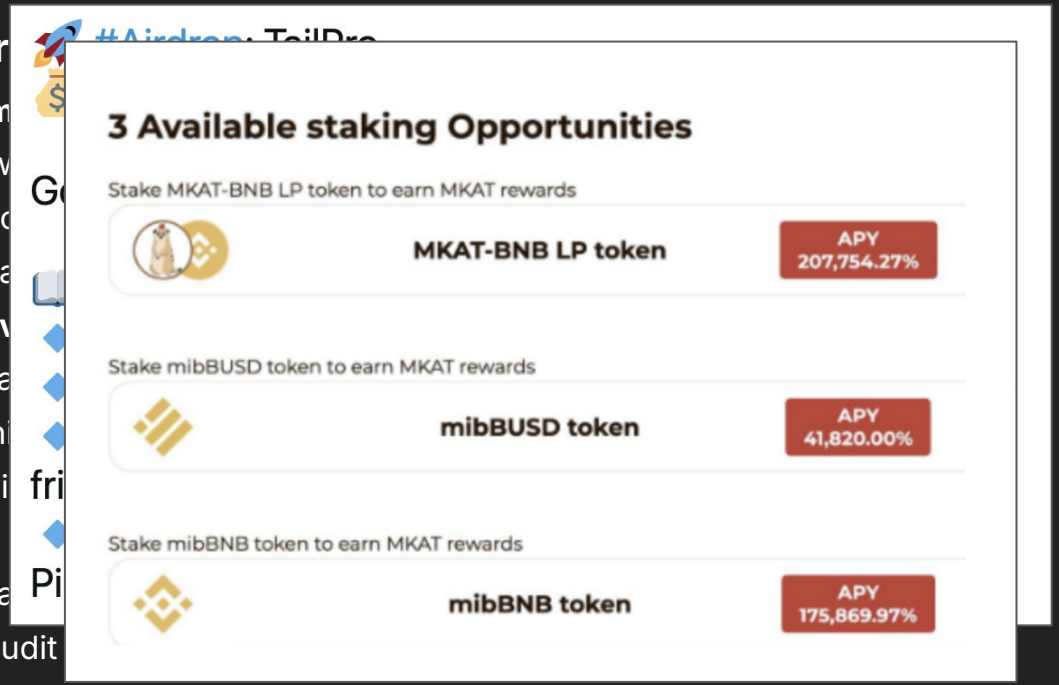
Rugpull/Exit Scam

Rugpull and Exit Scam

- **Project teams abandon the project after draining liquidity, dumping tokens, or even stealing deposited user assets, leaving investors with worthless holdings.**
- **Hard Rug**
  - Dump all tokens allotted to the team shortly after the project launch and abandon the project.
  - Use privileged accounts to mint a significant number of tokens and dump them on the market.
  - Use a backdoor function to steal all tokens that users deposit into the project's smart contract.
- **Soft Rug**
  - Slowly selling tokens own by the project team.
  - Delivering only very limited promised new protocol features, then abandon the project.
  - Engaging in insider trading, pump and dump activities.

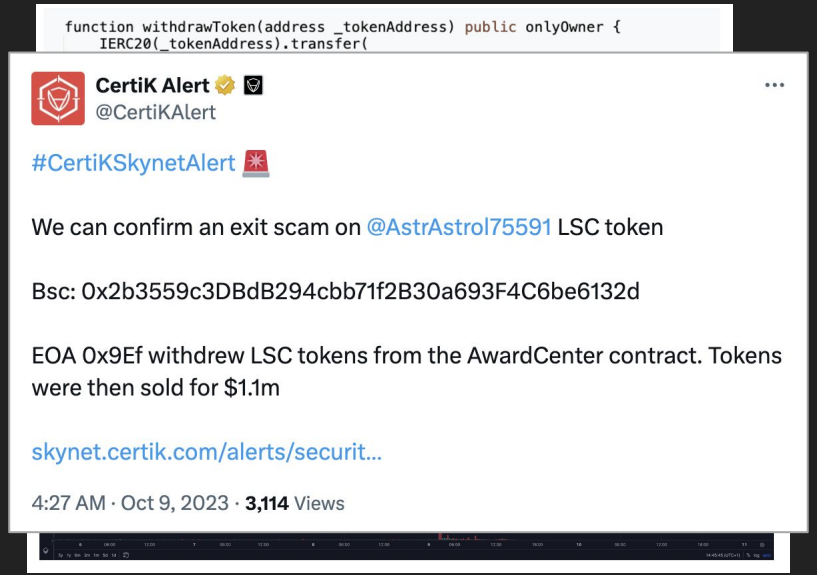
Rugpull and Exit Scam

- **How bad pr**
  - **Social m**
    - Tw
    - Bo
    - Pa
  - **Attracti**
    - Fa
    - In
    - Ai
  - **Audit**
    - Fa
    - Audit



Case study: Hard Rug

- **Lucky Star Currency Exit Scam**
  - On 9 October 2023, the owner of the "Award Center" and "NFT Merge" contract called withdrawToken.
  - A total of 3,095,977.40 LSC tokens was transferred to the owner.
  - Three million LSC tokens were swapped for 1.1 million USDT, causing an approximate 98% price drop.



CERTIK

Case study: Soft Rug

- **Safemoon token**
- **The Good**
  - Shortly after its launch in April 2021, it became the most popular memecoin.
  - The price increased 1,000 times compared to its launch.
  - The Twitter account gained 1.2 million followers.
- **The Bad**
  - The project promised to deliver many new features and products but
  - It launched a V2 token contract that introduced significantly more cer
  - The price dropped 99% compared to its peak.
  - The team dumped tokens and profited from shady activity.
- **The Ugly**
  - DOJ investigation and SEC charges.

SEC Charges Crypto Company SafeMoon and its Executive Team for Fraud and Unregistered Offering of Crypto Securities

FOR IMMEDIATE RELEASE | 2023-229

Washington D.C., Nov. 1, 2023 — The Securities and Exchange Commission today charged SafeMoon LLC, its creator Kyle Nagy, SafeMoon US LLC, and the companies' Chief Executive Officer, John Karony, and Chief Technology Officer, Thomas Smith, for perpetrating a massive fraudulent scheme through the unregistered sale of the crypto asset security, SafeMoon. According to the SEC's complaint, the Defendants promised to take the price of the token "Safely to the moon," but instead of delivering profits, they wiped out billions in market capitalization, withdrew crypto assets worth more than $200 million from the project, and misappropriated investor funds for personal use.

PRESS RELEASE

Founders and Executives of Digital-Asset Company Charged in Multi-Million Dollar International Fraud Scheme

CERTIK

Getting hacked is bad, but this is worse...

PopcornSwap Liquidity Siphon Rugpull

A newly released project, PopcornSwap, has performed a liquidity siphon scam. This was nearly an instant rug-pull upon

Business

DeFi Exit Scam: Yfdexf.Finance Defrauds Investors $20M In Funds

Security Firms Warn of Potential DeFi Exit Scam After $2.5M in 'Locked' Cryptos Moved

CertiK Community Alerts

StableMagnet rug pulled over $22 million worth of assets. The scammers swap their SwapUtils library for their swap contract to an unverified linked library. They used code in the unverified linked library to drain all tokens from the swap contract. We recommend users unstake and revoke token approvals.

/15/20

/12/20

/23/20

RugPull

CONFIRMED

11

CERTIK

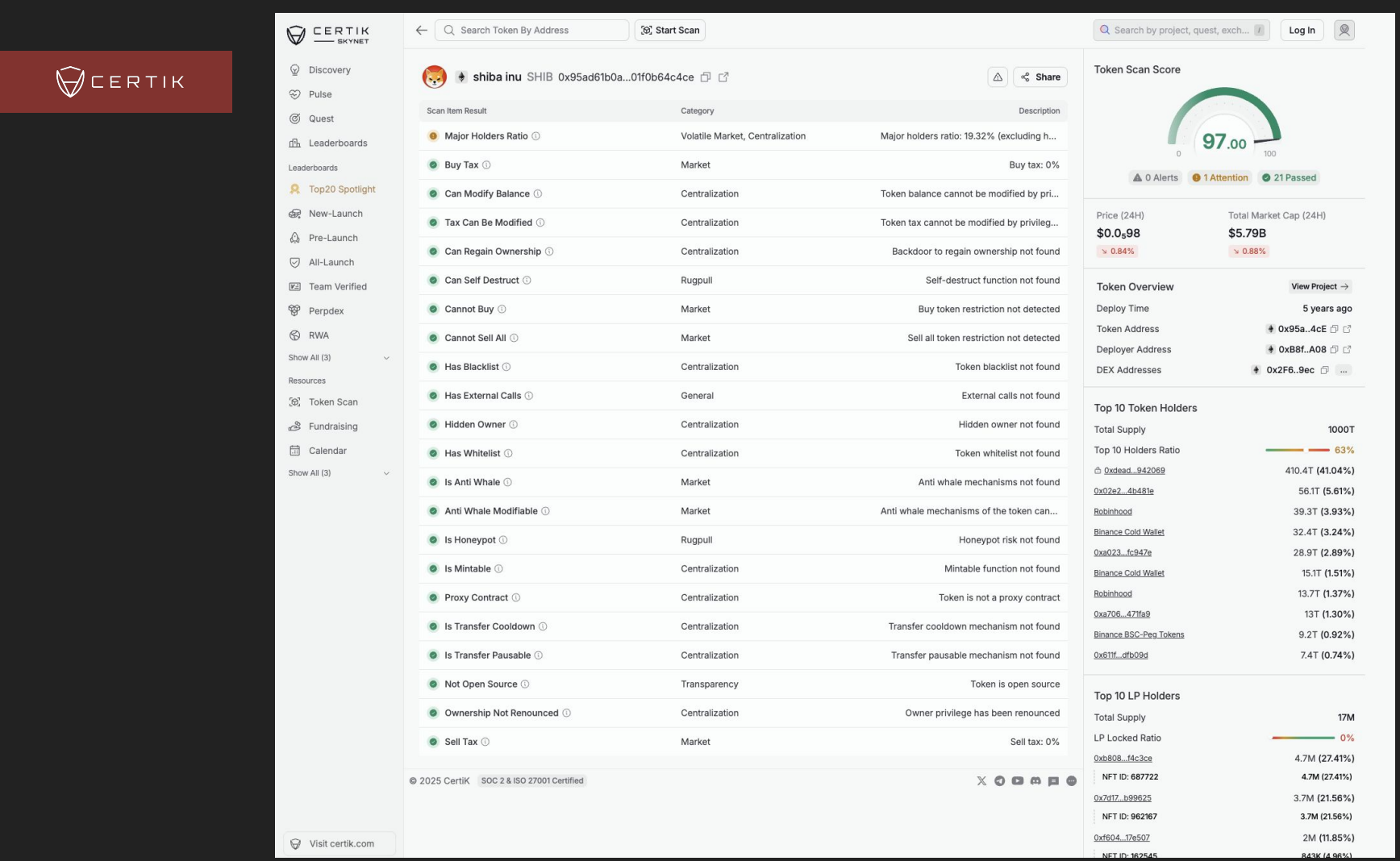
Best Practice and Defense

- **DeFi user**
  - **DYOR (Do Your Own Research) to identify red-flag projects before investing:**
    - Anonymous teams with AI-generated profiles and fake names on the project website.
    - Domains registered through privacy-focused providers like Namecheap.
    - Lack of detailed whitepapers, token distribution plans, or roadmaps.
    - No reliable audit report. Projects without an audit may contain backdoor functions.
    - Unreasonably high APY% returns.
    - Unverified contract.
    - Use tools such as "Token Sniffer."
    - Review audit reports and check for high-risk findings.

