MERKLE SCIENCE

HACKHUB 2022

Insights into hacks and exploits of 2022, the vulnerabilities that led to them, and key mitigation strategies

Table of Contents

This guide is an educational venture, aimed at providing an overview of hacks and exploits in the Web3 landscape. This guide does not constitute or replace legal, financial, or investment advice. We encourage you to seek to consult relevant advisors before making these decisions. Any examples or checklists in this guide are purely illustrations; they are not exhaustive and are not intended to impose or imply particular rules or requirements.

Every effort is made to provide accurate and complete information. However, we cannot guarantee that there will be no errors. We make no claims, promises, or guarantees about the accuracy, completeness, or adequacy of the content in this guide and will not be held liable for errors, omissions or inaccuracies whatsoever. This guide is an educational venture, aimed at providing an overview of hacks and exploits in the Web3 landscape. This guide does not constitute or replace legal, financial, or investment advice. We encourage you to seek to consult relevant advisors before making these decisions.

Table of Acronyms

2FA - 2-Factor Authentication

API - Application Programming Interface

DASP - Decentralised Application Security Project

DeFi - Decentralised Finance

DEX - Decentralised Exchange

IP Address - Internet Protocol Address

POTY - Person of the Year

SEC - The Securities and Exchange Commission

SWC - Smart Contract Weakness Classification

Executive Summary

If cryptocurrency had a person of the year like TIME, 2022's would very well be the black-hat hacker. These cyber criminals defined 2022 with an unrelenting barrage of hacks, attacks, and exploits. These incidents were so ubiquitous that it became easy for people to tune out, hardened by news fatigue.

Because this indifference can affect everyone - including the stakeholders in the crypto community who should care the most - our team at Merkle Science decided to produce this Hackhub report. Rather than only detail different hacks over the past year, we made it a point to analyze the trends that were evident across them, so the industry could learn from its mistakes, stop repeating them and avoid losing the forest for the trees. No single hack - not even one as large as Ronin - deserves special evaluation. Our focus is on the vulnerabilities that keep popping up – these are the weak points the industry must shore up for 2023 and beyond.

To aid in this goal, our report serves to itemize the details of each hack (i.e. funds lost, type of attack, type of target, and so forth). Because it's important to standardize categorization - we cannot get anywhere as a community if we are using different working definitions of hacks - we referred only to the DASP 10 and SWC. DASP 10 and SWC Registry provide categorization and explanation of key smart contract vulnerabilities. Then we analyzed this data with proprietary dashboards. Many of the subsequent insights that we uncovered surprised even us.

Let's start with the superlatives. The most common attack vector were reentrancy vulnerabilities, with more than 48 different incidents throughout 2022. They were also the most diverse in terms of target selection (how very inclusive of them!), attacking aggregators, DeFi projects, marketplaces and gaming platforms.

While re-entrancy attacks were the most common, the most expensive attack vectors were access controls which accounted for US$1.5 billion in lost funds, more than five other hack types combined.

This level of damage is due to the nature of the access control. When attackers violate an access control, they are able to wield higher level privileges, such as the ability to mint tokens, or obtain consensus, such as the required number of signatures for a multisig.

Across all the hacks, the most common target were bridges, which suffered from over 30 separate attacks. Crypto bridges are a natural target for two reasons:

1. Because they link one blockchain to another, they are commonly transferring valuable assets, such as cryptocurrency or tokens;
2. Because crypto bridges are complex structures, there are often technical weak points on them that the developers may not be aware of but hackers are.

The weakness of bridges also extends to platforms. Within DeFi, which were the most hacked protocols, cross-chain bridges were the most commonly hacked. This is a trend that began in 2021 and continued, if not intensified, in 2022. Going forward, bridges will need to be strengthened and defended with the same zeal as the blockchains they connect.

While 2022 saw some feel good stories, such as white hat hackers who returned US$32.6 million to cross-chain bridge Nomad in the wake of a US$190 million hack, security should not hinge on miraculous recoveries. The industry needs more fundamental changes in how it protects bridges and other elements in the crypto ecosystem. These actions are as much cultural as they are technical.

For one, industry players must not treat cybersecurity as a token exercise that can be accomplished on a one-and-done basis. There should be recurring audits of code and systems, particularly when any new developments are released, as those also often introduce vulnerabilities and weak points. To ensure this occurs, organizations must shift away from the “move fast and break things” ethos that characterizes wider tech: Such an attitude may work when you’re building a shiny app, but not when you interface with people’s hard-earned money - even if it’s in the form of crypto.

This stakeholder-centric approach may seem obvious, but it’s been flagrantly absent in many of the busts of the last year. Industry players must act as better stewards of user funds and data by emphasizing that security is an organization-wide mandate and educating all staff on how to act in turn. Some of these best practices may be common sense, but common sense is not always common (we’re looking at you, Sky Mavis dev, who opened the spear phishing email from North Korea). Industry players, in short, must lead all business initiatives with a security-first philosophy.

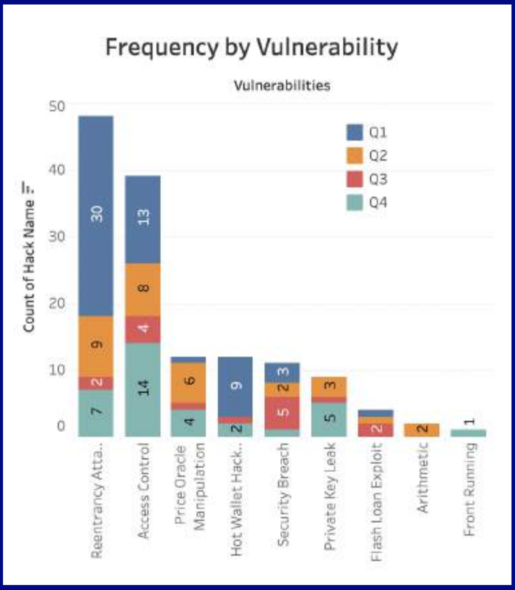
There are some bright points in 2022 that suggest these shifts can start to manifest in 2023. The most notable was the fact that hot wallet hacking fell precipitously over the course of the year. In January, there were 8 incidents of hot wallet hacking, zero from March to September, and a small bump in October and November. This drop can be attributed to exchanges adopting a zero-trust environment, wherein funds are placed in cold storage guarded by two- or even multi-factor authentication. The success of these security measures points to the fact that the industry can curb attacks through the standardization of best practices.

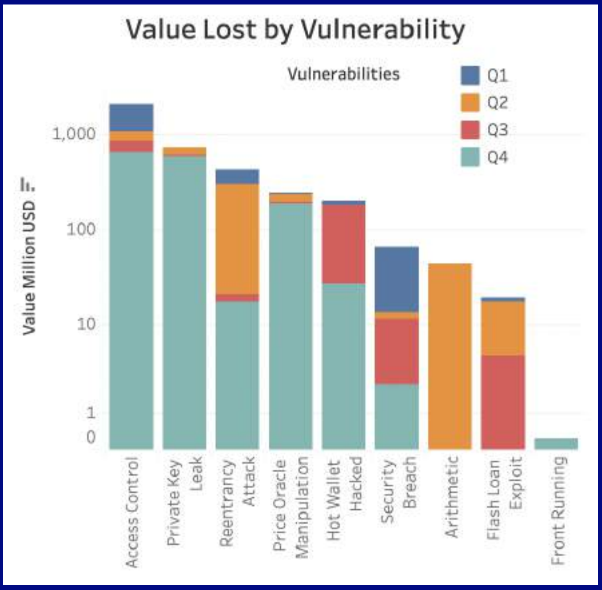
Unfortunately, as the industry improves its technical sophistication, hackers do, too. The professionalization of black-hat hackers is driven by the fact these are no longer lone wolves tricking people for a few tokens. These are criminal enterprises, often state-sponsored as we saw in the rise of the Lazarus Group. Backed by North Korea, the Lazarus Group was responsible for a variety of major hacks in 2022, such as the Ronin hack for US$600 million.

Leaders in the crypto community should view shadowy threats like the Lazarus Group as a challenge: Black-hat hackers are coming with their best, so we must strive to remain a step ahead. Only by anticipating where hackers will again attack - basing our judgment in part on their moves in 2022 - can we properly fortify the cryptocurrency ecosystem that we all call home.

This report sheds light on DeFi criminal activity in the blockchain industry in 2022, primarily involving smart contract exploits, security breaches, attacks on exchanges, NFT platforms, cross-chain bridges and wallets. Additionally, this report includes the flow of stolen funds from some of the major hacks of 2022, the vulnerabilities that were exploited, and some mitigation strategies that could have prevented the loss of billions of dollars of users’ funds to illicit actors.

Hackhub Quarterly Insights





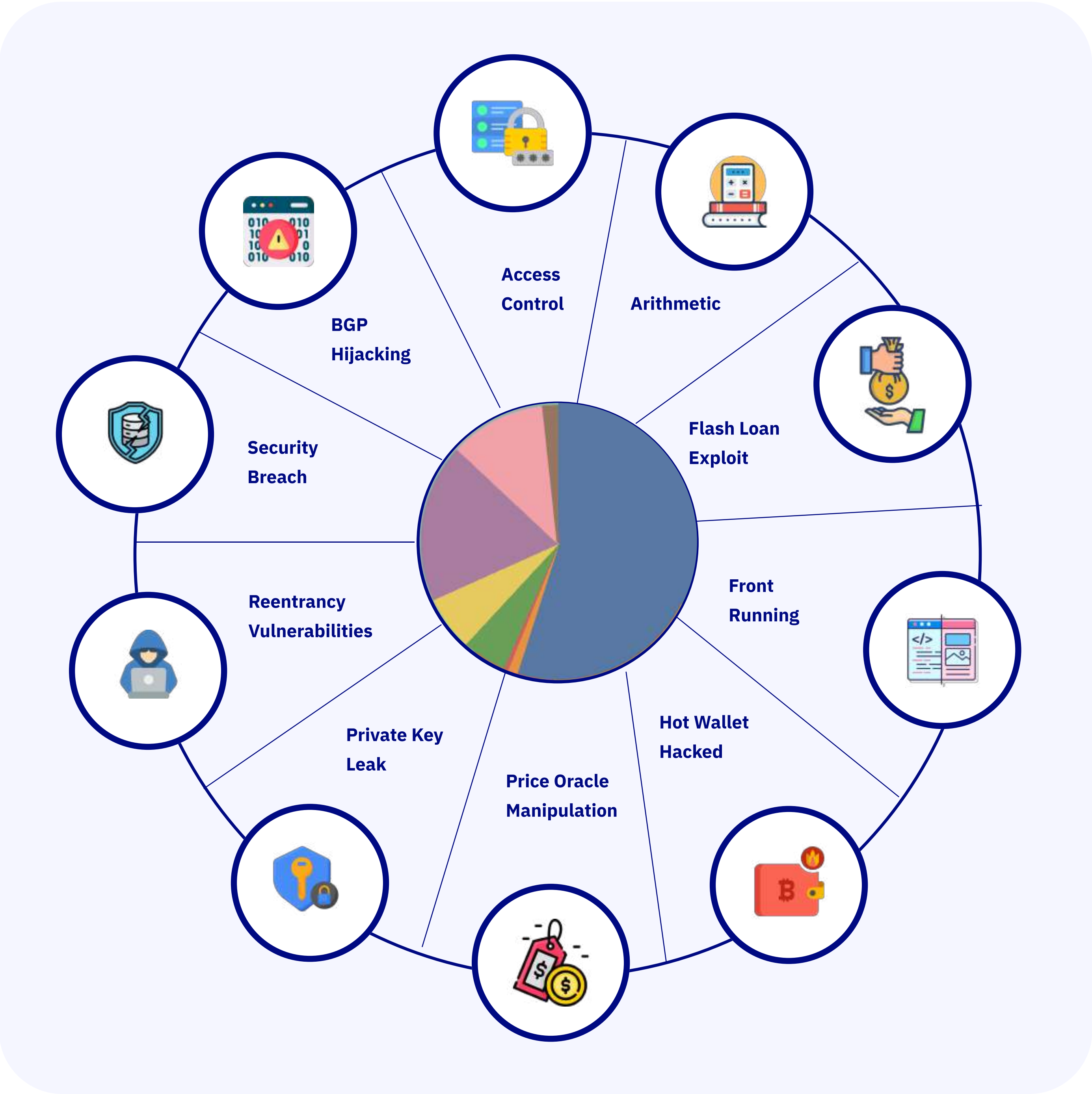
Q1 Insights
Blockchain vulnerability exploits led to a loss of over $1.2 billion in Q1 2022. The March 2022 Ronin hack, which saw the exploiters swindle $625 million, was the worst of the crypto hacks to date - beating out the $610 million Poly Network hack, that took place in August of 2021. Q1 amounted to almost 30% of the total amount lost in 2022. In Q1, smart contract exploits and security breaches were the preferred hacking techniques. Smart contract exploits accounted for 94% of attack strategies employed by hackers. While Ronin, Wormhole, IRA and Cashio were some of the biggest hacks due to access control vulnerabilities, Qubit was examples of reentrancy attacks that have accounted for the massive losses this year.

Q2 Insights
Hacks and exploits led to a loss of more than $709 million in Q2. Major hacking techniques used include smart contract exploits, security breaches, and price oracle manipulation. Approximately 76% of the funds were stolen due to smart contracts vulnerabilities like Reentrancy gaps, Arithmetic issues, and Access Control vulnerabilities. Hacks on DeFi protocols were more common in Q2, unlike Q1 in which cross-chain bridges were the centre of all hacks. Some of the biggest hacks of the quarter include attacks on Beanstalk Farms and Harmony Bridge that together led to a loss of $282 million.

Q3 Insights
We have seen a loss of $383 million in Q3 2022. Major hacking techniques include smart contract exploits, flash loan attacks and security breaches. Most funds were lost due to reentrancy vulnerabilities in smart contracts and attacks on insecure hot wallets. Some of the biggest hacks of the quarter include the Nomad and Wintermute hacks which have been listed as one of the top ten biggest hacks of the year. Platforms hacked in this quarter include cross-chain bridges, lending protocols and DeFi services.

Q4 Insights
The total losses in Q4 reached $1.5 in 2022, which was the highest among the four quarters. Major attacks took place due to private key leaks, smart contract vulnerabilities, and hot wallet attacks. Unlike other quarters, the leading cause for attacks were non-smart contract vulnerabilities. Some of the biggest hacks of the quarter include the FTX hack, the BSC Token Hub exploit, and the Mango Markets Exploit. The most frequently hacked platforms include decentralized exchanges, bridges and DeFi services.

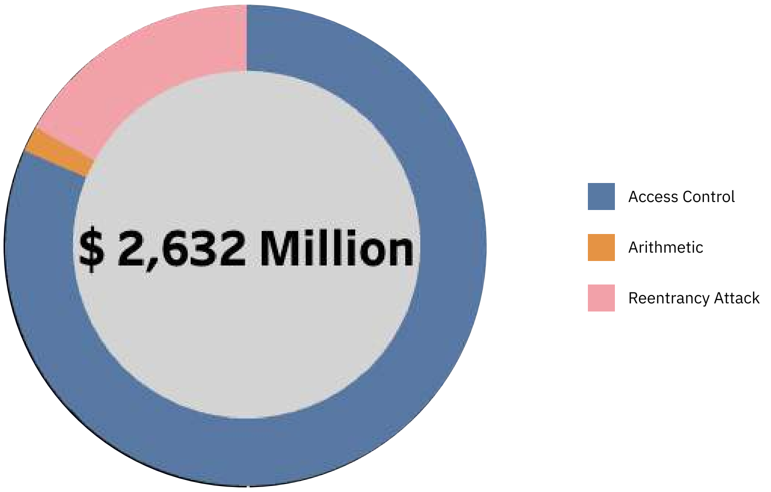
Hacks in 2022:
Causes, Cases, & Mitigation Techniques



In the forthcoming chapter we analyse top vulnerabilities and hacks of 2022

- 🟦 Access Control
- 🟫 Other Security Breaches
- ■ Reentrancy Attack
- 🟪 Private Key Leak
- 🟨 Price Oracle Manipulation
- 🟩 Hot Wallet Attacks
- 🟧 Arithmetic

Hacks due to Smart Contract Vulnerabilities



The largest hacks of 2022 had a common vector: vulnerabilities in smart contracts. Smart contracts are self-executing processes in which the terms of the agreement between the concerned parties are written in a set of codes and recorded on the blockchain. The code is powered by a blockchain platform, such as Ethereum, and is publicly viewable which provides much-needed trust and transparency. One key benefit of smart contracts is they do not require any human or organizational intervention to verify and enforce the conditions of the agreement.

Another key advantage of smart contracts is their immutability. Once deployed to the blockchain, smart contracts cannot be altered - the underlying code is secured using cryptographic techniques, making it extremely difficult for anyone to change the code without detection.