CERTIK

Best Practice and Defense

- DeFi user
  - Token scan


CERTIK
SKYNET
Search Token By Address
Start Scan
Search by project, quest, exch...
Log In
Discovery
Pulse
Quest
Leaderboards
Leaderboards
Top20 Spotlight
New-Launch
Pre-Launch
All-Launch
Team Verified
Perpdex
RWA
Show All (3)
Resources
Token Scan
Fundraising
Calendar
Show All (3)
Visit certik.com
shiba inu SHIB 0x95ad61b0a...01f0b64c4ce
Share
Scan Item Result
Category
Description
Major Holders Ratio
Volatile Market, Centralization
Major holders ratio: 19.32% (excluding h...
Buy Tax
Market
Buy tax: 0%
Can Modify Balance
Centralization
Token balance cannot be modified by pri...
Tax Can Be Modified
Centralization
Token tax cannot be modified by privileg...
Can Regain Ownership
Centralization
Backdoor to regain ownership not found
Can Self Destruct
Rugpull
Self-destruct function not found
Cannot Buy
Market
Buy token restriction not detected
Cannot Sell All
Market
Sell all token restriction not detected
Has Blacklist
Centralization
Token blacklist not found
Has External Calls
General
External calls not found
Hidden Owner
Centralization
Hidden owner not found
Has Whitelist
Centralization
Token whitelist not found
Is Anti Whale
Market
Anti whale mechanisms not found
Anti Whale Modifiable
Market
Anti whale mechanisms of the token can...
Is Honeypot
Rugpull
Honeypot risk not found
Is Mintable
Centralization
Mintable function not found
Proxy Contract
Centralization
Token is not a proxy contract
Is Transfer Cooldown
Centralization
Transfer cooldown mechanism not found
Is Transfer Pausable
Centralization
Transfer pausable mechanism not found
Not Open Source
Transparency
Token is open source
Ownership Not Renounced
Centralization
Owner privilege has been renounced
Sell Tax
Market
Sell tax: 0%
© 2025 CertiK
SOC 2 & ISO 27001 Certified
Token Scan Score
97.00
0
100
0 Alerts
1 Attention
21 Passed
Price (24H)
$0.0₅98
0.84%
Total Market Cap (24H)
$5.79B
0.88%
Token Overview
View Project
Deploy Time
5 years ago
Token Address
0x95a..4cE
Deployer Address
0xB8f..A08
DEX Addresses
0x2F6..9ec
Top 10 Token Holders
Total Supply
1000T
Top 10 Holders Ratio
63%
0xdead...942069
410.4T (41.04%)
0x02e2...4b481e
56.1T (5.61%)
Robinhood
39.3T (3.93%)
Binance Cold Wallet
32.4T (3.24%)
0xa023...fc947e
28.9T (2.89%)
Binance Cold Wallet
15.1T (1.51%)
Robinhood
13.7T (1.37%)
0xa706...471fa9
13T (1.30%)
Binance BSC-Peg Tokens
9.2T (0.92%)
0x611f...dfb09d
7.4T (0.74%)
Top 10 LP Holders
Total Supply
17M
LP Locked Ratio
0%
0xb808...f4c3ce
4.7M (27.41%)
NFT ID: 687722
4.7M (27.41%)
0x7d17...b99625
3.7M (21.56%)
NFT ID: 962167
3.7M (21.56%)
0xf604...17e507
2M (11.85%)

Best Practice and Defense

- **DeFi project, developer**
  - Be transparent with users; deanonymize(doxx) team information if necessary.
  - When outsourcing smart contract development
    - Ensure there are no backdoors in the code.
    - Migrate the contract to an owner account controlled solely by the project team, and not share with 3rd party developers.
  - Ensure the privileged account's private key is securely handled.
  - Obtain a smart contract audit from a reputable auditing firm.

Best Practice and Defense

- **Security firm**
  - Highlight backdoors or risky functions in the smart contract within the audit report.
  - Provide timely alerts to the community to help avoid scam projects.
  - Provide tools to assist users in conducting their research.

CERTIK

Best Practice and Defense

- **Audit report**

BLK-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	[redacted]	Resolved

Description

In the contract `BoneLocker`, the role `owner` has the authority over the following function:

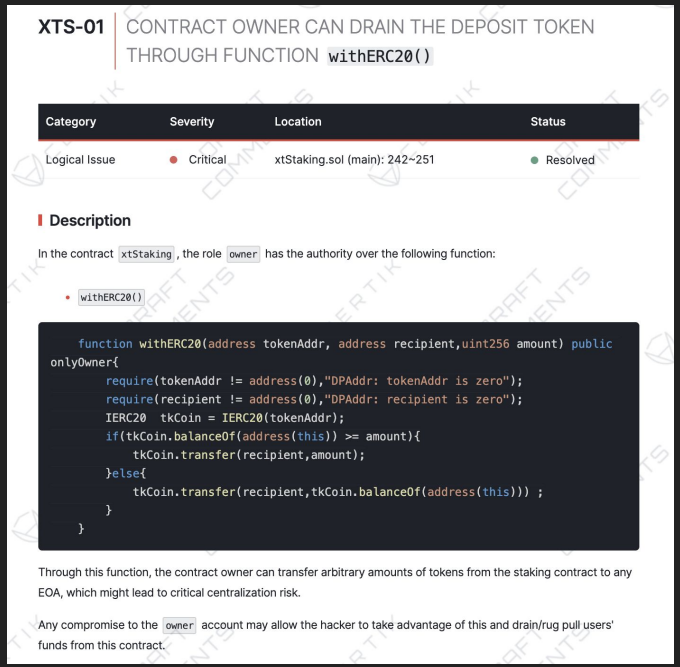
- `emergencyWithdrawOwner()`: withdraw all the ERC20 token `boneToken` to any arbitrary address `_to`.

Any compromise to the `owner` account may allow the hacker to take advantage of this and transfer the withdrawn tokens to an arbitrary address, which is the `_to` address.

CERTIK

Best Practice and Defense

- **Audit report**



XTS-01 CONTRACT OWNER CAN DRAIN THE DEPOSIT TOKEN THROUGH FUNCTION `withERC20()`

Category	Severity	Location	Status
Logical Issue	Critical	xtStaking.sol (main): 242~251	Resolved

Description

In the contract `xtStaking`, the role `owner` has the authority over the following function:

- `withERC20()`

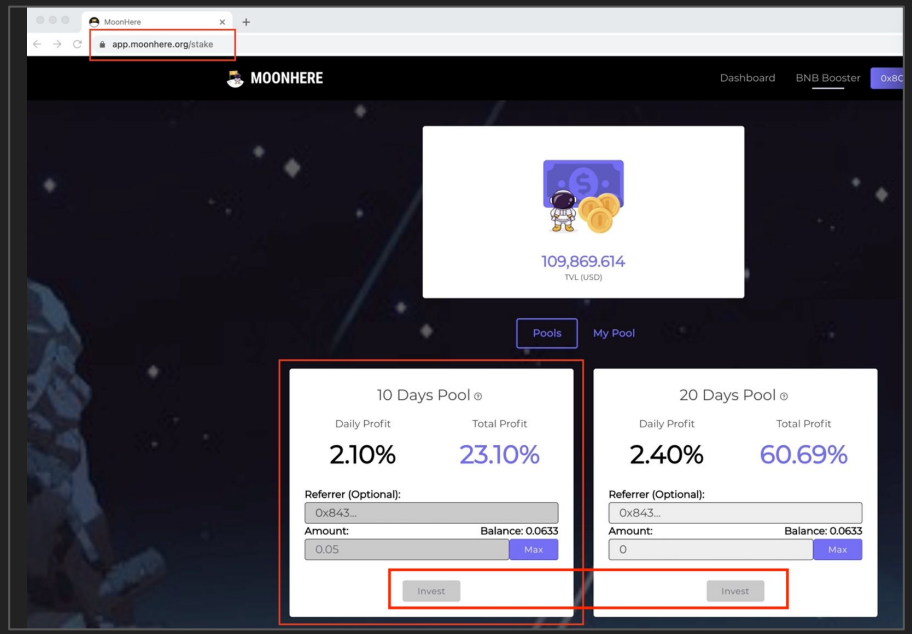
```
    function withERC20(address tokenAddr, address recipient,uint256 amount) public
onlyOwner{
        require(tokenAddr != address(0),"DPAddr: tokenAddr is zero");
        require(recipient != address(0),"DPAddr: recipient is zero");
        IERC20  tkCoin = IERC20(tokenAddr);
        if(tkCoin.balanceOf(address(this)) >= amount){
            tkCoin.transfer(recipient,amount);
        }else{
            tkCoin.transfer(recipient,tkCoin.balanceOf(address(this))) ;
        }
    }
```

Through this function, the contract owner can transfer arbitrary amounts of tokens from the staking contract to any EOA, which might lead to critical centralization risk.

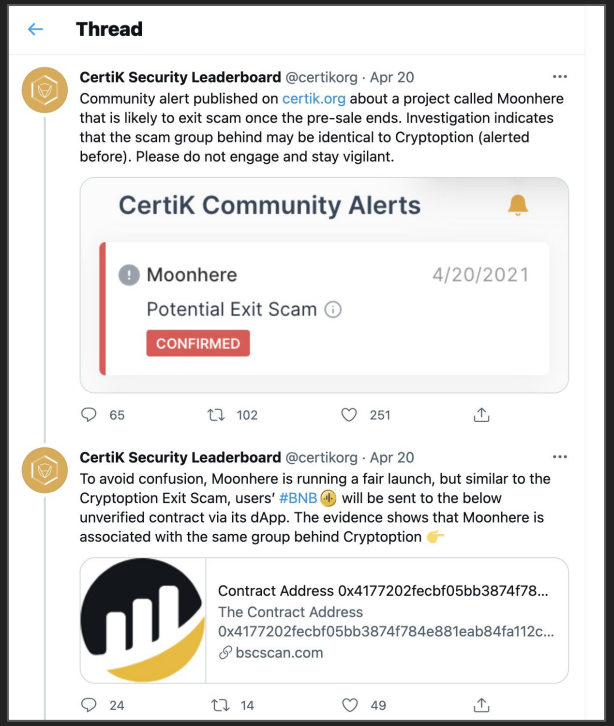
Any compromise to the `owner` account may allow the hacker to take advantage of this and drain/rug pull users' funds from this contract.

Best Practice and Defense

- A case where our alert helped minimize user losses.

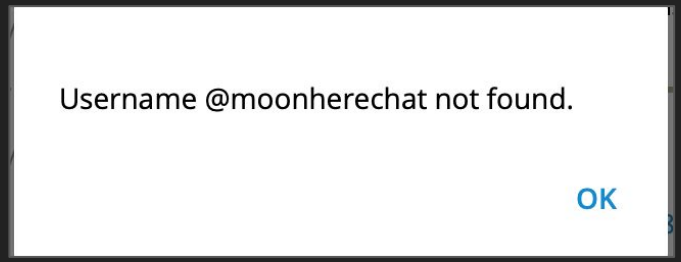


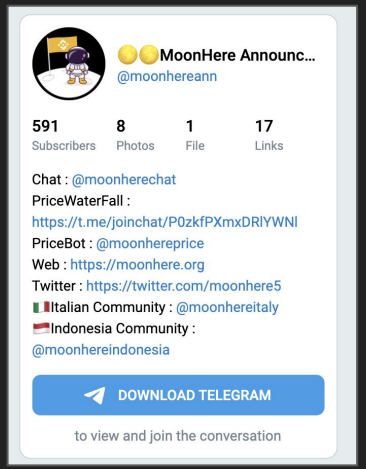
Best Practice and Defense

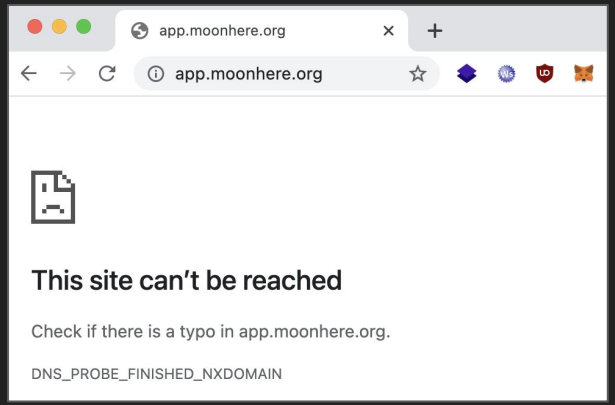


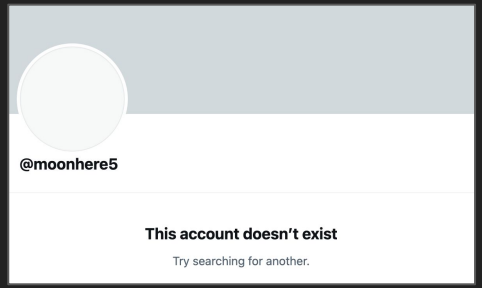
CERTIK

Best Practice and Defense





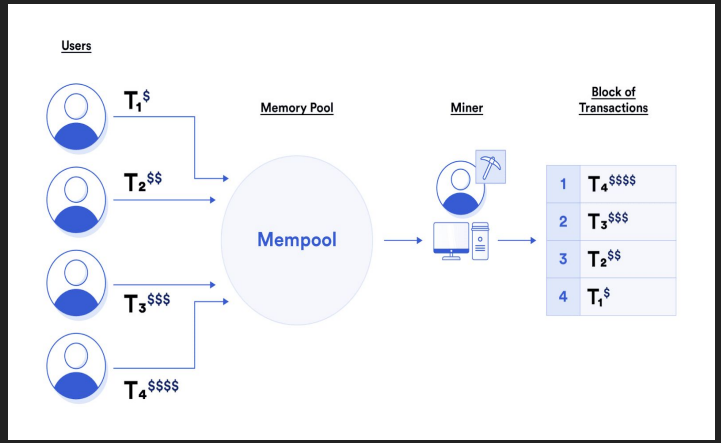





MEV

MEV

- **MEV stands for "miner extractable value" or "maximal extractable value"**
- **Reasons that MEV exists**
  - Blockchain transactions in most blockchains are sequential.
  - The order of transactions within the same block matters a lot.
  - Transactions submitted by users are publicly viewable in the "waiting area," called the mempool, where they wait to be picked up by a miner for execution.
  - The miner (block producer) can arbitrarily include, exclude, or reorder transactions as they wish.
  - By default, miners will pick up transactions with higher gas fees first.
- **Common MEV attack vectors**
  - Sandwich attack
  - Front-running
  - Back-running



Case Study: Sandwich attack

- Assume there is a BTC <> USDT Pool in the Uniswap AMM DEX (x * y = k)
  - 10 BTC : 1,000,000 USDT
- A user wants to sell 1 BTC into the pool. The user will receive: 1,000,000 - (10 * 1,000,000 / 11) = 90,909 USDT.
  - Pool status: 11 BTC, 909,091 USDT.

CERTIK

Case Study: Sandwich attack

CERTIK

Case Study: Sandwich attack

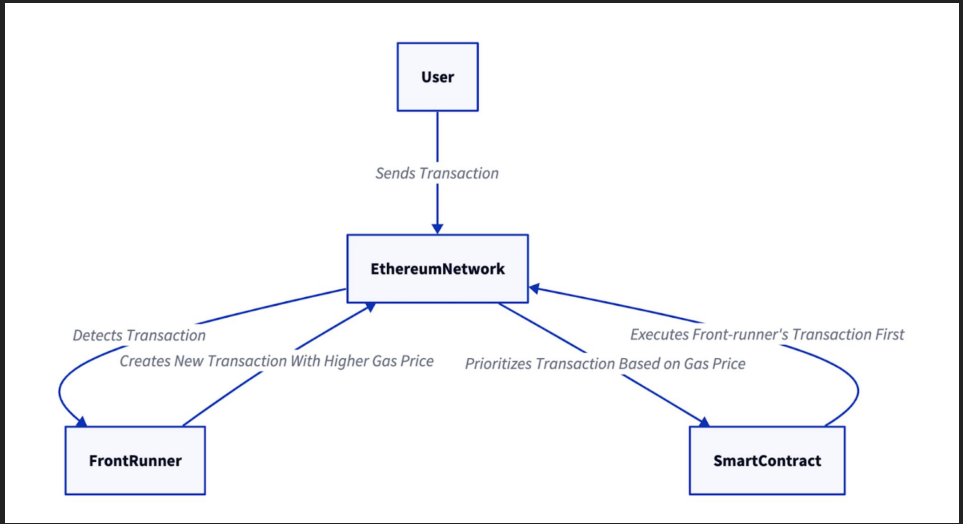
1. An attacker sells 100 BTC into the pool. The attacker will receive 909,091 USDT.
    a. Pool status: [110 BTC : 90,909 USDT]

2. Now, the user trades the 1 BTC in this pool. The user will receive 90,909 - (110 * 90,909 / 111) = **819 USDT**.
    a. Pool status: [111 BTC : 90,090 USDT]

3. The attacker uses all the 909,091 USDT previously obtained to buy Bitcoin. The attacker gains 100.992 BTC.
    a. Pool status: [10.008 BTC : 999,181 USDT]
    b. Attacker's profit: 100.992 - 100 = 0.992 BTC.

Case Study: Front-running

- If an attacker notices a profitable pending transaction in the mempool, they may be able to craft a transaction that performs the same action but pays a higher gas fee to front-run the victim's transaction.



CERTIK

Case Study: Front-running

- **Nomad Bridge Exploit**
  - In August 2022, the Nomad Bridge had a vulnerability that allowed attackers to bypass the message verification process.
  - Enabling anyone to make unauthorized withdrawals from the bridge contract.
  - Total lost: $190 million
- **A "not so smart attacker"**
  - The attacker crafted an exploit that only drained ~1% of the tokens from the bridge contract.
  - The exploit transaction was sent to the public mempool.
  - Front-running bots were able to identify this profitable trade and started sending out identical attack transactions.

CERTIK

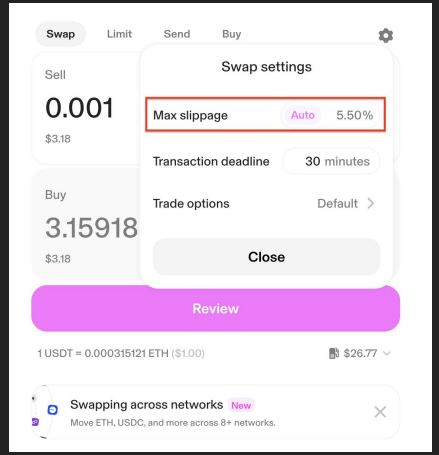
Case Study: Front-running

- **500+"Copycats" exploiters**
  - 14 addresses each stole over $2,000,000 USD
  - 22 addresses each stole between $1,000,000 and $2,000,000 USD
  - 281 addresses each stole between $100,000 and $1,000,000 USD
  - 215 addresses each stole between $1,000 and $100,000 USD

CERTIK

Best Practice and Defense

- **DeFi user**
  - Set an appropriate slippage rate when trading on a DEX or other DeFi platform.
    - If the trade result is less than a preset number, the trade will be canceled in the smart contract.
  - Split a large trade into smaller ones.
  - Trade with aggregators such as 1inch; it splits orders across multiple pools and routes them through the most efficient path.
  - Use a private mempool service for important transactions that might be front-run.