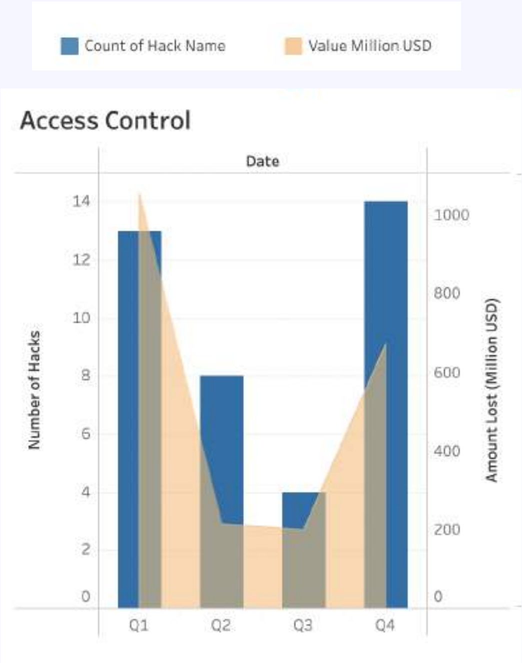
Another aspect of smart contracts that adds to their security is the use of languages that are designed to be resistant to vulnerabilities and bugs. For example, the Solidity language, commonly used to write smart contracts on the Ethereum blockchain, includes features like type checking and exception handling that help prevent common vulnerabilities such as those caused by incorrect data types or unexpected inputs.

For these reasons, smart contract adoption is anticipated to skyrocket. According to recent projections, the value of using smart contracts will reach $1.5 billion by 2032 , a tenfold increase from where it is now. Smart contracts are being employed in a variety of sectors, including financial services, banking and healthcare.

Unfortunately, as smart contracts gain popularity the technology is also being widely misused by scammers and hackers to swindle huge amounts of users' funds. The DeFi sector lost more than alone as a result of smart contract vulnerabilities, eclipsing the entire amount plundered in both 2020 and 2021 altogether.

In 2022, a basic programming error triggered the hacking of some protocols, whereas weak contract logic or invalid calculations caused the hacking of other protocols. It is essential to understand some of the major smart contract vulnerabilities and how the industry has been affected by them.

Hacks due to Access Control Vulnerabilities



Per our analysis, access control is the second most popular vulnerability to be exploited in the year 2022 in terms of number of attacks. As mentioned previously, it holds the lion share of total losses caused through out the year with 2.14 Billion USD. While the total number of attacks are circa 40, just Quarter 1 and Quarter 4 have seen 13 and 14 respectively, however, in the terms of amount lost, access control contributes to over 1 billion in Quarter 1 and 670 million USD in Quarter 4.

Smart contracts are smart not only because they can execute code when certain conditions are met, but because they can enable permissions similar to software. This is known as access control, which defines the roles of users in a smart contract, and in extension, their privileges or restrictions.

Because smart contracts are agreements often built around the creation or exchange of value, access control is crucial for security. As an example, one popular application powered by smart contracts is Uniswap, which is a decentralized exchange that allows users to trade cryptocurrency without any intermediary. Another is Compound, which enables borrowers and investors to offer and take loans without an intermediary.

The critical functions in the smart contracts of Uniswap, Compound, and other applications require the implementation of access control. These critical functions include those broadly related to value, such as the ability to mint tokens, freeze funds, or make withdrawals, as well as those related to administration, such as the ability to validate or vote on proposals or upgrade the smart contract itself.

Access control is important because it prevents the abuse, corruption, and manipulation of critical functions in a smart contract. With access control, only the validated entity can trigger privileged operations. Conversely - and perhaps much more crucially - non-validated entities cannot trigger privileged operations when proper access control is in place.

There are two means of access control: ownership control and role-based access control.

Ownership control

This form of access control is ideal for contracts that require a single administrative user, who has exclusive ability to perform administrative functions. This ownership, for instance, can be provided via Ownable, a contract module that enables access control.

In object-oriented programming, the constructor is a method that is called whenever an object of a class is initialized. What distinguishes the constructor() function in Solidity is that it is provided within the smart contract and initializes the deployer of the smart contract as the first owner, which is only invoked during this deployment. The first owner can then call functions that have the onlyOwner() modifier.

Since ownership is limited to one user, the owner can execute two commands to change this: transferOwnership to give the account to someone else and renounceOwnership to abandon ownership, which then prohibits the calling of functions with onlyOwner(). renounceOwnership is common when a project no longer needs centralized control, which highlights the pros and cons of this model:
While having a single administrator is operationally efficient, it also brings the risks associated with centralization. With ownership control, there is only a single point of failure for hacks and attacks, consequently, if the account of the owner is compromised, the security of the the platform will be jeopardized making it vulnerable to attacks.
On 8th September 2022, Ragnarok Online Invasion, a cryptocurrency, deployed on Binance Smart Chain, fell prey to access control vulnerability of the ownership transfer function. Since, the transferOwnership function's visibility was set to public, the attacker was able to transfer ownership to himself and steal around 158 BNB from the protocol.

Role-based access control

This form of access control is similar to what people may encounter on other enterprise software or platforms. On Facebook, for example, there are multiple roles for managing a brand's page, including admin, editor, moderator, advertiser, and analyst. Each role has its own set of actions that it can perform. An analyst can view page insights, for instance.

Role-based permissions are governed - in enterprise software and in smart contracts by the principle of least privilege: The role that each actor in a system is assigned to provides only as much access as they need to perform their given functions. To return to the above example, while the analyst can freely examine data, he cannot edit page information, respond to customer inquiries, or perform any other function non-essential to his own work.

The principle of least privilege is crucial for smart contracts because it grants specific abilities to users, who cannot act beyond these intended roles. Upon deploying a contract, the developer can specify different roles, who have the ability to perform certain actions, such as minting or burning tokens, implementing upgrades, and even triggering emergency actions, such as pausing withdrawals or revoking access.

By segmenting the responsibilities of roles, this role-based access control increases decentralization, which improves security. As with ownership control, there is no longer a single point of failure. The trust assumption for each user is also substantially less because the damage that can be done from any given account - whether intentional or unintentional - is much smaller due to their specialization.

Role-based access control can be paired with other measures to further enhance security. One such example is a multi-signature account, or multi-sig, which will require the signature of a minimum number of accounts to execute a function. This improves decentralization, reducing the chances of abuse by a single party, such as rug-pulling users. Fault tolerance is also achieved because multi-sig follows an m-of-n signing scheme, where m (the actual signers) of n (the total number of possible signers) have to agree. If only 5 out of 10 sign a wallet, but 6 out of 10 are required, the proposal will not go into effect, reducing the chances of a handful of bad actors attempting to pass malicious updates. But even multi-sig is not without its vulnerabilities. In July 2017, the wallet software Parity was hacked. In a post-mortem issued by the company, they explained that the hack stemmed from 4000 lines of Javascript, HTML, and CSS that introduced a new wallet user interface. While the addition was reviewed by a Solidity expert, the team did not give the changes too much scrutiny as they believed it to primarily be a user interface issue.
As the new code was made public, the hackers were able to discover the vulnerability: They could now reset ownership and permissions arbitrarily, rather than just when the contracts were created. They subsequently set up three multi-sig wallets that had a large amount of ETH. By signing these newly designated multi-sig wallets, they were able to transfer the ETH into their own accounts, which it then laundered. Multisig failed here because it allowed bad actors to seize control of an account by sheer numbers, similar to a hostile takeover in the world of equities.

While Parity took responsibility for the exploit and discussed measures they would take to prevent future hacks, they also said some of the blame lay with the Solidity language itself.

“One would be to change the default access mode of functions to ‘private’, rather than the eminently insecure ‘public’. Another, even safer, change would be to make it illegal to include functions that have neither an explicit access modifier nor a guard modifier,” the team wrote.

Another incident that highlights the vulnerability of multi-sig was the Sky Mavis hack in March 2022. The hack was orchestrated by Lazarus, a North Korean cybercrime group, which dangled a fake job offer to a senior developer at Sky Mavis. To make the ruse more believable, the hackers put the developer through an interview process, before sending over a job description. The developer opened the malicious file, which compromised his computer and eventually gave the hackers access to the Sky Mavis servers. This is where multisig failed. To sign off on deposits and withdrawals, 5 of 9 signatures were needed. In an ideal scenario, each of these 9 wallets would have been owned by separate individuals or entities. The Ronin blockchain had some centralization, however, and so 4 out of the 9 keys were actually owned by Sky Mavis. Because the Lazarus Group now had access to the Sky Mavis servers, they had these 4 keys. They gained the necessary 5th key by hacking into the community-run Axie decentralized autonomous organization. From there, the Lazarus Group signed off on transactions as they pleased, walking away with $600 million worth in cryptocurrency.

Access control vulnerabilities
As the Parity example showed, there are many vulnerabilities in access control.

- **DASP 10** - Smart contracts initialize its deployer as its first owner, which gives it special privileges in either ownership or role-based access control model. But while the deployer may have been initialized as the first owner, any person can later call this function and the subsequent privileges it provides, such as the ability to transfer funds.

- **Broken authentication flaw** - This is a breakdown in access control: Hackers are able to bypass the authentication mechanism, owing to some oversight on part of the developers, such as bugs, logic errors, or poor configuration. Because some of these development mishaps may be egregious, hackers do not need extensive technical sophistication to take advantage of a broken authentication flaw. Once in a system, they can use higher-level functions to their advantage.

- **Vertical privilege escalation** - This is one subtype of the broken authentication flaw that is what it sounds like: A hacker is accessing functionality (i.e. the privilege) that should only be available to a role higher in the hierarchy. For example, the hacker may be calling a mint function that should be reserved only for the administrators of a smart contract. This kind of vertical privilege escalation can be accomplished by metadata manipulation or even force browsing to authenticated or privileged pages as an unauthenticated or standard user respectively. This exploit skirts around the principle of least privilege: Even if roles only have as many permissions as they need to perform their duty, the attacker uses their ingenuity to access higher-level functions anyway.

- **Horizontal privilege escalation** - Horizontal privilege escalation is the other type of privilege escalation attack. Unlike vertical privilege escalation, which avails of privileges usually exclusive to a higher role, horizontal privilege escalation is about using a known exploit to gain access to other functionalities with similar ownership. For example, if attackers have already compromised an admin account with one exploit, they may use the same technique to take over other admin accounts, so they can obtain the quorum necessary to sign off on a multi-sig.

- **Platform misconfiguration** - When permissions are misconfigured, attackers can access to manipulate settings

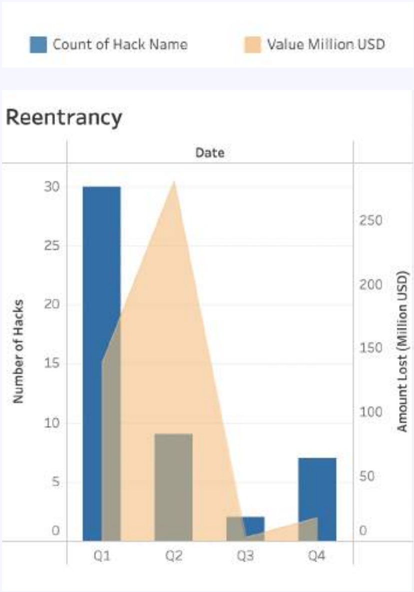
-

Access control mitigation

Because access control attacks are not unique to smart contracts, some of the best practices for mitigating these are the same as with other technologies. One of the resources that the SWCRegistry recommends is in fact access control relevant to all software and hardware. To this end, developers are encouraged to carefully configure access control, and to create clearly distinct trust zones between the different levels of privileges. When designing this compartmentalization, developers should indeed follow the principle of least privilege, so that users only have access to what they absolutely need for their designated role. Access control that follows the principle of least privilege will minimize the chance of security breaches.

Access control hacks are also easier when smart contracts have insecure visibility settings, according to the DASP. This provides would-be attackers a window into the private values or logic of the smart contract, which can

Hacks due to Arithmetic Issues



Exploits of the arithmetic vulnerability have only occurred twice in the year 2022 which amounted to a total loss of 45 Mil USD, with both the attacks having occurred in the 2nd quarter. While contextually a total loss of 45 Mil USD and 2 attacks in the entire year might not be as threatening, going forward we need to stay vigilant of these new vulnerabilities being exploited.

Introduction to Arithmetic Issues

Arithmetic issues mainly correspond to integer overflow and underflow. Arithmetic issues have been a long-standing issue in tech. They are especially dangerous when it comes to smart contracts, where attackers can use this vulnerability as an avenue for malicious actions, such as denial of service or theft.

To define arithmetic issues, the DASP-10 cites kumabits.com founder Jules Dourlens: “An overflow condition gives incorrect results and, particularly if the possibility has not been anticipated, can compromise a program’s reliability and security.” This definition is very much aligned with the SWC-101 standardization task.

To understand integer overflow and underflow, it’s important to understand the two types of integers in Solidity. There first are unsigned integers, which must be non-negative. The other data type is signed integers, which can be positive or negative.

An integer overflow or underflow would arise if the operation attempted to create a value beyond the range that can be represented by bits, which is [0, 2^{256}-1] for unsigned integers. Because the range cannot represent the value, the value will revert to the minimum number if it is beyond the maximum, or the value will reset to the maximum number if it is less than the minimum.

While a fixed range may be problematic, smart contracts need to be memory efficient for practical reasons. While there is a limit to the data storable in a smart contract, this is mostly theoretical. The more practical constraints are costs (there are gas costs associated with processing data) and efficiency (as with all code, the cleaner it is written, the faster it will run).

Smart contracts do not have floating point support, which uses a significant with a base ten exponent, due to their unpredictability - they could provide different results and even lead to round-off errors. Smart contracts instead use integers to represent value in keeping with a convention on financial programs. As an example, it's easier to say something is worth 50¢ than 0.5 dollars, which allows for greater accuracy and granularity.

Introduction to integer overflow and underflow

Both types of arithmetic issues are caused by assigning a value to a variable that exceeds or falls below the data capacity. This overflow or underflow occurs because the Ethereum Virtual Machine specifies the fixed-size data type for integers. As a result, there are only a certain range of numbers it can represent.

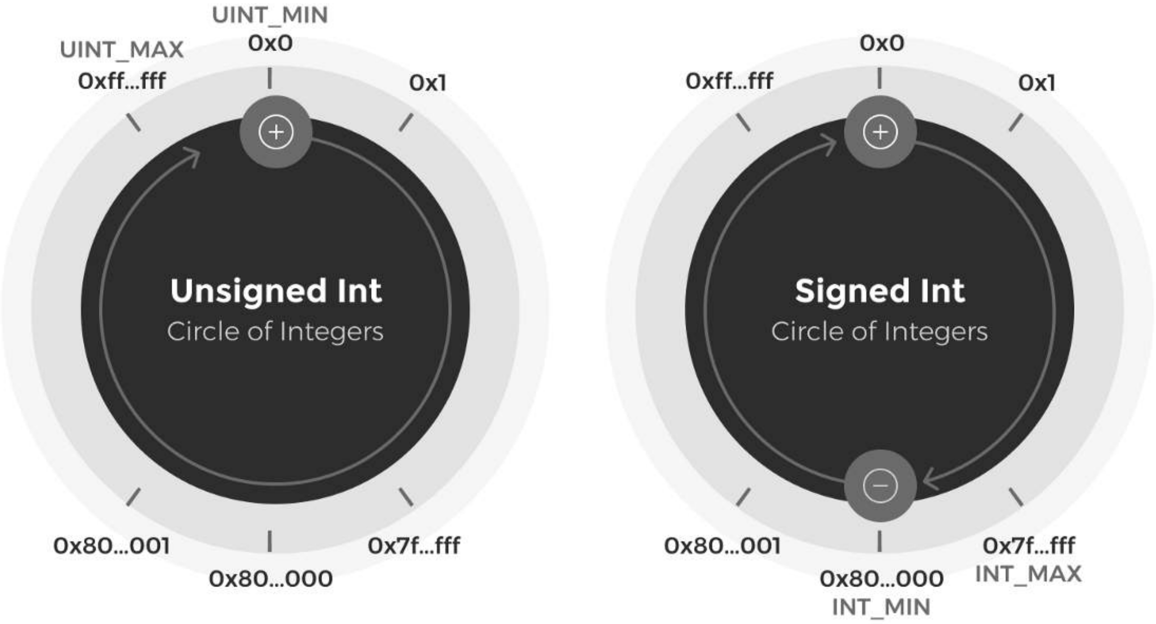
In most everyday usage, this range works fine. It breaks down during edge cases where a person, intentionally or unintentionally, attempts to store a value higher than maximum or lower than the minimum. The resulting effect is similar to how integers generally work in programming. A maximum value is set, as is a minimum value. When the maximum value is reached, the number is reset to the minimum.

Famously, the Y2K bug had a conceptual or logical failure that would result in an issue akin to an integer overflow. Because most programs had represented dates by only their final two digits (i.e. ‘75, ‘86, ‘97) rather than their full value (i.e. 1975, 1986, 1997), many feared that the New Year would see widespread chaos, as systems failed to distinguish between 2000 and 1900.

While Y2K ended up causing mostly minor inconveniences, it is clear from this example why arithmetic issues would be especially problematic for smart contracts. An underflow, for example, could result in a value going from the minimum to the maximum.

Though arithmetic issues can have potentially disastrous consequences, they are often overlooked because they violate common-sense assumptions. Most people would assume that deducting a larger number from a smaller number would result in a zero, or at worst case, a negative value, just as how banking works in the real world. Our personal bank accounts can get down to zero, and if we over-draw, we can even bring it to a negative number (i.e. we now owe the bank money). The idea of an underflow, or a minimum being reset to a maximum, goes counter to this concept.

The same applies with overflows. When we add two numbers together, the underlying assumption is that the sum will be a larger number. This is so ingrained in us as to almost be a rule, which is why we look past possible exceptions: Two large numbers will result in a smaller number when its sum happens to exceed the maximum. Arithmetic issues are often directly exploited, as in the case of underflows that allow attackers to walk away with a large number of tokens. But they are just as often entry points for remote-code execution exploits.



Anatomy of Arithmetic Issue Attack

Source: https://valid.network/post/integer-overflow-in-ethereum

How to prevent arithmetic issues

- Manual analysis - The development team can identify the range, including both the maximum and minimum values, that can be stored for their program. Then they must test these figures for proper validation and verification. From there, the development team can implement guardrails that prevent values larger than the maximum or lower than the minimum from being stored.

- Tools - There are a variety of open-source tools that can help developers automatically uncover any potential arithmetic issues. Orisis, Mythril, and Manticore are one of the many execution tools that can be used for running syntactic, semantic, and a runtume analysis of smart contracts.

A tool, however, is only as good as the person using it. To this end, developers need to use tools only for their intended purpose: Some, for example, are only for syntactic and semantic checks, while others are for run-time analysis. Using tools for anything other than their intended purpose may result in overlooked arithmetic issues.

- Trusted, proven math libraries - To avoid arithmetic issues, the SWC also recommends the use of vetted and safe math libraries. This best practice would have prevented the Saddle Finance hack described below, wherein the use of an outdated library introduced calculation errors that had already been updated.

- Audit smart contracts before deployment - While the DASP also encourages the use of the tools and methods described above, it also cautions that this industry is still very much emerging. To make the most out of these, there is a cultural change that must occur as well. Unlike software, where bugs can be fixed in real time, any issues with smart contracts are essentially unchangeable once deployed. It's therefore crucial that these nascent tools are used from the beginning, so that they can be remedied before they are released.

Hacks caused by arithmetic issues

Aku Dreams - Micah Johnson, a former professional baseball player, founded the studio Aku Dreams. Unlike other celebrities who came into the NFT space relatively late, Johnson was an early adopter, creating and selling NFTs since the heyday of the technology in the early 2020s. Through Aku Dreams, Johnson launched Akutars, a flagship collection of 15,000 NFT avatars.

Akutar was supposed to be his flagship collection, but its launch was mired in controversy from the beginning. In the run-up to a public auction for the NFTs, at least one smart contract researcher warned of the possible exploit, but these were dismissed by the Aku Dreams team who argued that it had been addressed. When the public auction went on as planned, a griefer - someone who did not plan to steal cryptocurrency but wanted to make a point - exploited this vulnerability. He executed a “griefing contract,” which prevented Aku Dreams from processing refunds to those who underbid, effectively locking away $33 million in ETH.

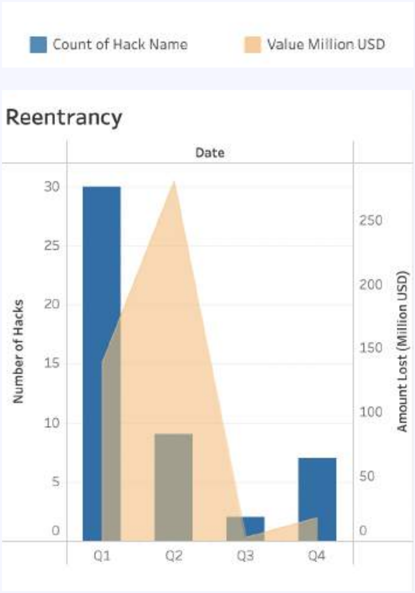
The exact exploit was an underflow which reset each bid status to 1. In a thread on Twitter, a Solidity developer explained that this occurred because the count of bids did not increment in lockstep with the mint amount. But as the bid count required correct incrementing, withdrawal was prevented and the processRefunds() function could not be called.

Saddle Finance - In April 2022, automated market maker Saddle Finance lost $10 million due to a hack (though auditing firm BlockSec later helped recover $3.8 of the funds). The roots of the vulnerability lay in the fact that Saddle reimplemented Curve’s Meta pool in Solidity, rather than fork the Vyper code. As a result, Saddle used an older calculation library that contained faults which newer versions had already corrected.

The arithmetic error was curiously only present when swapping tokens for the liquidity provider (LP) tokens, in this case, saddleUSD-V2, but not for other functions such as depositing or withdrawing. As a result, LP tokens were mispriced: For 1 dollar of sUSD, a person could retrieve less, or crucially, more than $1 in saddleUSD-V2.

Using this vulnerability, the hacker made a series of flash loans, assisted by sUSD to saddleUSD-V2 swaps in the Meta pool. This manipulated the price of LP tokens, which the hacker swapped for more US dollars, which he subsequently withdrew.

Hacks due to Reentrancy Vulnerabilities



In the year 2022, reentrancy attacks account for a total loss of 441 Mil USD. With over 45 attacks throughout the year, reentrancy attacks stand 3rd in the list of the vulnerabilities that caused the maximum amount of losses. Quarter 1 2022 has seen as many as 30 reentrancy attacks which accounted for 141 Mil USD.

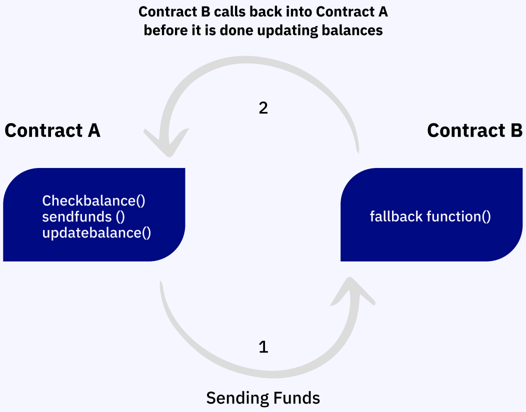
A reentrancy attack exploits vulnerabilities in code to gain access to an individual or organization's systems. In essence, it is a type of attack that enables an attacker to repeatedly call a vulnerable smart contract function in order to deplete the targeted account's resources or gain control of the targeted account. A successful reentrancy attack is often performed by exploiting a vulnerability in the code that allows the attacker to withdraw funds prior to the smart contract updating the balance of the targeted account.

In this type of attack, malicious actors use the same code multiple times to make unauthorized transactions. The way it works is that the malicious actor sends a transaction to the smart contract, and then sends a second transaction before the contract can finish processing the first transaction. Because the second transaction requests the same data from the smart contract as the first, it causes the contract to process the same code multiple times.

For a better understanding, a reentrancy attack is a scenario in which a malicious contract calls a function of a vulnerable contract, after the completion of which, the malicious contract fails to update its state. This results in the contract assuming that the triggered action isn't complete and continues to repeat the same action again and again.

For example, let’s assume ‘Contract A’ is a vulnerable contract, and ‘Contract B’ is a malicious contract.

- An attacker makes a call to ‘Contract A’ to transfer funds to ‘Contract B’.
- On receiving the call, ‘Contract A transfers the requested amount of funds to ‘Contract B’.
- Upon receiving these funds, the malicious contract executes a fallback function that calls back into the vulnerable contract before it updates its balance, thus repeating the process all over again.
- This process continues until the vulnerable contract is drained of all its funds.



Exploring how a re-entrancy attack works

source: https://hackernoon.com/hack-solidity-reentrancy-attack

A reentrancy attack can be carried out in the following five steps:

- A hacker builds a malicious smart contract.
- This contract is then used to call the target contract’s withdrawal function.
- Once the funds have been received, a fallback function in the malicious smart contract is triggered, that repeatedly calls the withdrawal function.
- The target’s smart contract can’t update the attacker’s balance as the program flow is interrupted halfway.
- This process is called multiple times until all the funds get stolen

It makes sense that withdrawal functions in smart contracts will adhere to the Check-Interaction-Effects structure. The Checks-Effects-Interactions pattern is a security consideration in smart contracts that ensures all code paths through a contract complete all required checks of the supplied parameters before modifying the contract's state (Checks); only then it makes any changes to the state (Effects); it may make calls to functions in other contracts after all planned state changes have been made (Interactions). Accordingly, they would confirm a withdrawal's legitimacy, carry out the transfer, and update their internal status. However, as Ethereum has fallback methods, this coding design introduces a reentrancy risk. A smart contract that invoked the susceptible function during the interaction phase of the procedure has the option to execute the set of code instructions in the fallback feature. The second time this code invokes the vulnerable function, it will execute before the program updates itself internally.

Different Types of Reentrancy Vulnerabilities

- **Single-function Attacks**

A single reentrancy attack takes place when there is a vulnerability in the same function that the attacker is trying to call recursively. If a contract uses functions like call, send or transfer it may shift control flow to an external or malicious contact, with a fallback function. The contract fails to update the state of the contract, this causes the state of the contract to be incomplete even when the transfer is complete. Therefore, when this fallback function is triggered, the flow of control may not return to the previous contract and the caller might do ‘n’ number of unexpected things such as calling the function again, calling another function or even calling another contract.

- **Cross-function Attacks**

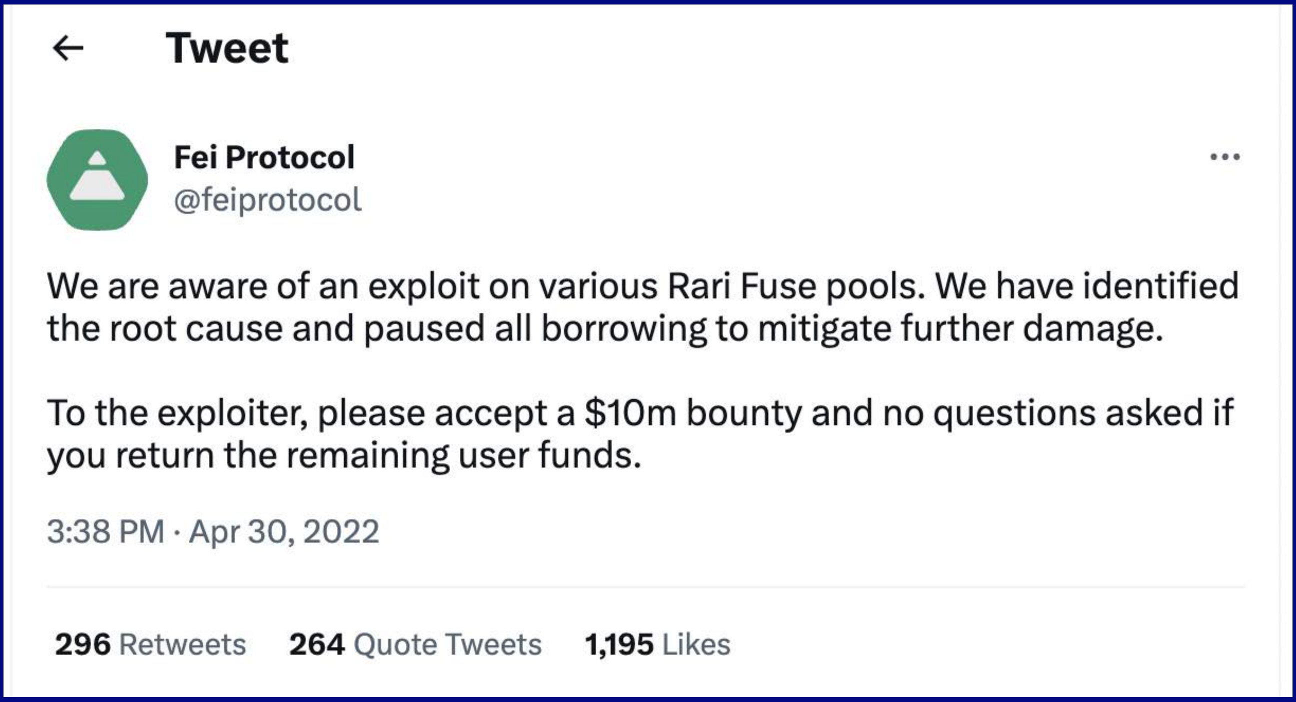
A similar attack can be done when two different functions or contracts share the same state. The only difference between a Single- function attack and a Cross-function attack is that in single-function attacks, the same vulnerable function is recursively called by an attacker, whereas in a cross-function attack, the function reentered is not the one making the external call.

Rari Exploit

Multiple Rari Fuse pools caused the loss of more than $80 million for the FEI protocol due to a fatal flaw in the code. The FEI protocol is an algorithmic stable cryptocurrency project for decentralized finance that aims to help DAOs with deep liquidity in borrowing and exchange markets by implementing PCV and providing Liquidity-as-a-Service (LaaS).

The major cause of the exploit was a reentrancy vulnerability, which allowed a hacker to borrow assets while removing all given collateral. As security, the attacker flash loaned a large amount of wETH and borrowed tokens from the pool. Because the code did not follow the standard pattern of check-effect-interaction, the attacker called "exitMarket()" to withdraw all of his collateral since the loan records had not been properly updated.

- The attack started with invoking the doTransferOut () function, allowing the hacker to acquire ETH before the borrow() function updated its status.
- The attacker calls for the exitMarket() function and this causes the asset to no longer be collateral and makes it ready for withdrawal as there is no record or proof of borrowed ETH. The hacker receives the borrowed ETH for free.
- The hacker proceeds in this fashion till all borrowable assets have been exhausted from a number of lending pools.
- After repaying the flash loan, the attacker transferred the remaining assets as profit.



We are aware of an exploit on various Rari Fuse pools. We have identified the root cause and paused all borrowing to mitigate further damage.

To the exploiter, please accept a $10m bounty and no questions asked if you return the remaining user funds.

3:38 PM · Apr 30, 2022

296 Retweets **264** Quote Tweets **1,195** Likes

Fei Protocol announcing the exploit

How to mitigate attacks due to reentrancy vulnerabilities:

Updating balance before transferring:
Reentrancy attacks are a type of vulnerability in smart contracts that can be mitigated by updating the balance before transferring funds. This is because reentrancy attacks exploit the fact that a contract's state can be changed multiple times during a single transaction. By updating the balance first, it ensures that the contract has enough funds to complete the transfer and prevents any further changes to its state.

Auditing:
Auditing contracts helps in mitigating reentrancy attacks by ensuring the arithmetic safety of Ethereum smart contracts, enforcing the writing of robust smart contracts, providing a formal verification for the hash time lock, and evaluating a variety of attacks and their mitigations. Additionally, procedures can be established for security and privacy programs to help individuals ensure that their data is accurate.

Reentrancy Guard:
Reentrancy guards helps in mitigating reentrancy attacks in smart contracts. These guards are code that prevents a contract from being called again before the previous call has finished executing. This prevents attackers from exploiting vulnerabilities in the contract's code that lead to the stealing of funds or executing malicious code. Reentrancy guards are an important part of smart contract security and are usually audited using tools like VeriSmart and Verx. Procedures for security programs also help in ensuring data accuracy and privacy.

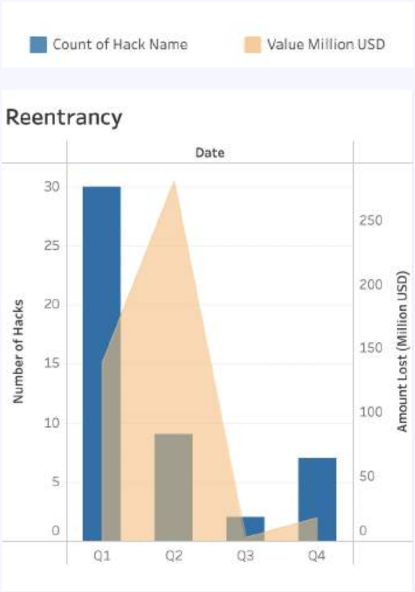
Checking the call stack Depth:
The call stack depth is the maximum number of function calls that are made before the stack overflows and causes an exception. In Solidity, this limit is set to 1024 frames. When a contract exceeds this limit, it fails or becomes vulnerable to reentrancy attacks. The "Stack Too Deep" error is a common issue in Solidity programming that occurs when there are too many variables declared in a single function. This error is resolved by refactoring the code and splitting the function into smaller functions with fewer variables. Reentrancy attacks are a type of vulnerability in smart contracts that are mitigated by checking the call stack depth. This helps prevent malicious actors from exploiting the contract's code and executing it multiple times. Flint is a programming language designed specifically for writing robust smart contracts. .

Use of Require Statement:
The use of “require statements” helps mitigate reentrancy attacks by ensuring that a contract's state is updated before any external calls are made. This prevents an attacker from recursively calling the target's withdraw function and draining funds from the target. The “require statement” checks if a certain condition is met before executing the rest of the code. Such aspects are used to enforce the correct run-time behaviour of smart contracts. By using this approach, developers avoid reentrancy attacks and ensure that their smart contracts are secure.