Best Practice and Defense

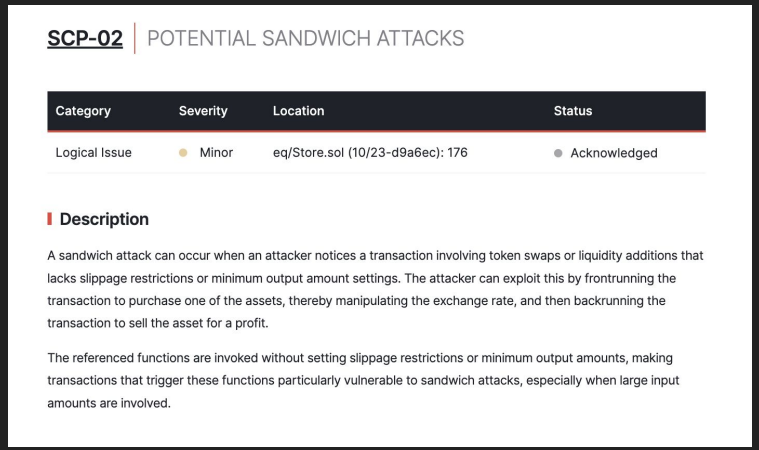
- **DeFi project, developer**
  - Implemented the "slippage" feature in the DEX's smart contract.
    - Warn users to set an appropriate slippage in the Dapp front end.
  - Avoid trading large amounts within the smart contract.
    - It's difficult to prevent sandwich attacks within the smart contract.
  - If a DeFi feature requires multiple transactions, implement it in a way that it cannot be attacked by front-running or back-running.

```
224         function swapExactTokensForTokens(
225             uint amountIn,
226             uint amountOutMin,
227             address[] calldata path,
228             address to,
229             uint deadline
230         ) external virtual override ensure(deadline) returns (uint[] memory amounts) {
231             amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
232             require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT')
233             TransferHelper.safeTransferFrom(
234                 path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path[1]), amounts[0]
235             );
236             _swap(amounts, path, to);
237         }
```

CERTIK

Best Practice and Defense

- **Security firm**
  - Look for code vulnerabilities that can be exploited to harm either the user or the DeFi project during smart contract audits.
  - Use a private mempool service for important transactions that might be front-run.
    - Whitehat rescue transaction



SCP-02 | POTENTIAL SANDWICH ATTACKS

Category	Severity	Location	Status
Logical Issue	Minor	eq/Store.sol (10/23-d9a6ec): 176	Acknowledged

Description

A sandwich attack can occur when an attacker notices a transaction involving token swaps or liquidity additions that lacks slippage restrictions or minimum output amount settings. The attacker can exploit this by frontrunning the transaction to purchase one of the assets, thereby manipulating the exchange rate, and then backrunning the transaction to sell the asset for a profit.

The referenced functions are invoked without setting slippage restrictions or minimum output amounts, making transactions that trigger these functions particularly vulnerable to sandwich attacks, especially when large input amounts are involved.


Phishing

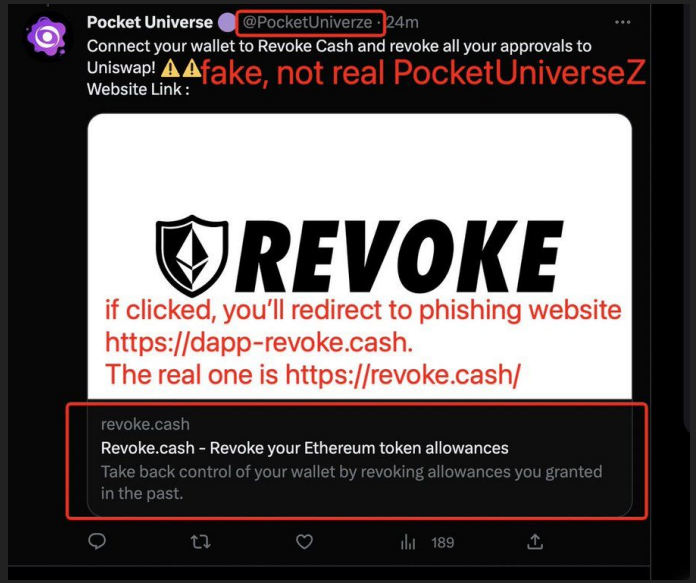
CERTIK

Web3 Phishing

- **Types of malicious activity that trick victims into performing actions that lead to token loss or wallet compromise.**
  - Direct token transfer or "approval"
  - Signature that can be used to obtain token transfer permission (e.g., "Permit")
  - Stealing of private key and seed phrase
- **Commonly used channel to execute phishing attacks**
  - Twitter Phishing Link
  - Phishing Website
  - Discord Scam Bots
  - Telegram Impersonation
  - Email Phishing Campaigns

Case study: Twitter Phishing Link

- It's common practice to remove token approval from an exploited smart contract.
- The exploited project often publishes a tweet warning users to revoke permissions.
- Attackers take advantage of the "post-incident" timing to deliver phishing attacks.



CERTIK

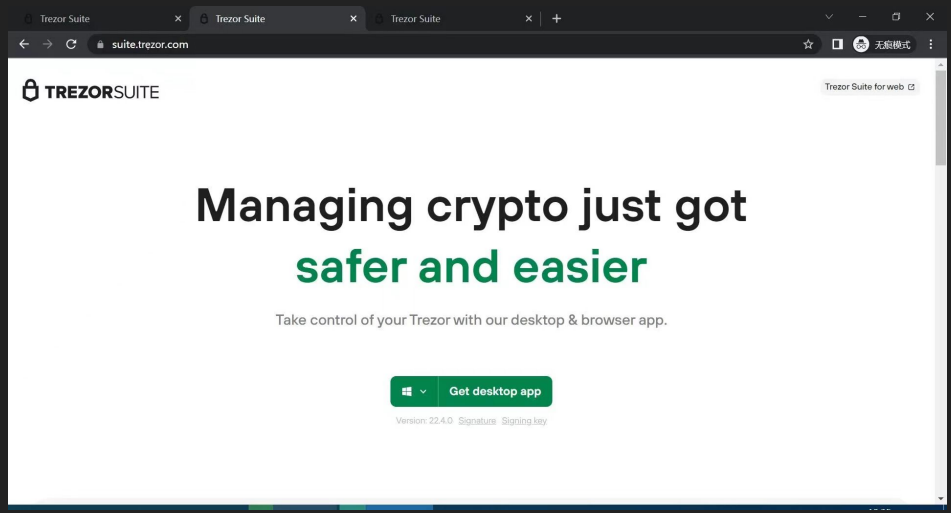
Case study: Twitter Phishing Link

- On July 15, 2020, a significant security breach occurred on Twitter, where hackers compromised 130 high-profile accounts,
- The attackers used these accounts to post fraudulent messages promoting a Bitcoin scam
- Enticing followers to send cryptocurrency to a specified address with the false promise of doubling their money.

CERTIK

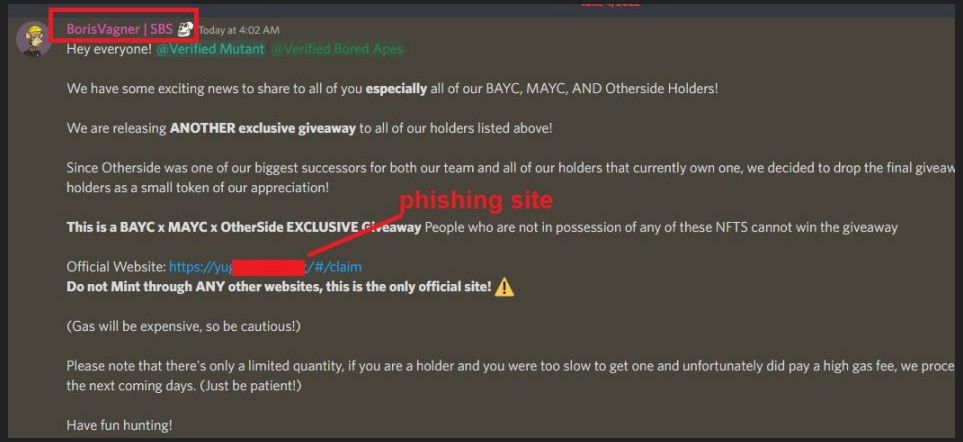
Case study: Phishing Website

- In April 2022, Trezor hardware wallet users received phishing emails from the unofficial domain "trezor.us" instead of the legitimate "trezor.io."
- The phishing emails directed users to a deceptive website with a URL like "https://suite.trẹzor.com," using a Unicode character ("ẹ") to mimic the legitimate site.
- Users who entered their seed phrases on the fake site had their wallets compromised, resulting in stolen assets.



Case study: Discord Scam Bots

- In June 2022, the Discord servers of popular NFT projects like Bored Ape Yacht Club (BAYC) were compromised.
- Attackers used bots to post phishing links promoting fake exclusive giveaways.
- Users who clicked these links and connected their wallets had their NFTs and tokens stolen.



Best Practice and Defense

- **DeFi user**
  - Do not share your private key or seed phrase with anyone.
  - Do not share your private key or seed phrase on any website.
  - Do not click on links from unknown sources.
  - Do not download and execute files from unknown sources.
  - Carefully review transaction data before signing with your wallet.
  - Download applications only from trusted sources, such as the iOS Store or Google Play Store.
  - Be cautious of anything that seems too good to be true.
  - Learn about new phishing attack vectors and avoid them.

Best Practice and Defense

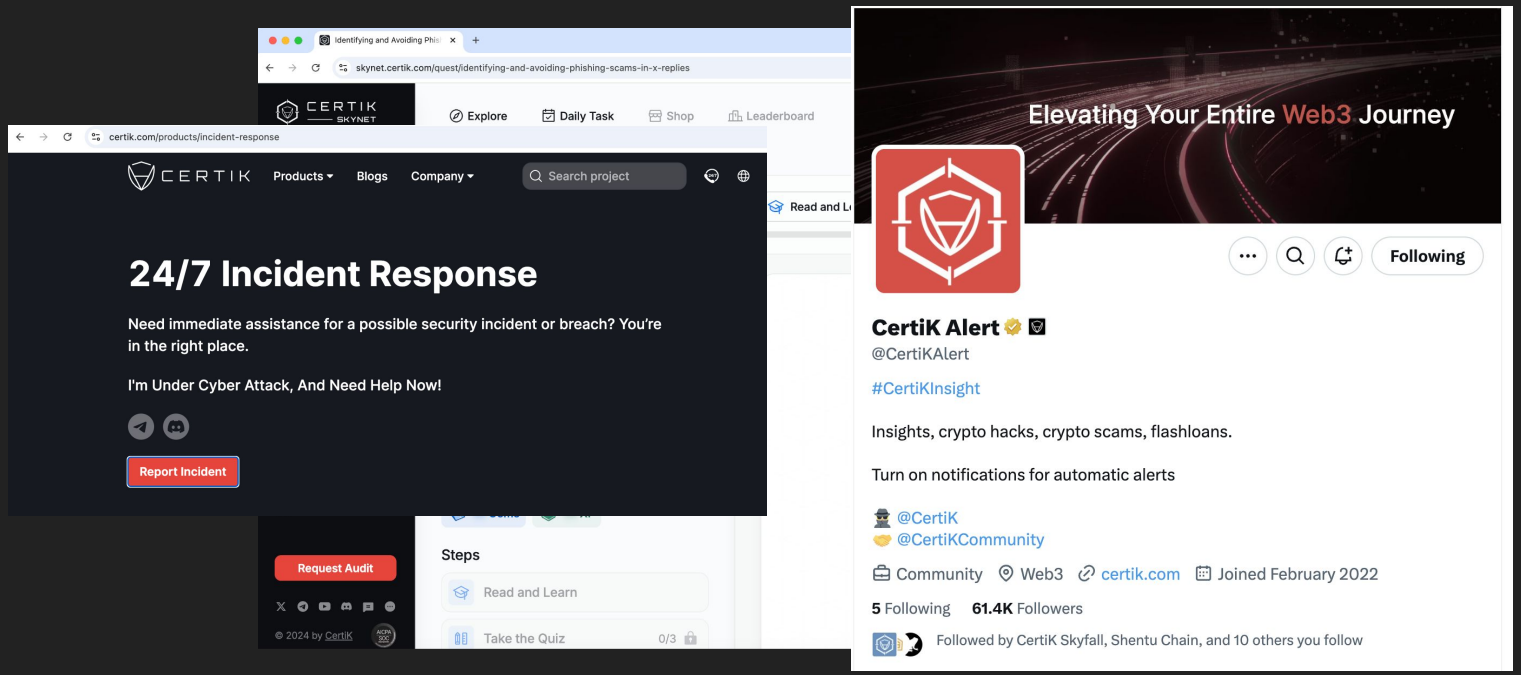
- **DeFi project, developer**
  - Items from the "DeFi" user section
  - Never initiate messages to any user on any communication platform (e.g., Telegram).
  - Routinely identify and remove malicious users from the community channel.
  - Use "The end of thread" on Twitter.
  - Warn users if any project channel (e.g., Twitter, Discord admin account) is compromised.
  - Frequently remind users not to fall victim to phishing attacks.

Best Practice and Defense

- **Security firm**
  - Monitor threats in the space and provide public real-time alerts.
  - Establish a phishing reporting channel.
  - Build tools for phishing detection or malicious transaction detection.
  - Publish educational materials for users to learn about recent phishing techniques.
  - Assist users and DeFi projects in tracking stolen funds.

CERTIK

Best Practice and Defense




Web3 Contract Exploit

CERTIK

Web3 Contract Exploit

- **Smart contracts refer to programs deployed and run on the blockchain.**
  - The code is often open-source and often visible on the blockchain.
  - Web3 contracts are immutable on most of blockchains by default after deployment.
  - Once a transaction is executed, it cannot be undone.
  - DeFi protocols manage massive capital, making them attractive targets for attackers seeking large payouts.
- **Smart contracts Exploits**
  - The attacker crafts a malicious transaction that exploits a vulnerability in the smart contract, with the goal, most of the time, of withdrawing crypto assets from the contract.
  - Occasionally, some critical vulnerabilities can stop a smart contract from functioning, putting it into a locked state that traps all the crypto assets inside the contract.

CERTIK

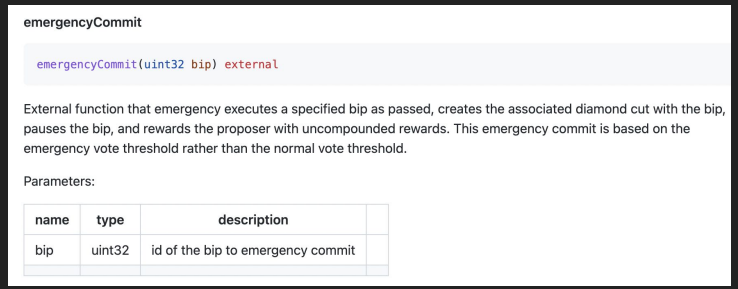
Web3 Contract Exploit

- **Common Web3 Contract Attack Vectors**
  - **Language Specific Risk**
    - Function Visibility, Compiler Version, Event, Low-level Call, Storage Risk etc
  - **Common Security Issues**
    - Input Validation, Reentrancy Attack, Access Control, etc
  - **Business Logic Design Flaw**
    - Abnormal Arbitrages, Inconsistent User Behavior results, funds being locked, malicious calls, etc
  - **Mathematical Operation Risk**
    - Precision Loss, Overflow/Underflow, Incorrect Math Calculation, etc
  - **Price/Balance Manipulation Risk**
    - Unexpected Price Update, Unintended Balance Change, etc
  - **Governance Risk**
    - Private Key Leakage, Voting Power Manipulation, etc
  - **Incentive Mechanisms Design Flaw**
    - Unfair Reward Distribution, etc
  - **Cross-Chain Risk**
    - Vulnerable Proof Verification, Replay Attack, etc)

CERTIK

Case Study: Governance Attack

- A flash loan exploit occurred on April 17, 2022, at Beanstalk Farms. Approximately $182 million was lost. The incident was due to a vulnerability in the governance mechanism, in which the attacker used a flash loan to amplify their voting power and ultimately control the result to pass a malicious proposal, draining the funds.
- **Attack Steps**
  - Create a malicious proposal.
  - Flashloan tokens to vote for this proposal.
  - Trigger emergencyCommit() to immediately execute the proposal.
  - Transfer assets in the contract to the attacker.



emergencyCommit

```
emergencyCommit(uint32 bip) external
```

External function that emergency executes a specified bip as passed, creates the associated diamond cut with the bip, pauses the bip, and rewards the proposer with uncompounded rewards. This emergency commit is based on the emergency vote threshold rather than the normal vote threshold.

Parameters:

name	type	description
bip	uint32	id of the bip to emergency commit

CERTIK

Case Study: Lack of Function Parameter Validation

- **TempleDAO Exploit, Oct 11, 2022**
  - The victim contract is a staking contract where users stake LP token and receive rewards
  - The "migrateStake()" function is intended to allow users to migrate their stake from an old staking contract to a new one
  - However, the `oldStaking` address is provided by the user, and the contract does not verify the user provided address
  - By providing a malicious `oldStaking` address, the function caller can receive staking credit for free

```
232     /**
233      * @notice For migrations to a new staking contract:
234      *         1. User/DApp checks if the user has a balance in the `oldStakingContract`
235      *         2. If yes, user calls this function `newStakingContract.migrateStake(oldStakingContract, balance)`
236      *         3. Staking balances are migrated to the new contract, user will start to earn rewards in the new contract.
237      *         4. Any claimable rewards in the old contract are sent directly to the user's wallet.
238      * @param oldStaking The old staking contract funds are being migrated from.
239      * @param amount The amount to migrate - generally this would be the staker's balance
240      */
241     function migrateStake(address oldStaking, uint256 amount) external {
242         StaxLPStaking(oldStaking).migrateWithdraw(msg.sender, amount);
243         _applyStake(msg.sender, amount);
244     }
```

Case Study: Reentrancy

- **HyperBear NFT, Feb 3, 2023**
  - The contract extends the ERC721 contract and allows users to mint the HyperBear NFT
  - The intended design is that a user can only mint once, and if addressMinted[msg.sender] is true, the canMint() function returns false
  - Note that addressMinted[msg.sender] is updated after the _safeMint() function call. Attacker was able to bypass the addressMinted flag check through Reentrancy.

```
128    function mintNFT(uint256 _numOfTokens, bytes memory _signature) public payable {
129        require(mintActive, 'Not active');
130        require(_numOfTokens <= mintLimit, "Can't mint more than limit per tx");
131        require(mintPrice.mul(_numOfTokens) <= msg.value, "Insufficient payable value");
132        require(totalSupply().add(_numOfTokens).add(partnerMintAmount) <= TOTAL_NFT, "Can't mint more than 10000");
133        (bool success, string memory reason) = canMint(msg.sender, _signature);
134        require(success, reason);
135
136        for(uint i = 0; i < _numOfTokens; i++) {
137            _safeMint(msg.sender, totalSupply() + 1);
138        }
139        addressMinted[msg.sender] = true;
140    }
```

Case Study: Reentrancy

- **HyperBear NFT, Feb 3, 2023 (cont'd)**
  - The _safeMint() function contains a _checkOnERC721Received() hook to the caller address, which allowed reentrancy to the mintNFT() function

```
258     function _safeMint(
259         address to,
260         uint256 tokenId,
261         bytes memory _data
262     ) internal virtual {
263         _mint(to, tokenId);
264         require(
265             _checkOnERC721Received(address(0), to, tokenId, _data),
266             "ERC721: transfer to non ERC721Receiver implementer"
267         );
268     }
369     function _checkOnERC721Received(
370         address from,
371         address to,
372         uint256 tokenId,
373         bytes memory _data
374     ) private returns (bool) {
375         if (to.isContract()) {
376             try IERC721Receiver(to).onERC721Received(_msgSender(), from, tokenId, _data) returns (bytes4 retval) {
377                 return retval == IERC721Receiver.onERC721Received.selector;
```

Best Practice and Defense

- **DeFi user**
  - Build basic security awareness
  - Always check if a project has been audited by a reputable security firm and review the audit findings to understand potential risks and mitigations. Avoid projects with no audit or unclear security assurances.
  - Follow official security channels and community alerts for the latest information on vulnerabilities or exploits. If a project is compromised, take immediate action to withdraw or secure your funds.
  - Be vigilant when signing approvals, especially with contracts that request high permissions. Limit approvals to necessary amounts and revoke unused permissions regularly to minimize risk.