Use of Function modifiers:
Function modifiers are used to restrict the access of external calls to certain functions, thus preventing malicious actors from exploiting them. Additionally, function modifiers are used to check the state of a contract before executing a transaction, which helps prevent unexpected behavior. Using function modifiers can help mitigate reentrancy attacks by restricting the access of external calls to certain functions. This ensures that only authorized users can call these functions, thus preventing malicious actors from exploiting them. Additionally, function modifiers can be used to check the state of a contract before executing a transaction, which helps prevent unexpected behavior.

In summation, reenterancy attack should identified and remediated by employing extensive audits and check-effects-interaction designs in the company of other security measures.

Flash Loans Attacks



According to our investigation there were 4 flash loan exploits in 2022 which amount to a total of 20 mil USD in losses. Of which, 1 occurred in Q1 which accounted for 2 Mil USD, 1 in Q2 which resulted in losses worth 13 Mil USD and 2 in Q3 which accounted for 5 Mil USD in losses. While the actual number of attacks involving flash loans are much higher, we have categorized a few as price oracle manipulation for ease of understanding.

Flash loan attacks are a type of smart contract security threat that has emerged in the decentralized finance (DeFi) space. Under this type of exploit, the attacker first borrows a hefty sum without collateral, executes a malicious transaction and then repays the loan before the end of the transaction block. These attacks are the most common among all DeFi attacks since they require lesser monetary resources and are the easiest to get away with. In fact, to carry out a flash loan attack, all that an attacker requires is a computer, an internet connection, and ingenuity. For attackers, these are low-risk, low-cost, high-reward schemes, which they rigorously use to launder millions of dollars of funds online..

What are Flash Loans?

A Flash Loan is an uncollateralized loan that allows users to borrow and return funds within the same transaction or a few seconds before a new block is created on the blockchain. The borrower quickly flips a profit on the amount and returns the initially borrowed funds with the help of flash loans. Flash loans are easily accessible unsecured loans that require no collateral and credit checks and have no fixed limits on the number of funds that one can borrow. Though hackers need to plan the attack in advance, the execution merely takes a few seconds. Flash loan attacks has opened doors for artificial arbitrage opportunities that involve manipulating asset prices to gain profit. Arbitrage is the practice of rapidly buying and selling the same asset in different markets to take advantage of the price differences between various markets.

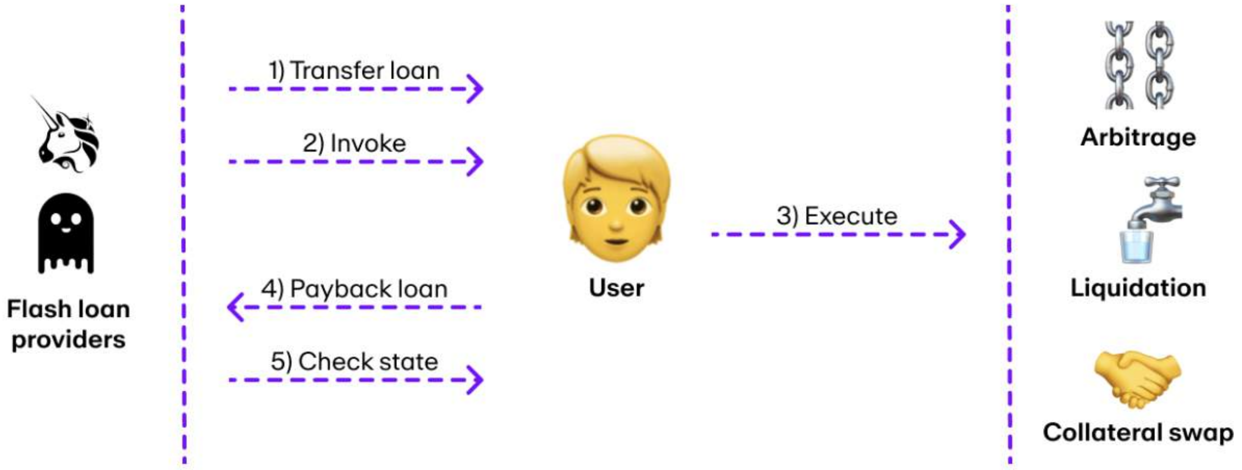
How do Flash Loans work?

A flash loan is often used for making a profit by taking advantage of price discrepancies between different markets and protocols. It is a bespoke smart contract that is based on specific predefined trading conditions.

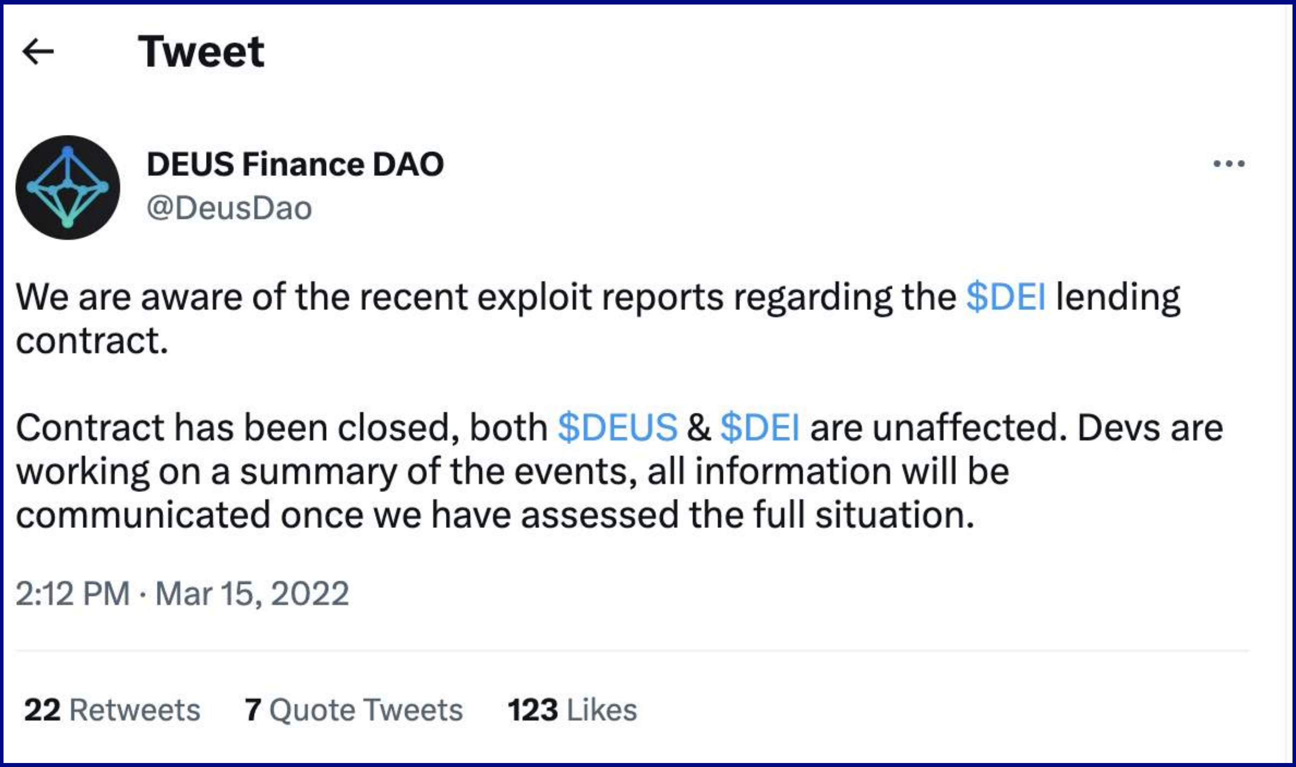
Using formerly designed smart contracts, flash loans are carried out in the following five steps:

Transfer requested assets - Flash Loan providers transfer the requested assets to the borrower.
Invoke operations - Users' pre-designed operations are invoked.
Execute Operations - Users interact with other contracts to execute operations with the borrowed assets.
Repay Loan - Once the execution is complete, the borrower returns the borrowed assets along with an extra fee if applicable.
Check the balance - Flash Loan providers check the balance. If the deposited amount is insufficient, they immediately revert the transaction.

One of the numerous ways of carrying out a flash loan attack is by exploiting vulnerable smart contracts. Flash Loans can only be borrowed on the condition that the borrower repays the exact amount of assets within the same transaction. Attackers usually find a way to alter the smart contract used in the process by changing the value of the cryptocurrencies they are trading. This essentially tricks the contract into thinking that the loan has been repaid when it hasn't.



Example of how a Flash Loan Attacks works


Tweet

DEUS Finance DAO
@DeusDao

We are aware of the recent exploit reports regarding the $DEI lending contract.

Contract has been closed, both $DEUS & $DEI are unaffected. Devs are working on a summary of the events, all information will be communicated once we have assessed the full situation.

2:12 PM · Mar 15, 2022

22 Retweets 7 Quote Tweets 123 Likes

Deus Finance announcing the exploits

Analysis of DEUS Finance Exploit

On April 28, 2022, a $13.4 million flash loan attack was carried out by targeting the Fantom contract of DEUS finance.

- In advance, the attacker deposits approximately 0.92 sex-sAMM-USDC/DEI LP tokens in the target contract.
- The attacker acquired approximately 143.2 million USDC via several flash loans and thereafter exchanged for 9,547,716 DEI via USD/DEI pair contracts. As there was a substantial rise in the USDC pairs, the attacker altered the oracle to increase the value of the asset used as collateral- sex-sAMM-USDC/DEI.
- The attacker uses the collateral provided in step 1 to invoke the DeiLenderSolidex.borrow function and take roughly 17,246,885 DEI.
- Lastly, the attacker repays the flashloan by exchanging the DEI for USDC.

Mitigation Strategies and best practices

Since, flash loan attacks have resulted in significant financial losses for DeFi protocols, users, and investors, it is essential to employ the best practices to mitigate flash loan attacks including some of the key industry standards. These best practices are highlighted below:

Implement Transaction Guards:
Another effective way to mitigate flash loan attacks is to implement transaction guards. Transaction guards are smart contracts that verify whether a transaction is valid before executing it. These smart contracts ensure that the transaction does not contain any malicious code or exploits that could be used to execute a flash loan attack. For example, Aave, a DeFi lending protocol, uses transaction guards to validate whether a borrower has sufficient collateral to take out a loan.

Implement Circuit Breakers:
Circuit breakers are another best practice for mitigating flash loan attacks. Circuit breakers are automated protocols that trigger a pause in a protocol's operations when certain conditions are met. For example, if a sudden drop in the value of a particular asset occurs, a circuit breaker is triggered to prevent further losses. Some DeFi protocols, such as MakerDAO, implement circuit breakers to mitigate flash loan attacks.

Using Decentralized Oracles:
Decentralized oracles provide reliable and tamper-proof price feeds that are used to verify the prices of different assets across various decentralized exchanges. By using decentralized oracles, exchanges ensure that the prices they are arbitraging are accurate and not manipulated, reducing the risk of flash loan attacks. Chainlink is an example of an organization that uses decentralized oracles to ensure the accuracy of data being fed into smart contracts.

Auditing:
Auditing involves a thorough review of smart contract code to identify any vulnerabilities that attackers could exploit. Auditing helps identify and mitigate potential risks associated with flash loans, such as reentrancy attacks, which are one of the most common types of attacks associated with flash loans. **ConsenSys Diligence** for instance, offers auditing services for smart contracts to identify potential vulnerabilities and ensure that they are secure and function as intended.

Slippage Limit:
The slippage limit refers to the maximum difference between the expected price and the actual price of an asset that an exchange is willing to tolerate. By setting a slippage limit, exchanges reduce the risk of being caught in a flash loan attack that exploits price differences. Uniswap is an example of an exchange that has a slippage limit in place. Uniswap's default slippage limit is 0.5%, meaning that if the execution price of a trade differs from the expected price by more than 0.5%, the trade will be rejected.

Blacklisting Ability:
Blacklisting ability refers to the ability to blacklist specific addresses associated with malicious activity, such as known flash loan attackers. By blacklisting these addresses, exchanges prevent attackers from performing flash loans on their platforms. For example Binance can block certain wallet addresses from using its platform, either permanently or temporarily.

Runtime Monitoring:
Runtime monitoring involves monitoring smart contracts and transactions in real-time to identify any abnormal or suspicious activities. By monitoring transactions in real time, exchanges detect and respond to flash loan attacks quickly, reducing the damage caused by such attacks. OpenZeppelin provides runtime monitoring services for smart contracts. OpenZeppelin's Defender service allows developers to monitor their smart contracts for security events, such as abnormal contract behavior, and receive alerts if any issues are detected.

Coverage Protection:
Coverage protection refers to purchasing insurance coverage to protect against losses associated with flash loan attacks. By purchasing coverage protection, exchanges transfer the risk associated with flash loan attacks to insurance companies, reducing their financial exposure. Nexus Mutual provides a decentralized insurance platform where users can purchase coverage for their smart contracts against various risks, such as bugs, hacks, and oracle failures. If a covered event occurs, Nexus Mutual will pay out a claim to the policyholder.

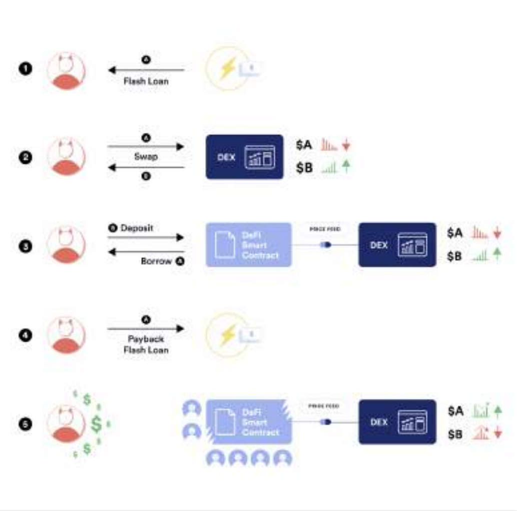
In conclusion, mitigating flash loans requires a comprehensive approach that includes using transaction guard, circuit breakers, decentralized oracles, auditing smart contract code, setting slippage limits, implementing blacklisting, monitoring transactions in real-time, and purchasing coverage protection. These measures help reduce the risk of flash loan attacks and protect user investments.

Flash loans and price oracle manipulation
In our investigation we found that protocols that used spot price of DEXs as sole price oracles suffered from flash loan attacks heavily. Malicious actors quickly influenced the price with a sizable trade by using such protocols.

To demonstrate the process, here is a brief explanation of the process.

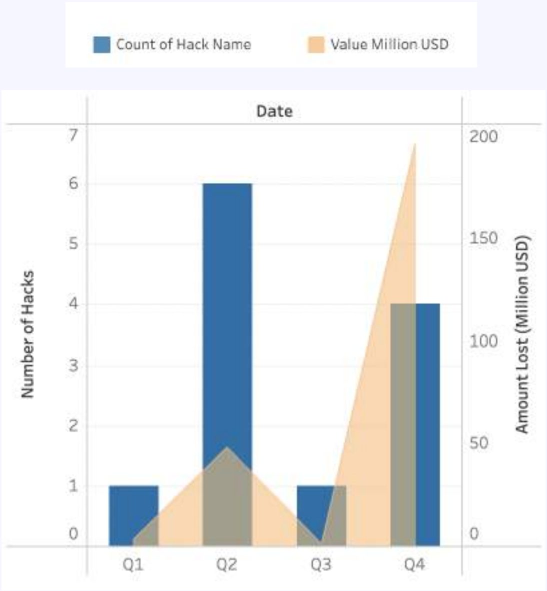
- An exploiter borrows a significant amount of token X from the protocol that enables flash loans.
- The exploiter then proceeds to swap token X for token Y on a DEX - causing the price of token X to go down and increasing the price of token Y - and then uses it as collateral for the DeFi protocol, which only receives price information from the aforementioned DEX.
- The attacker then makes a profit on the protocol's altered price feed, repays the initial flash loan, and borrows more token X than would normally be possible.

Protocols are prone to price point manipulation when volume switches to other exchanges or a person temporarily changes the price on that exchange if a single on-chain platform is used as the price feed. This is particularly dangerous for low-liquidity assets that are used as collateral in DeFi lending protocols.
To prevent price oracle attacks due to flash loans, using decentralized price feeds as market data source validators and avoiding manipulable DEX spot prices are two best practices to start with. Additionally, this helps ensure that DeFi protocols consistently receive an aggregate price that matches the trading market and is resilient to flash loans, mitigating price oracle manipulation attacks.



Examining the relationship between Flash loans and price oracle manipulation
source: https://chain.link/education-hub/flash-loans

Price Oracle Manipulation Attacks



According to our analysis in total, 2022 has seen 12 price oracle manipulation attacks which amount to a total loss of just over 250 mil USD. Although there are 6 recorded attacks which have resulted in 49 mil USD in losses in Q2, Q4 has seen only 4 attacks which have resulted in losses of as much as 197 Mil USD.

Introduction to Oracles

With respect to blockchain and decentralized finance (DeFi), an oracle is a vital intermediary service that provides off-chain data to on-chain smart contracts. Oracles are crucial for DeFi as they facilitate relevant data transmission between the on-chain and off-chain ecosystems, granting permission for smart contracts to securely interact with sources of data and other systems outside of the boundaries of the blockchain network.