Best Practice and Defense

- **DeFi project, developer**
  - **Follow coding best practices, such as**
    - Follow the "Checks-Effects-Interactions" Pattern
    - Apply nonReentrant modifier to critical functions that handle funds or state changes
    - Treat user-supplied addresses as potentially hostile, as they can include arbitrary code or logic. Always validate and restrict user-supplied address usage.
  - **Implement security checks specific to the business logic.**
    - Implement anti-flashloan mechanisms in governance by splitting the voting process across multiple blocks
    - Set Appropriate Voting Thresholds and Timelocks for Proposal
  - **Handle third-party dependency risks.**
    - Choose reliable, on-chain, multi-source price oracles such as Chainlink to secure accurate and robust pricing data. Avoid relying on a single AMM pair or low-liquidity source, as these are vulnerable to manipulation and may not reflect true market value

CERTIK

Best Practice and Defense

- **DeFi project, developer**
  - **Include comprehensive testing in the development lifecycle**
    - Unit testing
    - End-to-end testing
    - Fuzz testing
    - Invariant testing
  - **Provide security training on secure coding practices.**
  - **Ensure all code is audited before deploying on-chain and project launch.**
  - **Set up monitoring and an incident response plan to minimize losses.**
    - Automatically pause the contract to prevent further asset loss.
    - Warn users to withdraw assets from the affected contract.
    - Prepare compensation and recovery plan.

CERTIK

Best Practice and Defense

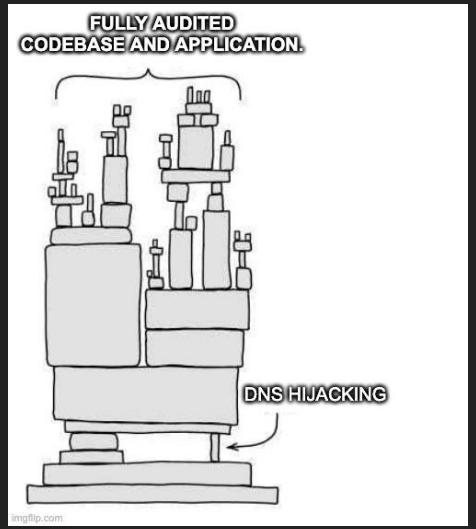
Best Practice and Defense

- **Security firm**
  - Provide comprehensive security audits for blockchain projects.
  - Develop detailed audit checklist to systematically identify potential attack vectors
  - Provide training and guidance on best security practices for developers to reduce the risk of insecure practices
  - Use fuzz testing, dynamic testing, and static analysis tools to uncover edge cases, simulate attacks, and reduce the likelihood of missed vulnerabilities during the audit process.
  - Establish alert channels to communicate security incidents or exploits to the community and investors promptly, providing users with real-time information to protect or recover their assets.
  - Develop a robust incident response process, including necessary whitehat rescue operation when appropriate to front-run or block attack transactions.


Cloud and Infrastructure Risks

Cloud and Infrastructure

- **The underlying technologies and services that support the deployment, hosting, and operation of the applications.**
- **Common Attack surface**
  - **Vulnerabilities in commonly used front-end stacks**
    - Next.js , Netlify, Vercel, Github pages
  - **DNS**
    - DNS/BGP hijacking, Sub-domain takeover
  - **Supply chain attack**
    - npm, yarn
  - **CDN**
    - Malicious JS file
  - **Misconfiguration in cloud service (e.g., S3 bucket)**
  - **Vulnerability in blockchain node server**



CERTIK

Case Study: DNS hijacking

- **DNS record**
  - Mapping of a domain name to an IP address
  - example.com <> 93.184.215.14
  - It's the address that the user's browser uses to load the website
- **Attack via DNS hijacking**
  - The attacker modifies the DNS record to redirect the user to a malicious site.
  - The malicious website contains JavaScript that tricks users into signing a malicious transaction to steal their assets.
  - Difficult for users to notice the anomaly.
- **Root cause**
  - The project's registrar account is compromised (e.g., GoDaddy, Cloudflare).
  - The DNS provider is compromised (e.g., Route 53, Squarespace).
  - Other undisclosed reasons.

Case Study: DNS hijacking

Post

dYdX
@dYdX

We just learned that dYdX v3 website (dYdX . exchange) has been compromised.

Please do not visit the website or click any links until further notice. An update will be provided when available.

This message does not relate to dYdX v4.

Last edited 12:07 PM · Jul 23, 2024 · **379.8K** Views

Compound Labs
@compoundfinance

🚨 **URGENT:** The Compound Labs website (compound[.]finance) has been compromised.

Please do not visit the website or clink any links until further notice. An update will be provided when available.

This is our final message // end of tweet. 🚨

5:15 AM · Jul 11, 2024 · **141.7K** Views

Curve Finance
@CurveFinance

Don't use curve.fi site - nameserver is compromised. Investigation is ongoing: likely the NS itself has a problem

2:20 AM · Aug 10, 2022 · Twitter Web App

TL;DR

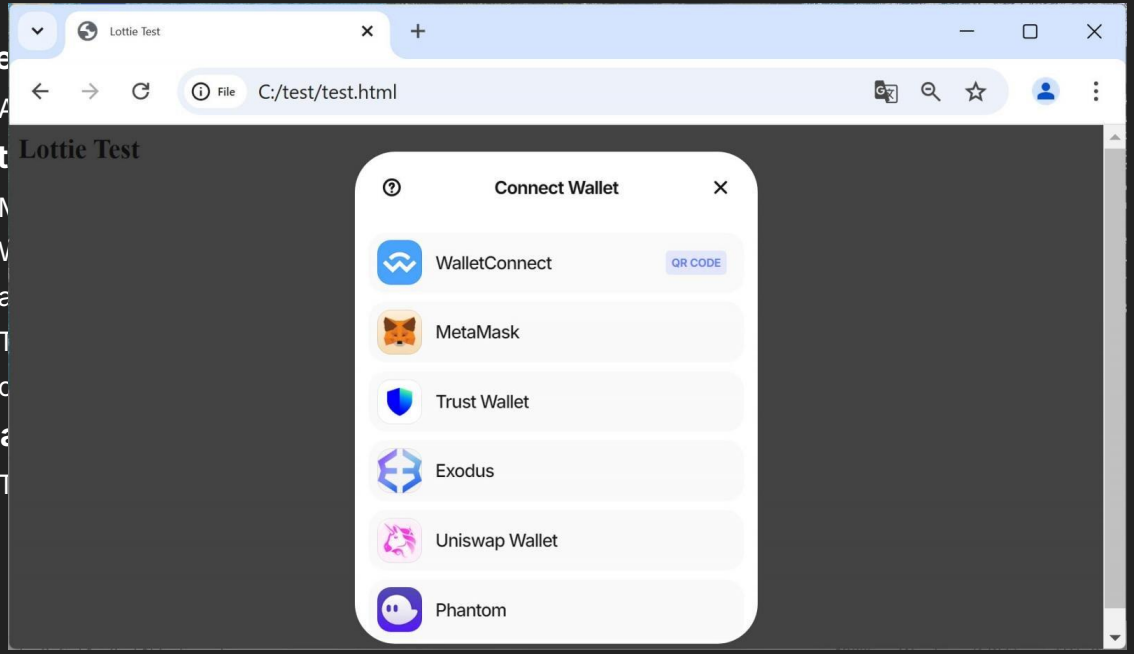
At approximately 4:20 PM EST Aug. 09 2022, Curve Finance(http://curve.fi)'s DNS record was compromised and pointed to a cloned malicious site. The attacker injected malicious code into that site that asked users to give token approvals to an unverified contract. In total, 7 users were affected by the exploit culminating in ~$612k losses.

Celer CelerNetwork
@CelerNetwork

📢(1/n)A DNS cache poisoning attack on cBridge's frontend UI appprox. during 08/17 07:45pm to 10:00 pm UTC caused some users to be redirected to malicious smart contracts that can drain all approved token amount. FIRST, PLEASE check&revoke any approval to the followings:

Case Study: Supply chain attack

- **@lottie**
  - A
- **Exploit**
  1. M
  2. W a
  3. T c
- **Root ca**
  - T

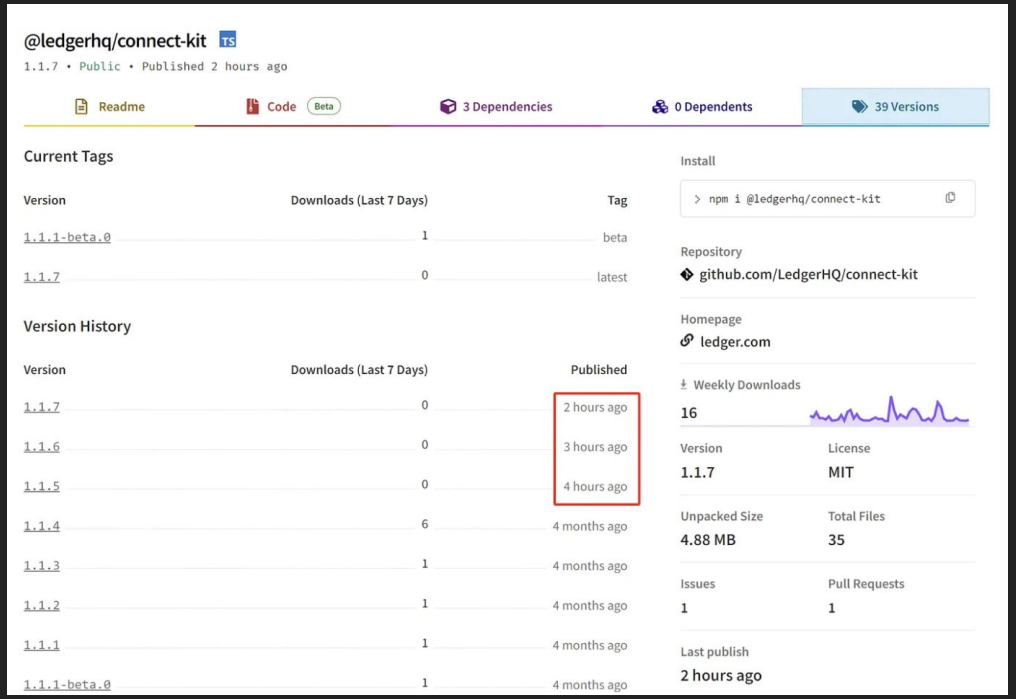


CERTIK

Case Study: Supply chain attack

- **Ledger Connect Kit**
  - open-source JavaScript library allowing developers to connect DApps to the Ledger hardware wallet
  - Used in almost all premium DeFi projects.
- **Exploitation**
  1. The attacker phished a former Ledger employee to leverage the individual's access on NPMJS.
  2. The API key was compromised instead of the NPM login account.
  3. The attacker injected malicious code into the library, tricking users into signing malicious transactions.
- **Root cause**
  - The token belonged to a former employee.
  - Access was not manually revoked on NPMJS during the employee offboarding process.
  - The employee fell victim to a phishing attack.

Case Study: Supply chain attack



@ledgerhq/connect-kit

1.1.7 • Public • Published 2 hours ago

Readme | Code (Beta) | 3 Dependencies | 0 Dependents | 39 Versions

Current Tags

Version	Downloads (Last 7 Days)	Tag
1.1.1-beta.0	1	beta
1.1.7	0	latest

Version History

Version	Downloads (Last 7 Days)	Published
1.1.7	0	2 hours ago
1.1.6	0	3 hours ago
1.1.5	0	4 hours ago
1.1.4	6	4 months ago
1.1.3	1	4 months ago
1.1.2	1	4 months ago
1.1.1	1	4 months ago
1.1.1-beta.0	1	4 months ago

Install

```
> npm i @ledgerhq/connect-kit
```

Repository

github.com/LedgerHQ/connect-kit

Homepage

ledger.com

Weekly Downloads

16

Version	License
1.1.7	MIT

Unpacked Size	Total Files
4.88 MB	35

Issues	Pull Requests
1	1

Last publish

2 hours ago

MetaMask 🦊👆
@MetaMask

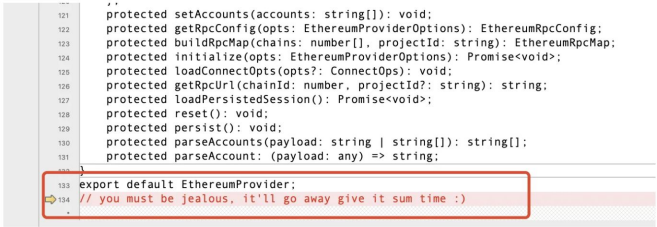
🚨Ledger users: @blockaid_ has identified an attack on Ledger Connect Kit. Please stop using dapps.

> **Blockaid** @blockaid_ · 5h
> 🚨 We've detected a potential supply chain attack on ledgerconnect kit 🚨
> The attacker injected a wallet draining payload into the popular NPM package. This currently affects a couple of popular dapps including but not limited to hey.xyz

Case Study: Supply chain attack

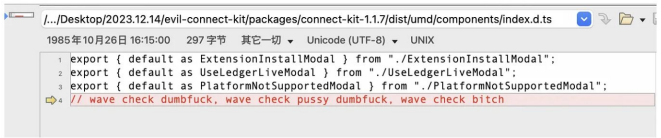
The attacker left some message in the code and implanted malicious JS code.

@ledgerhq/connect-kit 1.1.6: The attacker left a message in the code and implanted malicious JS code.



```
    protected setAccounts(accounts: string[]): void;
    protected getRpcConfig(opts: EthereumProviderOptions): EthereumRpcConfig;
    protected buildRpcMap(chains: number[], projectId: string): EthereumRpcMap;
    protected initialize(opts: EthereumProviderOptions): Promise<void>;
    protected loadConnectOpts(opts?: ConnectOps): void;
    protected getRpcUrl(chainId: number, projectId?: string): string;
    protected loadPersistedSession(): Promise<void>;
    protected reset(): void;
    protected persist(): void;
    protected parseAccounts(payload: string | string[]): string[];
    protected parseAccount: (payload: any) => string;
}
export default EthereumProvider;
// you must be jealous, it'll go away give it sum time :)
```

@ledgerhq/connect-kit 1.1.7: The attacker left a message in the code and implanted malicious JS code.



```
/.../Desktop/2023.12.14/evil-connect-kit/packages/connect-kit-1.1.7/dist/umd/components/index.d.ts
1985年10月26日 16:15:00  297 字节  其它一切  Unicode (UTF-8)  UNIX
export { default as ExtensionInstallModal } from "./ExtensionInstallModal";
export { default as UseLedgerLiveModal } from "./UseLedgerLiveModal";
export { default as PlatformNotSupportedModal } from "./PlatformNotSupportedModal";
// wave check dumbfuck, wave check pussy dumbfuck, wave check bitch
```

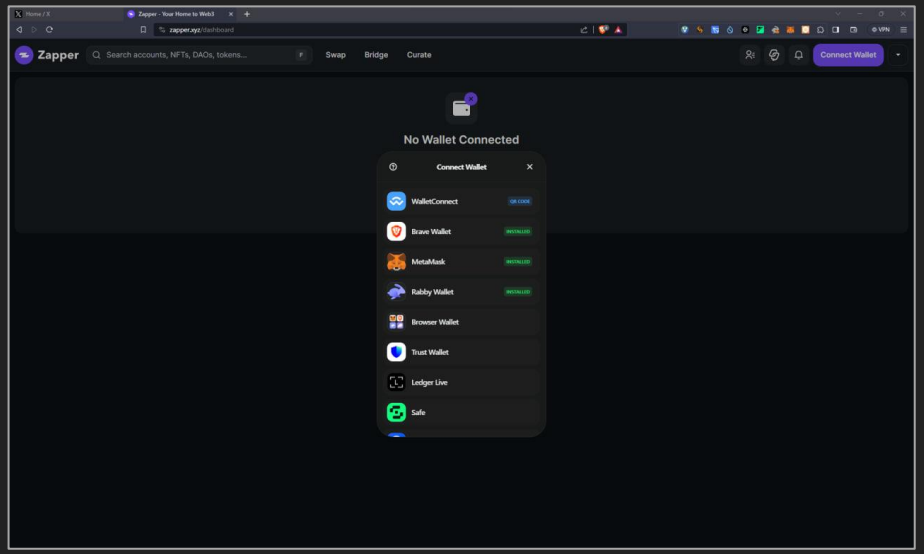
CERTIK

Case Study: Blockchain node server RCE(remote code execution)

- **InfStones**
  - Blockchain node/validator infrastructure provider.
  - Launch a blockchain node with just a few clicks.
  - A private key stored on or connected to the server is required to perform block validation.
- **Vulnerability and exploitation**
  - A network service called "Tailon" for reading log files is running on TCP port 55555.
  - The exposed service can be used to run system commands on the server.
  - The researcher found that over 400 nodes owned by the company are vulnerable.
- **Impact**
  - Enables theft of assets from the validator nodes.
  - Can cause the nodes to get "slashed."

Best Practice and Defense

- **DeFi user**
  - If the user notice anything suspicious, stop interacting with the application.
    - Malicious transaction or message signing request from a DNS-hijacked website.



Best Practice and Defense

- **DeFi project, developer**
  - Use a reputable DNS registrar.
  - Employ strong protection over the account that can modify infrastructure configurations (e.g., DNS, software packages).
  - Implement strict Content Security Policies to mitigate risks from malicious JavaScript files served via CDNs.
  - Using a pinned version in software packages*
  - Provide developer training on DevOpsSec best practices and strictly adhere to them.
  - If anything is compromised, immediately notify the community to avoid interaction with the compromised component.
  - Set up a comprehensive incident response plan to quickly resolve issues if exploited.
  - Conduct regular phishing awareness training.
  - Hire a security firm to conduct an infrastructure security assessment.

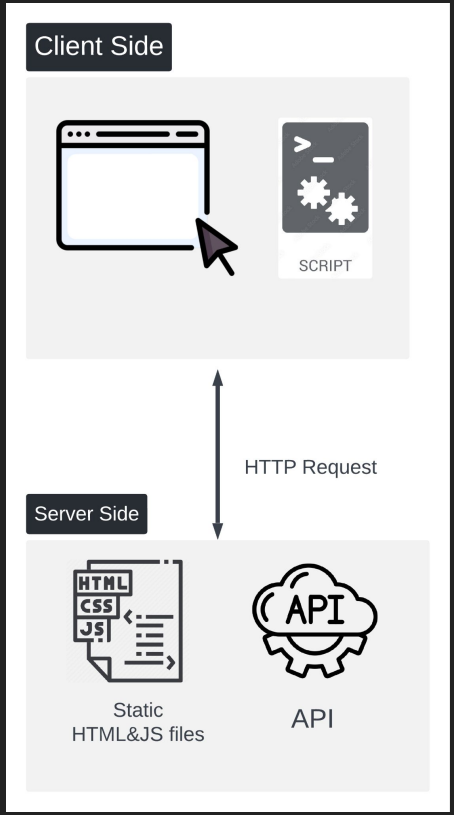
Best Practice and Defense

- **Security firm**
  - Establish a reporting channel for users to submit incidents.
  - Monitor threats in the space and provide public real-time alerts.
    - Compromised website
    - Compromised dependency
    - Reach out to clients to deliver warnings and offer assistance.
  - Perform security audits of cloud infrastructure to help clients identify and resolve vulnerabilities and weakness.
  - Provide training and guidance on best security practices for developers to reduce the risk of human error and insecure practices.
  - Provide phishing awareness training.


Web2 Application Security Risk

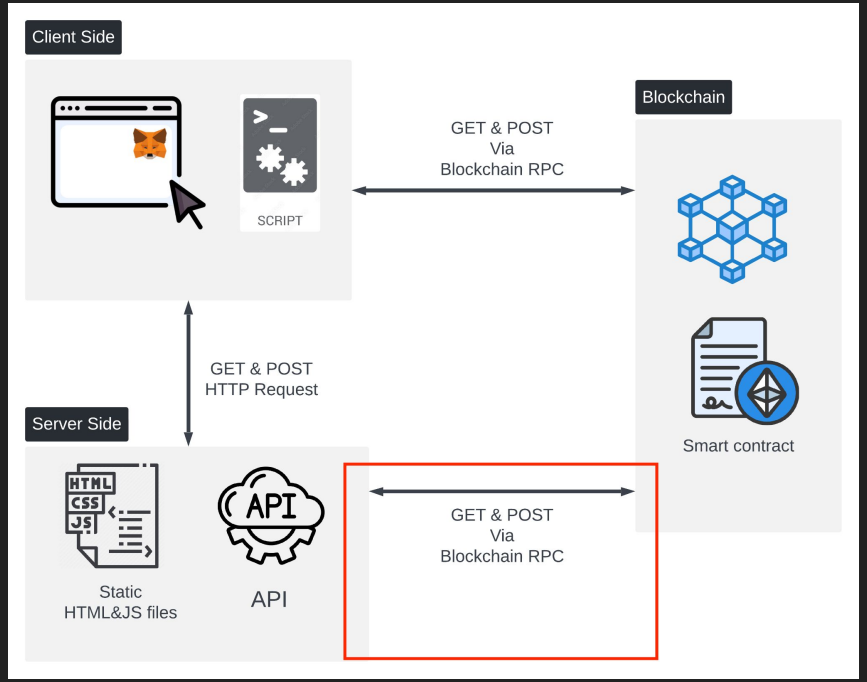
Standard Web2 Application

- **Client-side application**
- **Server-side components**
  - Web server & APIs
  - Database
  - etc



Web3 Dapp Architecture

- **Client-side application**
  - An essential UI for users to interact with the application.
  - Read data from the blockchain (smart contract).
  - Construct a transaction with user input for the wallet to sign.
  - The wallet send the signed tx to the blockchain.
- **Server-side components**
  - Statis HTML/JS files, Server API
  - The server communicates with the blockchain to read and write(via transactions) data
  - The private key of privileged roles (e.g., owner/admin) of the smart contract is stored on the server.