Oracles are required in DeFi as decentralized applications (dApps) require access to up-to-date external information in order to run correctly. As an example of external information, a decentralized exchange (DEX) must leverage outside resources to get current cryptocurrency market values while a decentralized insurance platform needs direct contact with real-time statistics to ascertain whether prerequisites have been met.

Oracles amplify DeFi's practicality and efficacy as they serve as reliable agents that can authenticate and transmit data to smart contracts, thereby allowing smart contracts to carry out respective operations based on the precise information.

Oracles provide this data to the platform and ensure accuracy of the information by attesting to its veracity. In order for a DeFi platform to use an oracle, it must either create its own internal oracle system, or attach itself to an existing well-reputed external oracle provider. The latter option is generally more reliable as it allows for the utilization of higher quality data sources, improved reliability and security of data being provided to the DeFi platform.

It is further worth mentioning that oracles are responsible for fetching data from external sources, such as websites, APIs, sensors, and other sources, and then transmitting that data to the blockchain.
Oracles are not the data source themselves. Instead, they act as a bridge between off-chain data and on-chain smart contracts. They typically use APIs, web scraping, or other means to retrieve data from real-world sources and then transmit that data to the blockchain.

To ensure the accuracy and reliability of the data they provide, oracles use various techniques such as consensus-based data verification, multi-sourced data aggregation, and cryptographic proof of authenticity. Chainlink is an example of oracle that provides real-time prices of assets. Chainlink operates a decentralized network of oracles that aggregate and transmit real-time data from multiple sources, including exchanges, market data providers, and other sources. Another example is Band Protocol, which offers a similar service as Chainlink but uses a different consensus mechanism to verify data accuracy.

Types of Oracles

Oracles are mainly of two types: **centralized** and **decentralized**.

Centralized oracles rely on a single entity or node to supply data to the smart contracts. This type of oracle is usually easier to set up and maintain, making it more accessible to developers. However, it also introduces a single point of failure, which is a major security risk. If the centralized oracle fails or supplies bad data, the smart contract may execute an erroneous function that could lead to financial loss or other adverse outcomes.

In addition to the single point of failure risk, centralized oracles are also vulnerable to manipulation and corruption. Since the oracle operator has control over the data feed, they may be tempted to supply biased or false data for personal gain. This creates a conflict of interest and introduces a trust issue between the oracle and the smart contract.

Decentralized oracles, on the other hand, rely on a network of nodes to supply data to the smart contract. This type of oracle is more secure because there is no single point of failure. The data is aggregated from multiple sources, and a consensus mechanism is used to determine the correct value. This approach ensures that the data is accurate and reliable.

Even if some nodes collude to provide biased or incorrect data, the consensus mechanism would detect the inconsistency and would not include the incorrect data in the final value. This is because the consensus mechanism is designed to only consider data that is agreed upon by the majority of nodes, and not by a minority of nodes acting maliciously.

However, decentralized oracles can be more complex to set up and maintain than centralized oracles. They also require a higher level of trust between the nodes that supply the data. If one or more nodes are compromised, they could provide bad data to the smart contract, leading to incorrect outcomes.
In a decentralized oracle, multiple nodes are involved in the data verification process. These nodes use Schelling point game theory to reach a consensus on the accuracy of the data. The Schelling point is a concept in game theory that refers to the focal point or the most likely outcome that individuals will choose in the absence of communication or coordination.

This mechanism provides an additional layer of security by making it more difficult for bad actors to manipulate the data supplied to the smart contract. Even if a few nodes are compromised, the majority - more than 51%- of nodes would still be able to identify the correct Schelling point and reject the bad data.

How do attacks on Oracles take place?
Oracles are essential components in decentralized systems that provide off-chain data to smart contracts. However, they are a point of centralization in the decentralized ecosystem and can be vulnerable to attacks. Here are the common ways attacks on oracles take place:

Data Manipulation: Oracles can be attacked by manipulating the data they provide to smart contracts. An attacker can compromise the oracle's data source or provide false data to the oracle. As a result, the smart contract executes on the basis of the manipulated data, which can lead to financial losses.
Denial of Service (DoS): An attacker can launch a Denial of Service (DoS) attack on the oracle, which can disrupt its normal operation. The attacker can flood the oracle with an overwhelming amount of requests, which can cause the oracle to become unavailable. A DoS attack can prevent the oracle from providing data to smart contracts, leading to financial losses.
Sybil Attack: A Sybil attack involves an attacker creating multiple fake identities to gain control over the oracle. The attacker can use fake identities to manipulate the data provided by the oracle or launch other types of attacks.

How to Prevent Price Oracles Attack
To prevent such attacks, oracles must be sufficiently decentralized to prevent malicious actors from gaining control over them. Also, protocols should implement a mechanism for verifying the data provided by the oracle before executing smart contracts based on that data. Here are some technical aspects related to this:

- **Reputation-based Verification**: Oracle can build a reputation system that tracks the accuracy of its data. Smart contracts can use this reputation score to determine the reliability of the oracle's data. If the reputation score is low, smart contracts can choose not to use oracle's data.
- **Multi-Source Data Aggregation:** Smart contracts can obtain data from multiple oracles to ensure the accuracy of the data. If the data provided by one oracle is corrupted or manipulated, the smart contract can use the data provided by the other oracles.
- **Delayed Execution:** Smart contracts can delay their execution until the data provided by the oracle is verified. This mechanism can prevent the smart contract from executing based on false data.

Spot price manipulation involves the manipulation of the current market price of an asset to influence the price of related financial products, such as derivatives or futures contracts. Indirect price manipulation, on the other hand, involves manipulating the underlying factors that influence the price of an asset, such as supply or demand, to indirectly manipulate the price of related financial products. To prevent spot price manipulation or indirect price manipulation attacks, it is crucial to have a decentralized oracle system that obtains data from multiple sources and verifies the accuracy of the data before providing it to smart contracts. Additionally, protocols should implement a mechanism to detect and prevent Sybil attacks, monitor the reputation of data providers, and prevent denial of service attacks.
In the context of oracles, freeloading and direct price manipulation are two types of potential attacks that can occur on an Automated Market Maker (AMM) system that relies on an oracle to obtain external price information.

Freeloading: Freeloading refers to a scenario in which an attacker takes advantage of a public oracle without providing any compensation in return. In the case of an AMM, an attacker could query an oracle to obtain the current price of a token, and then use that information to trade on the AMM, without contributing to the cost of running the oracle. To mitigate this risk, many oracles charge fees for access to their price feeds, which helps to incentivize honest use of the oracle and support the cost of maintaining the system.

Direct Price Manipulation: Direct price manipulation refers to the scenario where an attacker directly manipulates the price of a token on an AMM, in order to profit from trades that take advantage of the manipulated price. In the context of an oracle, direct price manipulation can occur if the oracle is compromised or if an attacker is able to influence the price feed that the oracle provides to the AMM. An attacker might use a variety of techniques to manipulate the price of a token, such as by spoofing orders or creating false transactions to create the appearance of increased demand for a token. To mitigate this risk, many oracles use multiple sources of price information and employ various security measures to prevent manipulation, such as using tamper-resistant hardware and implementing strict access controls. Additionally, AMMs may use more complex algorithms that are less susceptible to manipulation by attackers.

Checklist to prevent oracle attacks:

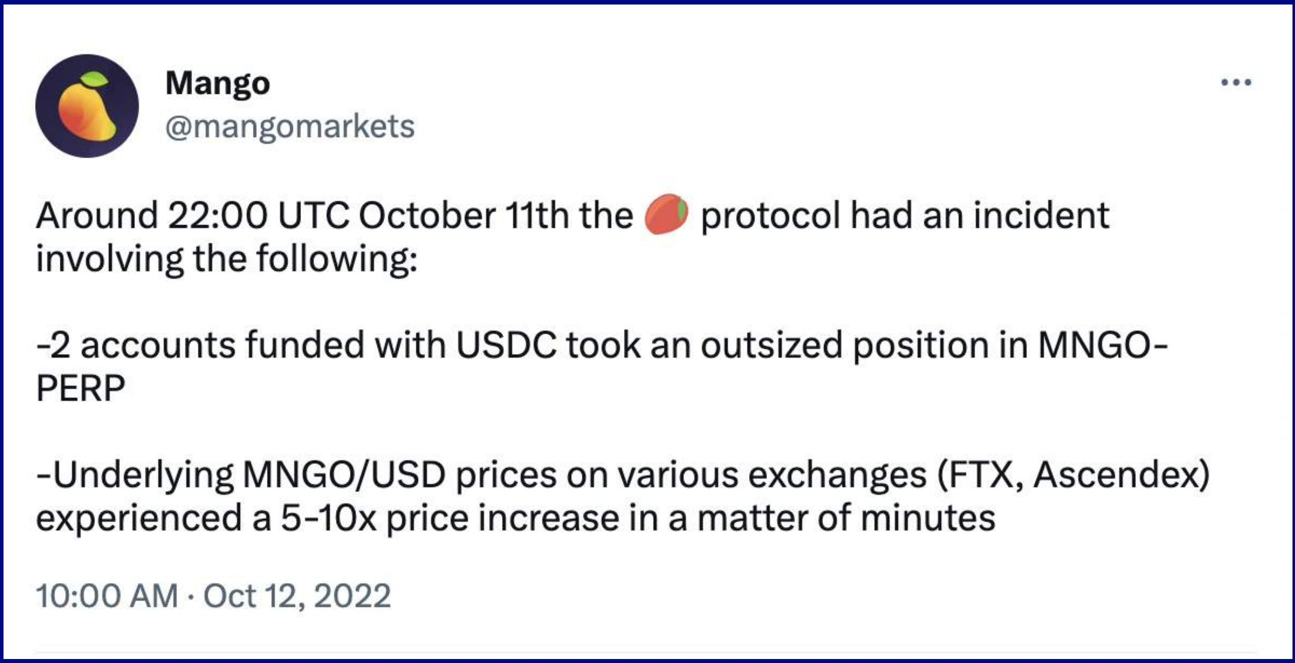
- **Multiple source feeds:** One way to prevent oracle attacks is to use multiple source feeds instead of relying on a single one. This helps to avoid a single point of failure, as well as protect against inaccurate or manipulated data from a single source. For example, aggregating the prices of multiple decentralized exchanges to calculate the overall market price can help prevent price manipulation from a single exchange.
- **Crypto-economic incentives**: Another way to prevent oracle attacks is to use crypto-economic incentives. This can include providing rewards to data providers for accurate data. By aligning incentives with the desired outcome, it can help to deter malicious behavior and ensure the integrity of the data provided.
- **Time-Weighted Average Price (TWAP) mechanism:** A TWAP mechanism is another way to prevent oracle attacks by reducing the impact of short-term price fluctuations. A TWAP calculates the average price of an asset over a specified time period, rather than relying on a single price point. This can help prevent manipulation from a single source or a flash crash that may otherwise affect the accuracy of the data.
- **Preventing flash loan attacks:** Flash loan attacks can be prevented by introducing a delay between the time data is retrieved from the oracle and when it is executed. This delay can help to prevent a flash loan attack by making it more difficult for an attacker to manipulate the price of an asset during the execution period.
- **Outlier management:** Outliers in oracle reports refer to data points that are significantly different from the rest of the data. These can be due to various factors, such as data input errors, data manipulation, or even technical glitches. Outliers can have a significant impact on the accuracy of the data and can be used to exploit the system. Finally, outlier management can help prevent oracle attacks by filtering out anomalous data points. By using statistical methods to identify outliers, such as median absolute deviation or Z-score, it is possible to remove data points that do not conform to the expected distribution. This can help prevent the manipulation of the data by a single source or a malicious actor.

For example, Chainlink, a decentralized oracle network, has implemented outlier management to ensure the integrity of the data provided by their oracles. Chainlink uses a combination of statistical analysis, machine learning, and human review to detect outliers and provide accurate data to its users. Data Provider's Role: Data providers also play a crucial role in ensuring the accuracy of the data provided to oracles. They need to implement outlier detection mechanisms to filter out erroneous data points before providing the data to the oracles. This can help prevent manipulation of the data and ensure that the oracles receive accurate data to provide to the users. Uniswap, a decentralized exchange, for instance, has implemented an outlier detection mechanism to prevent manipulation of their price feeds. They use a three-step process that involves removing data points that are above or below a certain threshold, removing data points that are outside the expected distribution, and using a median value to provide the final price feed to the oracles.

Mango market Exploit

Mango Markets is a decentralized exchange (DEX) that is built on the Solana blockchain and offers spot margin trading with deep liquidity, leveraged derivatives, and risk management tools for traders. Mango Markets allows for spot margin trading using Serum DEX, and their order book provides perpetual futures for those interested. The governance of Mango Markets is conducted through Mango DAO, which is controlled by MNGO token holders. In terms of the mechanics of margin trading, Mango Markets offers up to 5x leverage for margin traders. Traders can use cross-collateralized accounts to borrow assets and trade on the platform. To maintain their position, traders must maintain a minimum margin ratio, which is the ratio of the value of the posted collateral to the value of the borrowed assets. If the value of the posted collateral falls below the minimum margin ratio, the position may be liquidated.

In January 2023, the SEC brought charges against a trader who exploited a token designed to become more valuable the more people used its underlying exchange. The token was used on both Mango Markets and FTX, which ultimately brought down the latter.



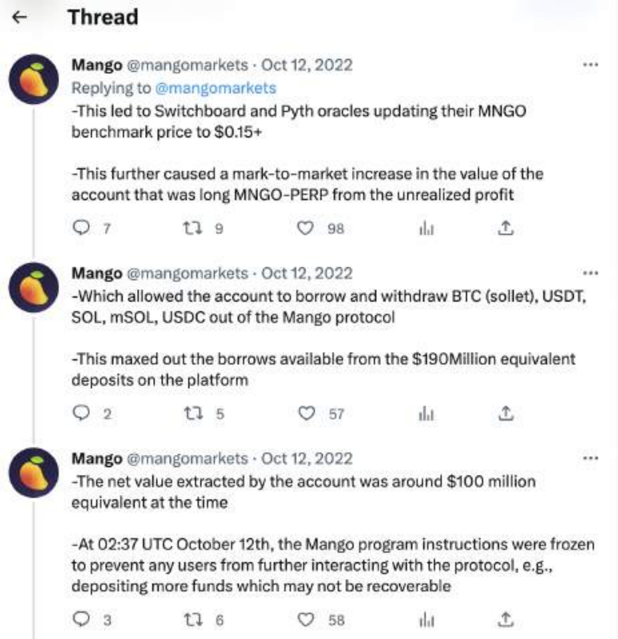
Mango
@mangomarkets

Around 22:00 UTC October 11th the 🥭 protocol had an incident involving the following:

-2 accounts funded with USDC took an outsized position in MNGO-PERP

-Underlying MNGO/USD prices on various exchanges (FTX, Ascendex) experienced a 5-10x price increase in a matter of minutes

10:00 AM · Oct 12, 2022



Thread

Mango @mangomarkets · Oct 12, 2022
Replying to @mangomarkets
-This led to Switchboard and Pyth oracles updating their MNGO benchmark price to $0.15+

-This further caused a mark-to-market increase in the value of the account that was long MNGO-PERP from the unrealized profit

7 9 98

Mango @mangomarkets · Oct 12, 2022
-Which allowed the account to borrow and withdraw BTC (sollet), USDT, SOL, mSOL, USDC out of the Mango protocol

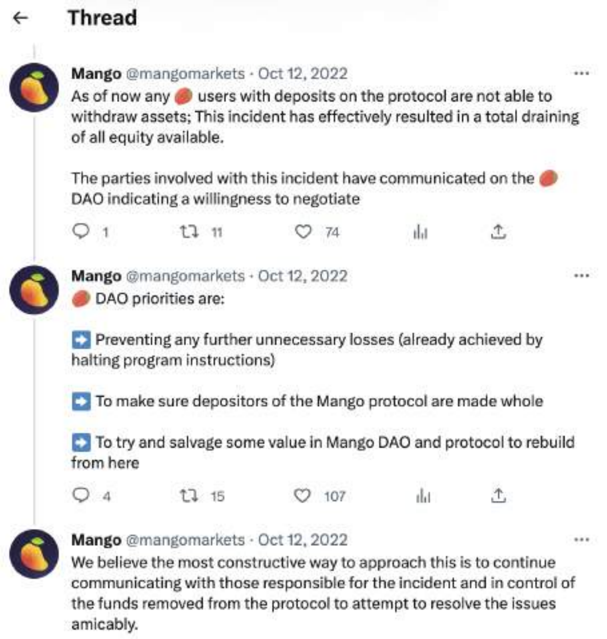
-This maxed out the borrows available from the $190Million equivalent deposits on the platform

2 5 57

Mango @mangomarkets · Oct 12, 2022
-The net value extracted by the account was around $100 million equivalent at the time

-At 02:37 UTC October 12th, the Mango program instructions were frozen to prevent any users from further interacting with the protocol, e.g., depositing more funds which may not be recoverable

3 6 58



Thread

Mango @mangomarkets · Oct 12, 2022
As of now any 🥭 users with deposits on the protocol are not able to withdraw assets; This incident has effectively resulted in a total draining of all equity available.

The parties involved with this incident have communicated on the 🥭 DAO indicating a willingness to negotiate

1 11 74

Mango @mangomarkets · Oct 12, 2022
🥭 DAO priorities are:

➡️ Preventing any further unnecessary losses (already achieved by halting program instructions)

➡️ To make sure depositors of the Mango protocol are made whole

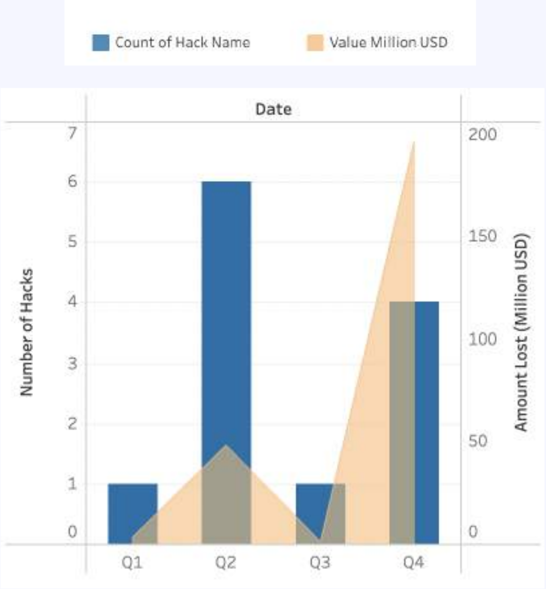
➡️ To try and salvage some value in Mango DAO and protocol to rebuild from here

4 15 107

Mango @mangomarkets · Oct 12, 2022
We believe the most constructive way to approach this is to continue communicating with those responsible for the incident and in control of the funds removed from the protocol to attempt to resolve the issues amicably.

Analysis of the attacks by Mango Market

Hot Wallet Attacks



Over 205 Mil USD were lost due to hot wallets being hacked in the year 2022. Q1 has seen as many as 9 such attacks, although the amount lost is no more than 17 Mil USD. Q4 has seen 2 attacks amounting to about 28 Mil USD and with just 1 attack in Q3, the losses are over 160 Mil.

Hot wallets are used to store and manage digital assets online. On an exchange, a hot wallet facilitates the buying and selling of cryptocurrencies by holding a small amount of the assets in a readily accessible location. However, the tradeoff of this convenience to readily transact in crypto is a plethora of threats. One of the most alarming of which is a hot wallet breach. In 2022 alone, such attacks on insecure online wallets have led to a loss of millions of dollars of users' funds, with names like the Deribit Exchange, and Wintermute being caught in the crossfire. A hot wallet security breach occurs when an unauthorized party gains access to the hot wallet and is able to steal the assets it holds. There are several ways in which hackers can exploit hot wallets to gain access to the assets they hold. Some common methods include:

Phishing attacks: Hackers can use phishing attacks to trick users into providing their login credentials, allowing the hackers to gain access to the hot wallet. In a phishing attack, the hacker creates a fake website or email that is designed to look like a legitimate site or message from a trusted source. The fake site or email may ask the user to enter their login credentials, such as their username and password, or to click on a link that installs malware on their device. Once the user has entered their login credentials or clicked on the link, the hacker can use this information to gain access to the user's hot wallet and steal the assets it contains.

Man-in-the-middle (MiM) attacks: Hackers can use man-in-the-middle attacks to intercept and manipulate transactions, allowing them to steal assets from the hot wallet. In a man-in-the-middle (MiM) attack, the hacker intercepts communications between two parties and tampers with them in order to steal cryptocurrency from a hot wallet. The hacker can perform the attack by setting up a fake network or by compromising a legitimate network and inserting themselves between the two parties as they communicate.

How to make hot wallets more secure
Here are some of the measures that users can take to make their hot wallets more secure.

- **Non-custodial wallets**

One option is to use a non-custodial wallet, which gives the user full control of their crypto wallet keys and removes third-party access. This means that the user is solely responsible for the security of their crypto assets, but it also reduces the risk of a third-party hack. Some of the best non-custodial wallets include Exodus, Electrum, and Ledger Nano X.

- **Practice Due Diligence**

Be cautious when providing personal information or downloading software related to your crypto assets. Always verify the legitimacy of the source and check for reviews or feedback from other users. Additionally, be aware of phishing scams and other online scams that aim to steal your crypto assets.

- **Multi-Factor Authentication**

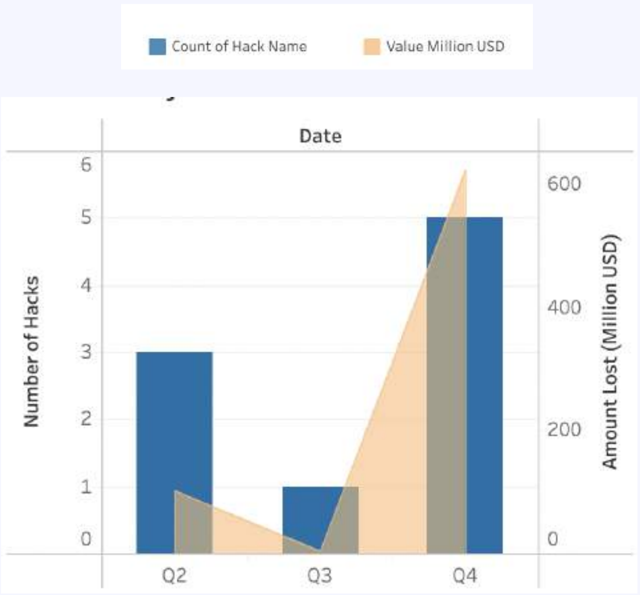
Multi-factor authentication (MFA) adds an extra layer of security to your online accounts. It requires additional verification steps beyond a password to access your account. This can include biometric identification or a one-time code sent to your phone. By using MFA, you reduce the risk of unauthorized access to your crypto assets.

By implementing these security measures, users can reduce the risk of their hot wallets being hacked and increase the safety of their crypto assets.
The Wintermute Hack
Wintermute, a renowned automated market maker, was hacked for $160 million on September 20, 2022. The hackers targeted one of the protocol's hot wallets, most likely because of its security flaw as a Profanity vanity address. Profanity generates vanity wallet addresses. Vanity wallets are unique crypto addresses that feature a string of characters that are simple to remember and recognise. Vanity addresses function similarly to normal addresses, except they include a unique alphanumeric character string, message, or distinctive word for the owner. The primary distinction between a vanity and a standard address is that the former is personalized and immediately identifiable.

A flaw in Profanity's algorithm permitted the Wintermute attack. Unlike other smart contract vulnerabilities, this flaw allowed an attacker to directly target compromised hot wallets. Insider-trading conspiracies were debunked. The protocol indicated that 90 assets were stolen following the hack. Two crypto assets worth around $1 million and $2.5 million, with the remaining tokens for less than $1 million. Upon investigation, it was discovered that an individual had successfully utilized around a thousand GPUs over a period of 50 days to brute-force private keys of every 7-character vanity address. While this requires a significant investment, some cryptocurrency mining farms operate with a larger number of GPUs and may consider cracking Profanity addresses as a means of generating great profits.

Security analysts have recently disclosed a vulnerability in Profanity and reported that attackers have already stolen $3.3 million. They have advised anyone with funds in wallets created with Profanity to move them to a different location immediately. The compromised Wintermute wallet appears to have been generated with a faulty vanity address generator, making the possibility of exploiting the Profanity vulnerability for theft appear likely.

Hacks due to Private Key Leaks



As per our analysis, there have been no less than 9 attacks directly as a result of Private Key Leaks in the year 2022, amounting to a total of 731 Mil USD. With 3 attacks, Q2 witnessed losses amounting to 103 Mil USD and as much as 623 Mil USD were stolen directly as a result of 5 such attacks in Q4.

A private key is a string of numbers and characters used to encrypt and decrypt sensitive data cryptographically. A private key is one of the most important parts of a cryptocurrency wallet, ownership of which assures complete custody over the assets it holds. Using a private key is a safe way to ensure wallet security provided, it is handled with utmost confidentiality and responsibility. The accessibility to a private key opens all doors for hackers to exploit a user's funds. Thus, securing and safeguarding a private key is of paramount importance. This year, attacks on the private keys have led to substantial losses in the crypto sphere. From major exchanges like FTX to popular gaming platforms like Wonder Hero, hackers have been rigorously plundering digital assets by attacking online wallets and gaining unauthorized access to the funds. More than $700 million were lost due to private key leaks

Hot Wallet Hacks and Private Key Leaks are to an extent similar in nature. However, for a better understanding, we have classified such attacks into two different categories, namely, 'Hot Wallet Attacks' in which attackers gain access to an exchange's hot wallet, and 'Private Key Leaks' in which user wallets were specifically targeted by exploiters.

How to mitigate Private Key Breaches:

- **Use of a short-term encryption key**

Using a series of encryption keys, each for a short period of time can reduce the amount of information revealed if a private key is compromised once. Additionally, using simple mnemonic phrases for private keys can also be helpful.

- **When transacting through an exchange, your security is as good as the exchange's**

The large number of users holding millions of dollars of assets on an exchange attracts illicit actors, who are always in search of loopholes to exploit. Therefore, private key security is one of the most critical things to keep in mind before carrying out any activity on the network.

- **Never store private keys on a system that can be easily manipulated**

Online networks are undeniably open to exploits at any given time. This is also true with cryptocurrency wallets and their private keys. To ensure the security of private keys, one should avoid storing them on internet-accessible systems that may be vulnerable to attacks. Instead, using a hardware wallet or other offline sources to store private keys can avoid unnecessary exposure to threats and reduce the probability of an attack.

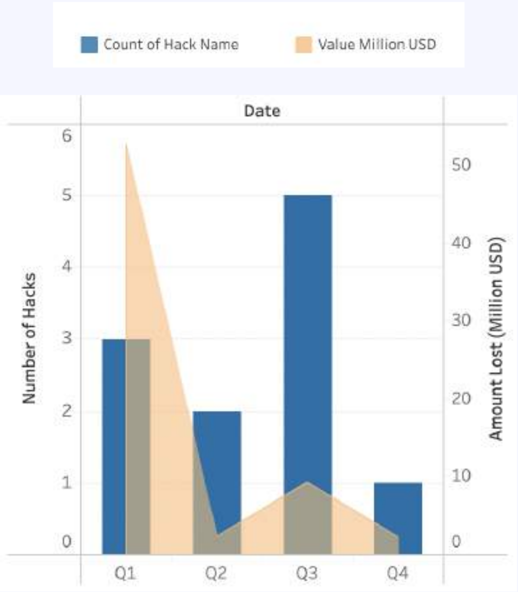
The FTX Exploit

FTX, one of the largest cryptocurrency exchanges in the world, filed for voluntary bankruptcy due to a lack of liquidity on 11th November 2022. Following this, the platform noticed an abrupt draining of funds from the exchange's accounts later that day. Further investigations revealed that more than $477 million were swindled from the exchange's US and international wallets.

The funds were then moved to other exchanges and converted into other cryptocurrencies. The attacker also interacted with decentralized exchanges and aggregators in order to obfuscate trails of the stolen funds.

The FTX representatives claimed this to be an inside job and urged users not to interact with the protocol's website and apps moving forward. The hack also made it to the list of the top ten biggest crypto hacks of 2022 acting as a catalyst for the prevailing recession, and deflation and significantly fueling the market contagion effect in the ecosystem.

Hacks due to other Security Breaches



We have observed about 11 security breach attacks in 2022 where the total loss amount is more than 65 Mil USD. While there were 5 such attacks in Q3 amounting to a total loss of 9 mil USD, in Q1 we have seen 3 attacks amounting to approximately 53 Mil USD.

A security breach is an incident that results in an unauthorized intervention or access to confidential or personal data, applications or networks. Depending upon the nature of the incident, security breaches can be anything from low to high or critical risk.

This year, hackers have swindled large amounts of users' funds through security breaches like BGP hijacking, frontend attacks and other such attacks that target a protocol's infrastructure by redirecting users to malicious sites and misleading them to lose their assets to illicit actors. In total, such security breaches have amounted to a loss of at least $66 million worth of digital assets.

Hackers routinely try to manipulate online servers to interrupt services or exploit flaws in network infrastructures. The year started with a massive breach in Crypto.com's online infrastructure that left $34 million worth of users' funds stolen from over 400 cryptocurrency wallets. Later, one of the largest decentralised exchanges on Cronos 'MM Finance' suffered from a front-end attack that allowed hackers to siphon over $2 million worth of digital assets. Following this, Kyber Network, a multichain DeFi platform was attacked in September, in which hackers exploited security gaps in the website's frontend, leading to a loss of $265,000 worth of users' funds.

Such attempts suggest that the industry is witnessing new forms of security threats that are capable of causing enormous damage to this sector, a major part of which goes unnoticed due to the lack of a standardised security infrastructure. These incidents emphasize the importance of taking appropriate security measures and incorporating safety standards like security training and regular audits that can help reduce the probability of such attacks.

Some major infrastructure attacks we witnessed this year include:

Frontend Attacks - A frontend attack is one in which an attacker injects code into a web application that leads to malicious activities when accessed by a user.
BGP Hijacking - BGP hijacking or IP hijacking is the illegitimate takeover of groups of IP addresses by corrupting Internet routing tables maintained using Border Gateway Protocol (BGP). It happens when hackers maliciously reroute internet traffic by falsely depicting to have ownership of a group of IP addresses that they do not actually own or control.

What happens when BGP is hijacked?

As a result of BGP hijacking, the traffic is rerouted to a wrong destination or website, which can be monitored or intercepted. On the users' side, page loading time may increase as that is one of the first signs of a BGP attack

How to prevent BGP hijacking?

Mutually Agreed Norms for Routing Security -
Protecting organizations against BGP hijacking is difficult. It requires vigilance, information exchange and best practices to be shared across the globe. This is where Mutually Agreed Norms comes into play. It is a global initiative, supported by the Internet Society, that provides fixes to reduce the most common routing threats.

Mutually Agreed Norms are built on four pillars:

- **Filtering**:
  - Making sure your and your customers' routing actions are correct.
- **Anti-spoofing:**
  - Enabling source address validation prevents spoofed packets from entering or leaving your network
- **Coordination**:
  - Facilitating global operational communication and coordination between network operators.
  - The Network operator maintains globally accessible up-to-date contact information.
- **Global validation**:
  - Publishing your data, including your routing policy and prefixes you intend to advertise, so your routing information can be validated by third parties.

DNS Attacks - A DNS attack is a cyberattack in which the attacker exploits vulnerabilities in the Domain Name System. The purpose of the Domain Name System is to translate user-friendly domain names into machine-readable IP addresses, via a DNS resolver.

These attacks target the availability and stability of a network's DNS service. Through this attack, hackers redirect users to a website different from the one they intend to visit, usually to steal their personal and confidential data.

How to prevent DNS attacks?

- Ensure that your router DNS is secure.
- Change your router's default login credentials.
- Install anti-malware protection that can automatically detect and notify you about all malicious activities on your network.

Crypto.com Hack

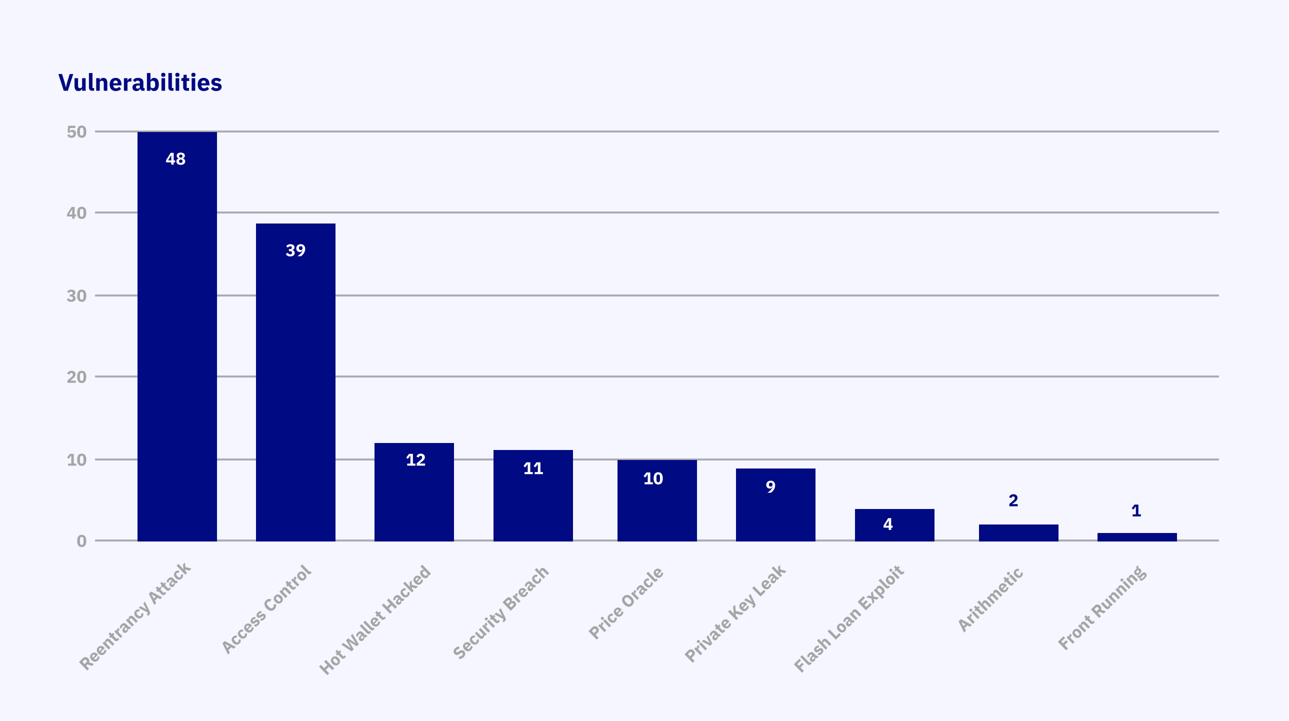
On January 17th, 2022, Crypto.com, one of the largest cryptocurrency exchanges was hit by an attack leading to a series of unauthorized withdrawals. The incident affected at least 483 user wallets with a total loss of more than $34 million.