CERTIK

Common AppSec-heavy Web3 product categories

- **Cryptocurrency Exchange(CEX)**
  - Centralized platform where users can buy, sell, and trade cryptocurrencies.
- **Data platform**
  - An application for users to view transaction information, token flows, analytical data, etc. Examples include Etherscan, Arkham, and Dune.
- **Wallet**
  - An application that holds the private key and uses it to perform signing when requested. Wallets can come in various forms such as browser extensions, mobile apps, and web apps.
- **Dapps**
  - A wide range of applications include functionality implemented through the project's on-chain smart contracts.

CERTIK

Web2 Client Side Attack

- **Initial malicious transaction**
  - XSS
  - Subdomain takeover
  - DNS hijacking
  - Supply chain attacks
- **Deceive users into signing malicious transactions with UI element tampering.**
  - HTML Injection
  - Clickjacking
- **Redirection to a phishing site**
  - Open redirect

CERTIK

Web2 Server Side Attack

- **Stealing keys from the server**
  - RCE, SQLi, LFI, Directory Traversal, SSRF, Debug page, Server misconfiguration, etc
- **Steal crypto assets from other user accounts or via admin features**
  - Broken access control
  - Privilege escalation
- **Spend/own assets more than the user's actual balance**
  - Business logic error
  - Race condition
- **Take down the server that hosts NFT images and metadata.**
  - Denial of service

CERTIK

Case study: Business Logic Error

- **Business logic error in Coinbases trading API**
  - Allow any users to sell any coin at coinbase with the price of Bitcoin
- The vulnerability was submitted by a whitehat to Coinbase's bug bounty program
  - 250,000$ bounty award

CERTIK

Case study: Denial of Service

- **Finding**
  - For blockchains with long block times (e.g., 12 seconds for Ethereum), the HTTP request and connection will remain open for a considerable amount of time.
  - **The API server can be easily subjected to a DoS attack with concurrent requests sent from a single machine.**

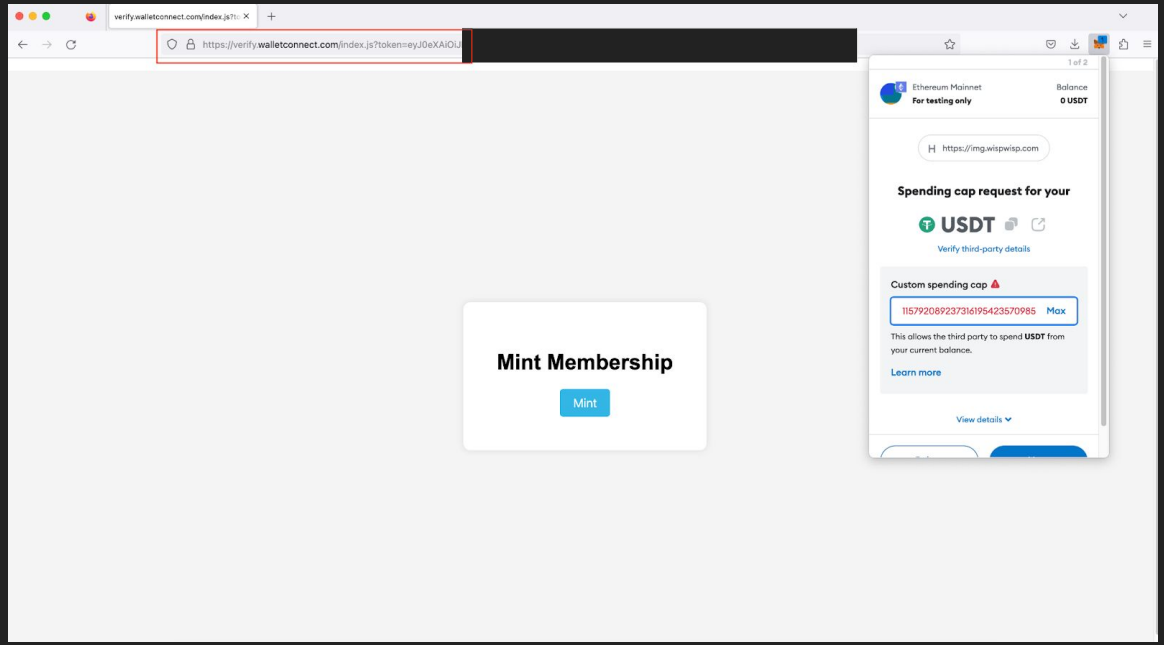
```
app.post('/deposit', async (req, res) => {
        // Instantiate the contract within the function
        const contract = new ethers.Contract(contractAddress, contractABI, wallet);

        // Sending the transaction to call the deposit function
        const txResponse = await contract.deposit();
        console.log(`Transaction sent: ${txResponse.hash}`);

        // Waiting for the transaction to be mined
        const receipt = await txResponse.wait();
        console.log(`Transaction mined: ${receipt.transactionHash}`);

        // Sending the response back to the user
        res.json({status: 'success', transactionHash: receipt.transactionHash, receipt:
        });
});
```

Case study: Client side XSS Attack

- **Cross-Site Scripting Attack in WalletConnect**
  - WalletConnect is one of the most popular tools that connect Dapps with users' wallets.
  - An XSS vulnerability was discovered in one of its APIs.
  - The vulnerability can be used to perform phishing attacks against users, exploiting the trust of the WalletConnect domain.
  - CertiK reported the vulnerability and received a bounty from the team.

Case study: Client side XSS Attack

- Cross-Site Scripting Attack in WalletConnect



The state of Dapp security

- With the advancement of the blockchain ecosystem, Dapps are becoming increasingly complex and are starting to integrate with heavy Web2 backend components.
- Often, vulnerabilities are introduced in Dapps because developers may lack knowledge of either Web2 or Web3 security.
- Web2 vulnerabilities in Web3 applications often have increased severity and higher impact when exploited. Nothing is more impactful than the loss of millions of $$$$.
    - Unlike a smart contract exploit, it's publicly viewable. The exploitability of the backend may have far less exposure, but it still happened.
- **To secure a Dapps, especially a Dapp with heavy backend components, it is as important to get a penetration test as it is to audit the smart contract.**

Best Practice and Defense

- **DeFi user**
  - For server-side attacks, there isn’t much users can do.
  - For client-side attacks, if the user notice anything suspicious, stop interacting with the application.
  - Avoid low-quality DeFi projects.

Best Practice and Defense

- **DeFi project, developer**
  - Provide developer training on Web2 security vulnerability topics.
  - Obtain Web2 penetration testing from a security firm.
  - Set up a bug bounty for whitehats to report security vulnerability
- **Security firm**
  - Offer penetration testing and security consulting services to DeFi projects.
  - Monitor threats within the ecosystem and alert the community to avoid compromised projects.
  - Conduct security research to understand the latest Dapp development trends and their attack surfaces, to better protect clients.

CertiK Penetration Testing Service Offering

Blackbox Security

- **Approach:** Assessment conducted externally
- **Requirements:** No need for source code; test account access recommended.
- **Focus:** Web portal, Mobile DApp, API.
- **Features:** External attacker simulation, cost-effective

Whitebox Security

- **Approach:** Source level code auditing
- **Requirements:** Source code needed
- **Focus**: Web, backend Infra, Mobile DApp, API, SDK
- **Features:** Comprehensive and high coverage, line-by-line review to uncover hidden bugs

Apply Standards

- OWASP Web
- OWASP MASVS + MASTG
- NIST
- PTES
- CVSS

CERTIFICATIONS

OSWE · OSCE · OSCP · aws certified

Tailor Web3 Pentest Solutions For

Crypto Exchange

Exchange pentest that covers beyond Web2 vulnerabilities.

- Infrastructure
- Front-end/client side security
- Server side APIs
- Data protection

Cover Exchange Specific Logics

- ✔ "Fake" deposits
- ✔ KYC/Onboarding process
- ✔ Buying with fiat
- ✔ Asset exchange: Manipulation risks
- ✔ Trading: Price and date tricks
- ✔ Loans: Valuation issues

BingX · CoinW · LATOKEN · OKX

Wallet

Comprehensive wallet security audit with a focus on key safety.

Full Wallet Platforms

- ✔ Web & Mobile
- ✔ Chrome Extension
- ✔ Infrastructure
- ✔ Desktop Application

MPC Wallet Assessment

- ✔ Multi-party Interaction
- ✔ Network Communication
- ✔ Backup Security
- ✔ Share generation & storage
- ✔ MPC Cryptography Impl.

Cover Wallet Risks

- ✔ Randomness
- ✔ Key Handling
- ✔ Use of Cryptography
- ✔ Key Generation & Storage
- ✔ User Protections
- ✔ Transaction Integrity

ZenGo · TRUST · WalletConnect · SAFEHERON · OKX WALLET · BYBIT

DApp

Advanced Web3 DApp Penetration Testing

Specialized pentest expertises with deep knowledge in blockchain and smart contracts, understanding unique Web3 attack vectors.

- **Cross Chain Bridge**
  - Deposit/Withdraw Logic
  - On-chain Event Processing
- **GameFi**
  - Game Cheating/Anti-Scripting Measures
- **Marketplace/DEX/Lending**
  - Client-side vulnerabilities(e.g. XSS)
  - Trading logic assessment

WEMIX · hatom · QTUM · GALA GAMES

Blockchain

Comprehensive Blockchain Dynamic Testing: Elevating Security Beyond Source Code Review

- L1/L2 Blockchain
- Validators Notes
- RPC Endpoints
- Blockchain Network

Assessment Types

- ✔ Network Penetration Testing
- ✔ Architecture Review
- ✔ Cloud Infrastructure Security Review
- ✔ Performance Testing

GALA GAMES · TON

CERTIK

Question?


Securing The Web3 World
CERTIK
Protect your community and your organization today.
Visit CertiK.com Get in touch at bd@certik.com