Crypto.com manages users' private keys for their cryptocurrency wallets. Exchanges like these allow users to log in to their wallets using a username and a password, and for increased security, a 2FA code.

In the attack, the platform identified multiple unauthorized withdrawals on user accounts without the 2FA authentication input by the user. Further investigations revealed that the attacker exploited a vulnerability in the platform's security infrastructure that enabled them to completely bypass the 2FA authentication. As a result, the additional security provided by 2FA was rendered and the wallets were only protected by a password.

By taking advantage of the 2FA bypass, the attackers swindled approximately 4836 ETH ($15 million), and 443 BTC ($19 million). These tokens were then sent to Tornado Cash, which made it impossible to further trace the funds.

Web3 Hacks: 2022 In a Glance



Prevalence of reentrancy attacks

Of the attacks we studied, attacks exploiting reentrancy vulnerability were the most prevalent and preferred means by the attackers, with **48 separate incidents** throughout 2022. These attacks were also one of the most diverse in their approach, targeting everything from aggregators and DeFi projects to marketplaces and gaming platforms. The three most common targets exploited using reentrancy vulnerability were lending protocols, bridges, and streaming services, which had **22, 12, and 8 attacks** respectively.

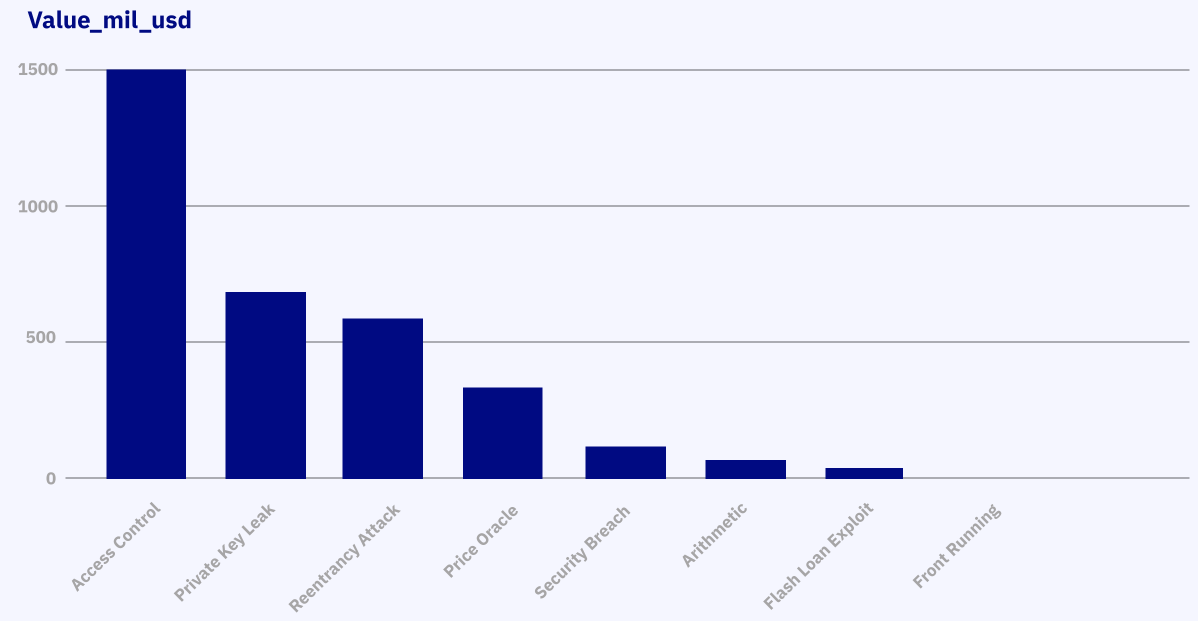
The XCarnival hack in June 2022 is an example of a reentrancy vulnerability exploit against a lending protocol. The XCarnival platform, users to deposit NFT as collateral to take out loans, which produces an orderID. One vulnerability was that the orderID remained open even after the collateral was withdrawn. An attacker put up collateral and withdrew it on a different contract that was then returned to the main contract for use on subsequent contracts. Because there was a valid orderID, the attacker was able to siphon off Ethereum equivalent to $3.8 million.

The Tinyman attack in January 2022 is an example of a reentrancy vulnerability exploit against a bridge. In keeping with the Tinyman Bridge protocol when a user recalls the protocol’s burn function, they are supposed to receive two different tokens, gobtc and ALGO. A flaw, however, enabled users to receive the same token, gobtc. Because gobtc is more valuable than ALGO, attackers used this vulnerability to create an arbitrage opportunity, one that saw them walking away with $3 million.

A successful reentrancy attack is often performed by exploiting a vulnerability in the code that allows the attacker to withdraw funds prior to the smart contract updating the balance of the targeted account. In this type of attack, malicious actors use the same code multiple times to make unauthorized transactions. The way it works is the malicious actor sends a transaction to the smart contract, and then sends a second transaction before the contract can finish processing the first transaction. Because the second transaction requests the same data from the smart contract as the first, it causes the contract to process the same code multiple times.

Reentrancy vulnerability exploits this ability in smart contracts by using a function to call an untrusted function, which in turn makes a recursive call to the original function. As the smart contract does not update the state in real-time (i.e. the orderID showing that the NFT collateral was already withdrawn in the xCarnival hack), the attacker can withdraw as much as they please before it runs its course.

Cost of access control hacks



While access control was the second most prevalent type of attack in 2022 with 39 incidents, just behind the 48 recorded reentrancy attacks, it was the most costly. Access control attacks accounted for US $1.5 billion in losses, or more than reentrancy attacks, price oracle, security breaches, arithmetic, and flash loans all combined.

The lucrativeness of access control hacks is visible through the second largest of its kind in 2022, which yielded attackers of the wormhole token bridge an astonishing $320 million. Here, the Solana blockchain, the instruction sysvar account contains all the instructions for the transaction being processed, which allows the program to obtain signatures from prior instructions. According to Kudelski Security Research, the load_instruction_at function did not check the validity of the sysvar account. The attackers took advantage of this vulnerability by creating a fake sysvar account that requested the creation of 120,000 wrapped Ethereum. Because the attacker had spoofed signatures that were marked true, the consensus needed for validator action approval (VAA) was achieved. The 120,000 wETH was subsequently minted and withdrawn into the attacker's account.

Wormhole Token Bridge Exploit is a classic example of an access control exploit. Due to vulnerabilities in visibility settings - such as the use of delegatecall in proxy libraries - an attacker can also initialize a function, instead of the smart contract owner. Thus, the attacker gained the designated privileges, such as the minting and withdrawal of 120,000 in the wormhole token bridge attack. Bridges are a frequent target for access control exploits as they require more interaction and contract approvals, giving attackers more opportunities to capitalize upon vulnerabilities (more on this in Key Takeaway Number 4).

Access control attacks are so lucrative because they give attackers complete power in using the smart contract for how they want. The situation is roughly analogous to the difference between a fake invoice sent to a business and a theft of their actual credentials. In the first example, the business may inadvertently send the invoice amount to a third party. In the second example, the attackers can do as they please with the corporate bank account, such as withdrawing to the daily limit multiple times over the course of a week (assuming there are no guardrails in place, like two-factor authentication, for large withdrawals). Access control, in effect, is a wide-open door into a person's virtual vault.

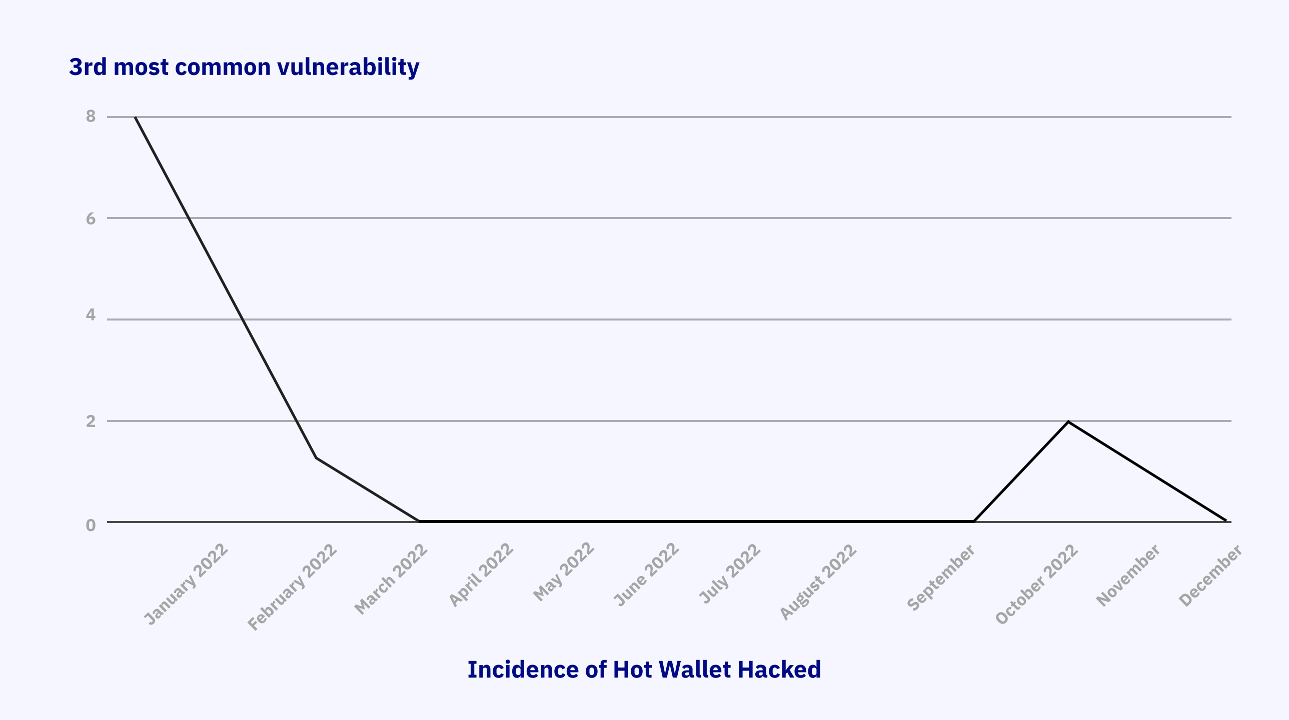
Hot wallet hacking goes cold

From a high of eight incidents in January 2022, hot wallet hacks dropped to zero from March 2022 to September 2022, before seeing a small bump once again in October and November 2022. By this measure, the prevalence of hot wallet hacking seems to be falling dramatically. Why is hot wallet hacking going cold?

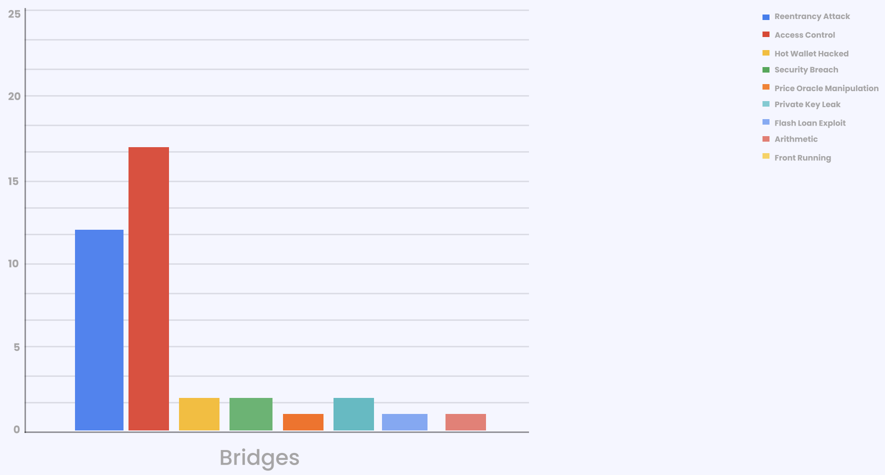
The easy answer is to look at the victims. Most hot wallet attacks were committed against decentralized exchanges, who not only have the impetus to prevent future attacks but can marshal the resources to put these short- and long-term plans into action.

Although Deribit, an exchange focused on cryptocurrency options and futures, stores most of its funds in a cold wallet, hackers were able to gain access to its hot wallet server in November 2022. From there, the attackers were able to initiate withdrawals amounting to $28 million. In response, the company stopped withdrawals temporarily as it investigated how attackers were able to infiltrate its wallet server. Similarly, after the exchange LCX lost $6.8 million from its hot wallet, the company suspended all inflows and outflows and enacted greater security measures around its funds in the Ethereum network.

The decline in hot wallet attacks can be attributed to the industry-wide response to them. More exchanges are storing their funds in cold storage; enabling a zero-trust environment, which often has two-factor or even multi-factor authentication to ensure only permissioned actors are accessing funds; and greater market education of exploits that could lead to hot wallet access, so fewer people inadvertently facilitate hacks. The fall of hot wallet attacks hopefully continues as a trend well into 2023 and beyond.



Breaking bridges



Of all the platforms we studied, bridges were the most commonly attacked. They suffered 12 reentrancy attacks, 17 access control attacks, 2 hot wallet attacks, 2 security breaches, 1 price oracle manipulation, 2 private key leaks, 1 flash loan exploit, and 1 arithmetic exploit. No other platform was attacked as widely as bridges.

While the functionality of bridges varied - there were reward pools, liquidity pools, DeFi stablecoin platforms, DeFi staking platforms, crypto token launchpads, cross-chain bridges, and AMM bridges - the reasons for their vulnerability are universal. To understand this vulnerability, it is first important to understand what exactly constitutes a bridge. The functionalities mentioned above, after all, span a wide variety of use cases.

A crypto bridge enables one blockchain to be interoperable with another, so that people may exchange data or assets between them. Building a crypto bridge is a complex undertaking, requiring a deep understanding of at least two blockchains, which may be vastly different from one another. Because a bridge connects multiple blockchains, there are more touchpoints, each of which represents a potential vulnerability.

Bridges, in other words, have a much broader technical infrastructure that needs to be protected. If you likened them to the real world, a bridge would be like a nation’s borders that span thousands of miles, multiple terrain types, and abutted many different countries. That kind of breadth - in the real world or in the digital - is difficult to defend.

Because of these technical challenges, many developers may successfully build a bridge, but fail to account for key vulnerabilities related to privacy or security. In other cases, developers may be aware of possible issues, but may not have the resources, such as a large development team, to fix them in a timely manner.

Attackers may prioritize searching for these exploits in bridges because of their purpose. They commonly transfer high-value assets from blockchain to blockchain, such as cryptocurrency or non-fungible tokens. Identifying vulnerabilities in these bridges before developers do will give attackers access to these funds, which can then be stolen, laundered, or exploited in some other way.

This is evident in the fact that attackers prioritized bridges that facilitated the exchange of value, rather than just information. To this end, decentralized finance (DeFi) protocols were the most commonly attacked, and within that category, bridges were the most common target. In this view, bridges would be roughly analogous to robberies in the real world: It’s easier for criminals to target the armored cars transporting money (i.e. the bridges) rather than banks themselves (i.e. the blockchains) to get away with large funds.

Involvement of North Korea

One of the most popular destinations for dark tourism in the world is North Korea. The tourists who have leaked photos and footage of the secretive state have documented a country at a standstill, reflecting what South Korea looked like more than several generations ago.

Even in the North Korean capital of Pyongyang, the streets have few cars. Stores and other public establishments have so few shoppers that actors are brought in to portray a more prosperous economy to outsiders. Apart from statues that honour the North Korean leadership, the infrastructure across the country is poorly developed.

These images would make anyone think that the constant sabre-rattling from North Korea is and will always remain empty posturing: The country just does not have the power to deliver on its threats. While North Korea may not have the military might fight its enemies in a conventional war, there is one sphere where the rogue nation stands on equal footing: cyberspace.

Across all the data of hacks from 2022, North Korea was one of the most frequently cited attackers. Attacks by North Korea were typically accomplished through Lazarus Group, its cybercrime group. The rise of the Lazarus Group as a global cybercrime force in 2022 owes as much to political as to technological reasons.

Due to political sanctions and a poor production base, North Korea has few trade partners. Over 90% of its **products are exported to China**, with the remaining going to a hodge-podge of other nations. Because North Korea has few opportunities for global trade, the nation needs more means to generate revenue to support state initiatives. With the advent of cryptocurrency, cybercrime has presented this exact opportunity.

Take the case of the Axie Infinity hack, an attack for which the Lazarus Group was heavily implicated. An email was sent to an engineer at Sky Mavis, the developer behind Axie Infinity, purporting to be from another firm that wanted to poach him with a much more lucrative offer. These communications were actually spear phishing attacks from the Lazarus Group.

What was novel about this attack was that the Lazarus Group did not send the corrupted files immediately, but waited until after an interview with the developer. After gaining his trust, the Lazarus Group then sent over a fake job description, which the developer opened, compromising his computer and giving the attackers access to the organization’s network. Through this entryway, the Lazarus Group eventually made off with $600 million.

While the United States government eventually recovered $30 million from the hack, the bounty of over a half billion dollars was well worth the effort. This attack underscores the effectiveness of cybercrime as an asymmetric form of revenue generation: With just a small group of attackers, the North Korean government can obtain a disproportionately large amount of funds to sponsor state activities. Before exploring the hacks that amounted to a loss of $3.9 billion this year in greater detail, let's examine hacks at a high level throughout each quarter.

Conclusion

The state of crypto crime is constantly evolving, and it is important to remain vigilant in order to protect against emerging threats. As demonstrated in our report, the crypto ecosystem suffered a loss of nearly $3.9 billion. Cybercriminals are increasingly using emerging technologies like DeFi to launder illicit funds, and exploiting vulnerabilities in the system such as reentrancy attacks, flash loans, and price oracle manipulation, arithmetic vulnerabilities, and access control vulnerabilities.

Moreover, the analysis of the hacks make it evident that along with ETH smart contracts, BNB chain was exploited by hackers significantly especially in single chain attacks. The patterns that have been followed by hackers to exploit vulnerabilities in these two chains are highly similar. Both chains were majorly exploited using reentrancy, access control, private keys and flash loan attacks. The situation however differs for BTC, as in comparison to ETH chain, it was negligible. The reason is that in 2022, the point of vulnerability remained decentralized platforms.

Therefore it is necessary to understand how these vulnerabilities are exploited and why significant security updates are required to make the crypto ecosystem secure and more resilient to attacks.

The fact that hackers are coming up with new vectors to inflict damage to decentralized platforms, it raises some important concerns- the use of due diligence and stringent means to develop secure smart contracts.

These contracts need to be validated and verified by multiple auditing practices including both manual and tool-based code analysis. Furthermore, the decentralized platforms can leverage the services of third-party auditors as well to improve the efficiency and security of the deployed smart contracts.

To ensure a robust verification and validation of smart contract audit, it is recommended to utilize semantic, syntactic and run-time monitoring auditing techniques in combination with the best industry practices. Semantic and syntactic audits safeguard against the need for redeployment of the smart contract with now a different address while run-time validations help in executing correct code logic. Nonetheless, some of the best practices include:

Manual auditing entails a team of smart contract developers and auditors analyzing every single line of code for compilation and errors. Furthermore, this strategy assists in identifying other undetected security vulnerabilities such as Logic errors, Backdoors and Design flaws.

Besides manual code analysis, the tool-based analysis of a smart contract is also a method to examine a set of code. Furthermore, the tool-based analysis enables comprehensive penetration testing, which aids in the rapid discovery of vulnerabilities that might be missed while performing manual analysis. As smart contracts can be complex and have varying computational requirements, measuring the gas cost required to execute the smart contract is one way to assess its efficiency. However, the accuracy of measuring gas cost is dependent on the code implementation, and some developers may implement contracts in more efficient ways than others, making it difficult to use as a definitive measure of efficiency.

To ensure the security of smart contracts, a security audit is usually conducted before the code is hosted on the Ethereum platform. Standards and protocols such as the Solidified Verification Standard can be used to execute smart contract security audits. Such audits can help improve the security of the entire Ethereum ecosystem and individual projects, and help block potential hacks or vulnerabilities.

Along with these practices, the following techniques are highly recommended;

Static analysis helps to analyze the source code of a smart contract without executing it. The goal of static analysis is to detect any potential vulnerabilities, bugs, or security issues that might exist in the code. The static analysis can be done using different tools that can identify code patterns that are associated with common vulnerabilities.
Example: One example of a static analysis tool is Mythril. It uses symbolic execution to analyze the smart contract's code for potential security issues. Mythril can detect vulnerabilities like reentrancy, integer overflow, and underflow, and it can also detect smart contracts that have been plagiarized.

Besides static analysis, Fuzzing, also known as fuzz testing, is used in smart contract auditing to test a smart contract by providing it with random or invalid inputs to identify any unexpected behavior or vulnerabilities. The goal of fuzzing is to ensure that the smart contract can handle unexpected or invalid inputs gracefully, without causing any security issues.
Example: One example of a fuzzing tool is Echidna. Echidna is a smart contract fuzzer that can generate test cases for smart contracts automatically. It can test the contract with different inputs and values and detect vulnerabilities like reentrancy, assertion failures, and transaction order dependence.

Furthermore, Formal verification is used to verify that a smart contract meets a specified set of requirements or specifications. The formal verification process involves mathematically proving that the smart contract behaves as intended under all possible scenarios and inputs.
Example: One example of a formal verification tool is the K framework. The K framework uses mathematical logic to model the behavior of smart contracts and can verify the correctness of the contract's behavior. It can also generate test cases and identify any potential vulnerabilities.

Mitigating these risks requires a multifaceted approach, and we recommend the use of various techniques such as automatic and manual testing as well as static analysis, fuzzing, and formal verification. Static analysis can help to identify potential security vulnerabilities in smart contracts and code by analyzing it without executing it. Fuzzing can be used to test the robustness of smart contracts by generating random inputs and observing how the contract behaves in response. Formal verification can help to mathematically prove the correctness of smart contracts and code, providing a high level of assurance that they are free from vulnerabilities.

By utilizing these mitigation techniques, we can better protect against crypto crime and ensure the continued growth and development of the blockchain ecosystem. However, it is important to note that these techniques are not foolproof and must be used in conjunction with other best practices such as secure coding, ongoing monitoring, and incorporating due diligence.

Moreover, from the user side, the users should make sure that the smart contract is verified and validated thoroughly before performing any transaction. This needs creating awareness among users where they are knowledgeable and understand the vulnerabilities and threats a particular smart contract poses.

Methodology

DASP-Top10, a document published by the Decentralized Application Security Project(DASP), defines the top 10 vulnerabilities that exist in smart contracts. We refer to DASP-10 categories to classify the hacks that we have come across thus far.

Based on secondary research such as articles, news and social posts released by experts of the industry, representatives of the hacked platforms, Merkle Science's in-house investigations as well as running the smart contract vulnerability scanners, we were able to classify the hacks and abuses of the year 2022 in the 2 broad categories of

- **SC Vulnerability** - where the hacks/thefts/abuses have occurred directly and solely due to the exploitation of a smart contract vulnerability.
- **Other** - Where the hacks/thefts/abuses have occurred either
    a. due to other security gaps (Hot Wallet Hacked, Security Breach)
    b. Indirectly due to a smart contract vulnerability which has resulted in other forms of exploitation. (Flash Loan, Price Manipulation)

SC Vulnerability/Other	Subcategory
SC Vulnerability	Access Control
SC Vulnerability	Reentrancy
SC Vulnerability	Arithmetic
Other	Flash Loan Exploit
Other	Hot Wallet Hacked
Other	Price Oracle Manipulation
Other	Private Key Leak
Other	Security Breach

Other type of attacks in 2022

- **Flash Loan Exploit:** Flash loan exploits usually occur due to smart contract vulnerabilities. However, since the attacks affect a different bridge/platform instead of the lending platform, these attacks have been classified as Flash Loan Exploits.
- **Price Oracle Manipulation:** Smart contract vulnerabilities facilitate borrowing of an unsecured flash loan and the flash loan in turn facilitates the attacker to manipulate the price of a token. However, since the final cause of the exploit is a price manipulation of tokens in one way or another, we have categorized such hacks as Price Oracle Manipulation.
- **Hot Wallet Hacked & Private Key Leak:** Both Hot Wallet Hacks and Private Key Leaks are similar in nature. However, if an attacker gains access to the hot wallet of an exchange, we have classified such attacks and exploits as 'Hot Wallet Hacked' whereas if the private keys of user wallets are taken over, we have classified such attacks as 'Private Key Leaks' While both Hot Waller Hacked and Private Key Leaks are a type of access control, we have specifically segregated them for ease of understanding.
- **Other Security Breach:** All other non smart contract related security and infrastructural breaches are classified under 'Other Security Breach', which are inclusive of but not limited to 2FA failure, Frontend manipulation and fake account creation.

As per our investigation, some hacks could be a combination of 2 or more categories/ subcategories of hack classification, however, we have accredited the final cause of the hack as the basis of classification. For example, the Inverse Finance Exploit was facilitated by a smart contract vulnerability which allowed the attacker to avail massive unsecured flash loans, the actual hack was resultant of the price manipulation executed by manipulating the price of a token through purchasing a large volume of the said token using the aforementioned flash loans. In this scenario, since the end reason of the exploit was the manipulation of prices, we have categorized the reason for Inverse Finance Exploit as a Price Oracle Manipulation

Smart Contract Vulnerability

Within the smart contract vulnerabilities, DASP has categorized smart contract vulnerabilities into the following subcategories:

- **Reentrancy -** A malicious external contract executes a recursive call on the victim smart contract in a virtually endless loop until all the funds held in the victim contract is drained out.
- **Access Control** - A vulnerability that allows an attacker to manipulate the validation/verification step thereby replacing ownership of the contract.
- Arithmetic - Integer overflow/underflow that is a resultant of the use of unsigned integers in the smart contract code.
- **Unchecked Low-Level Calls** - This occurs when the creator fails to check the validity of low-level calls that can result in unexpected behaviour in the program logic.
- Denial of Services - When an attacker can potentially effectively block the functioning of a smart contract
- **Bad Randomness -** Occurs when a smart contract uses randomness, it is often predictable. Hence a miner or attacker can guess, crack/manipulate this randomness to attain favourable outcomes.
- **Front Running -** Also known as transaction order dependence. It occurs when an attacker manipulates the order of transactions to enable the transaction of interest to occur before. This is done usually by setting a higher gas fee.
- **Time Manipulation** - a malicious miner can report an incorrect time of the transaction so that the transaction ends up in the block he is mining to attain favourable results
- **Short Addresses** - When an attacker bypasses the required parameter with a shorter than expected parameter, however, the Ethereum Virtual Machine appends ‘0’ at the end thereby facilitating a bypass of the authentication process.10. Unknown-Unknown: Any other smart contract vulnerability can be classified under unknown.

Although for the hacks that have occurred through the year 2022, we majorly witnessed 4 out of the 10 vulnerabilities, in the future hacks can happen that exploit the remaining vulnerabilities.

Glossary

Access control - Access control issues result from poor visibility settings, such as the use of tx.origin for validating callers or delegatecall in proxy libraries. Due to these vulnerabilities, an attacker can start the initialize function, not just the smart contract owner. Upon doing so, the attacker can perform designated privileges, such as withdrawing to their account, even if the function has already been called.

Arithmetic - This exploit is also known as integer overflow and integer underflow. An uint is an unsigned integer. A uint8 of 8 bits can store integers that range from 0 to 255. Trying to store a value larger than 255 will result in overflow; decrementing 1 from 0 will revert back to 255, which lead to underflow.

Underflow and overflow seem harmless on their own, but these patterns become serious vulnerabilities when they relate to wallet balances. Since debiting a wallet will always result in a positive number, an attacker who discovers this exploit will be able to withdraw more and more coins.

Flash loan exploit - Flash loans take advantage of the growing popularity of lending in decentralized finance (DeFi). Through smart contract protocols, flash loans enable borrowers to complete a three-step process in a single transaction, including the borrowing, use of funds, and repayment.

True to the name of this scheme, the attacker, who is highly capitalized, will quickly borrow a large amount of crypto through loans that are not collateralized. The goal is to manipulate the price of the crypto asset so that it can be sold on another exchange. The attacker gains value from the arbitrage that they themselves initiated with the outsized loan.

Front running - Front running attacks are led by bots. These bots will scan blockchains for pending transactions for significant trades that can affect the price of a crypto asset. When it finds one, the attacker will place a transaction with network fees larger than the target transaction, so that miners process theirs first. In this way, the attack “front runs” the other transaction in as little as milliseconds, enabling them to profit from the original trader.

Hot wallet attacks- A cold wallet is one that is not offline, such as a hardware wallet. A hot wallet is one that is connected to the internet, such as one held on an exchange. The problem with hot wallets is that the connection to the internet exposes them to more cybersecurity threats. When it comes to exchanges, both the hot wallets of users and those belonging to the organization itself can be hacked. An attacker, for instance, may use phishing emails to get access to a person’s laptop, which they can then use to hack the person’s hot wallet.

Price Oracle Manipulation - Oracles are third-party service providers that provide data - most commonly price feeds - to blockchain in one of two ways: either they get data from centralized exchanges via application performance interface (API), or perform calculations themselves using data from decentralized exchanges. Oracles are necessary because smart contracts are built to execute code, but not interface with the real world.

A price oracle manipulation is when protocols use data from oracles that has been somehow manipulated. As a result of the incorrect data, an attacker can liquidate funds, capitalize on arbitrage opportunities, or cause other damages.

Private Key Leak - Private keys of both individuals and enterprises like exchanges can be hacked. These private keys can be obtained in a variety of nefarious ways, such as spear phishing (in which an attacker sends a targeted email to an exchange employee, a known whale, or some other person or entity with major crypto holdings), ransomware (in which a person's information or data is held hostage in exchange for a ransom), or other types of malware. Once obtained, hackers can use private keys to drain a wallet of funds.

References

- SWC-107 · Overview. (n.d.). SWC Registry.
- Security Considerations — Solidity 0.8.17 documentation. (n.d.). Solidity.
- Solidity Data Types: Signed (int) and Unsigned Integers (uint). (2022, October 4). Alchemy.
- Taylor, R. (2022, April 23). Aku Dreams NFT Project Akutars Loses $34M in ETH to Code Bug. This Morning On Chain.
- What Is Crypto Bridging? A Guide to Cross-Chain Bridges, 2022
- Consensus.Bithub - Reentrancy - Ethereum Smart Contract Best Practices.
- iCommunity Labs - 5 Key Vulnerabilities of Smart Contracts, 2023, February 7 iCommunity Labs.
- Rachit Agarwal, Tanmay Thapliyal, Sandeep Kumar Shukla, “Vulnerability and Transaction behavior based detection of Malicious Smart Contracts”, arXiv, pp 1- 10, 06/2021
- BeInCrypto. (2022, May 1). DeFi Exploits Continue to Plague Industry as Saddle Finance Hack Sees $10M Stolen.
- Cointelegraph. (2022, April 25). AkuDreams dev team locks up $33M due to smart contract bug. Cointelegraph.[11] DeFi Exploits Continue to Plague Industry as Saddle Finance Hack Sees $10M Stolen. (2022, May 1).
- Ethereum Stack Exchange. (2019, March 23). How much data can I store in a smart contract, what is the cost and how it is implemented? Ethereum Stack Exchange.
- Feedalpha. (2021, December 31). Understanding The Different Facebook Page Roles - feedalpha | Social Media Automation. Feedalpha.
- GeeksforGeeks. (2022, May 11). Solidity - Constructors. GeeksforGeeks.
- Hack Analysis: Saddle Finance, April 2022 | by Immunefi | Immunefi. (2022, November 17). Medium.
- Halborn. (2022, January 17). How Centralization Enables Smart Contract Hacks and Scams. Halborn.
- Halborn. (2022, September 29). Explained: The Boy X Highspeed (BXH) Rug Pull (September 2022). Halborn.
- Levalle, Y. (2022, May 10). Post. Post | Dreamlab Technologies.
- Marx, S. (n.d.). SWC-101 · Overview. SWC Registry.
- National Geographic. (2022, May 19). Y2K bug. National Geographic Society.
- OpenZeppelin. (n.d.). Access Control. OpenZeppelin Docs.
- Parity. (2017, July 20). The Multi-sig Hack: A Postmortem. Parity Technologies.
- Robertson, A. (2022, March 29). Hack steals $625 million from NFT game Axie Infinity’s Ronin blockchain.
- The Verge. Saddle Finance - REKT 2. (2022, May 1). Rekt.news.
- SecureFlag. (n.d.). Broken Authentication Vulnerability | SecureFlag Security Knowledge Base. SecureFlag Knowledge Base.
- Solidity Data Types: Signed (int) and Unsigned Integers (uint). (2022, October 4). Alchemy.
- Taylor, R. (2022, April 23). Aku Dreams NFT Project Akutars Loses $34M in ETH to Code Bug. This Morning On Chain.
- Thomas, L. (2022, April 25). $34M Locked in a Smart Contract. Was the Akutars Exploit Avoidable? NFT Now.
- Van Boom, D. (2022, July 6). A Fake Job Offer Reportedly Led to Axie Infinity's $600M Hack. CNET.

Contact us

✉ contact@merklescience.com 🌐 www.merklescience.com

Follow Us

in merklescience

@MerkleScience

f merklescience

Merkle Science