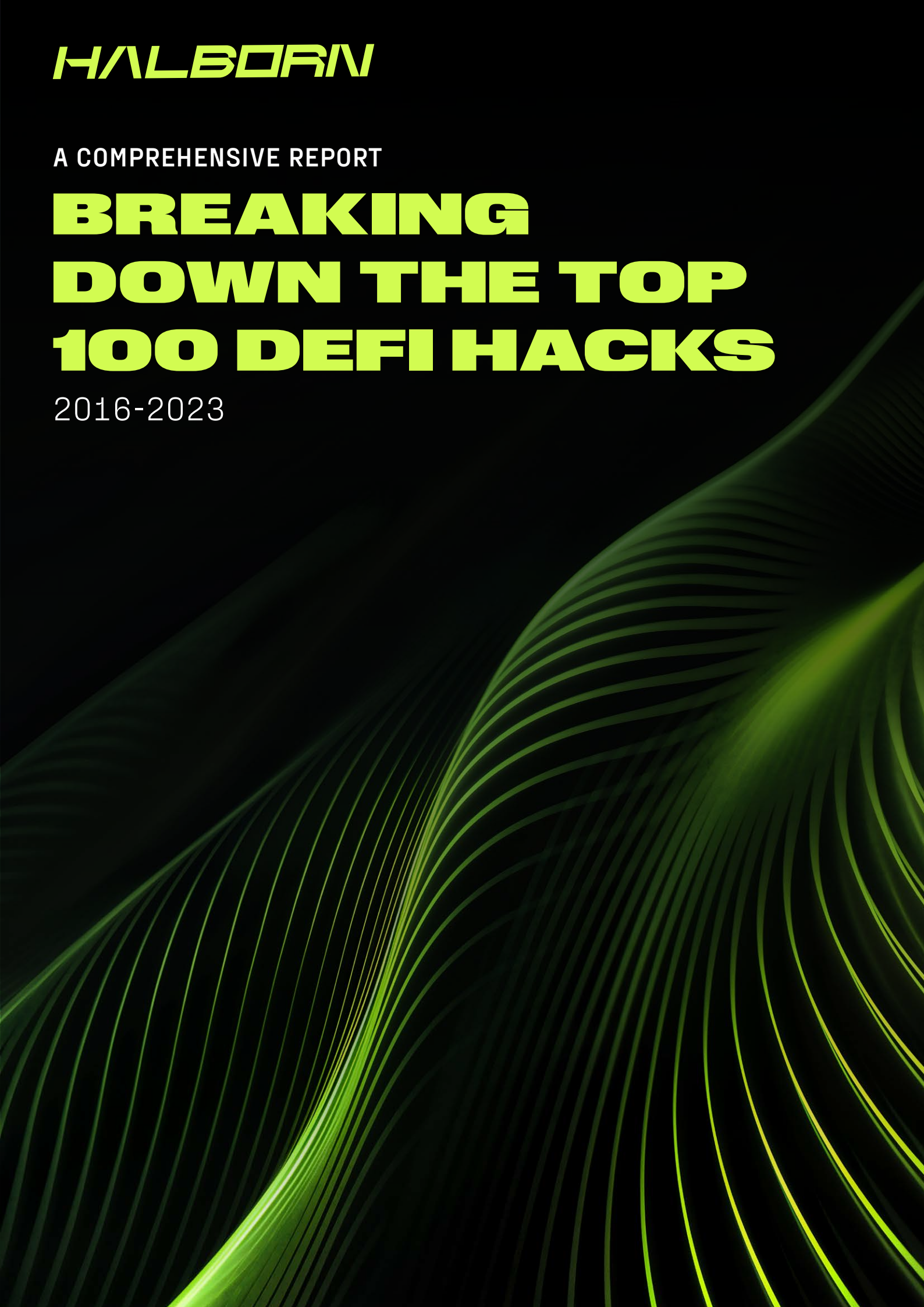


HALBORN

A COMPREHENSIVE REPORT

**BREAKING
DOWN THE TOP
100 DEFI HACKS**

2016-2023

The background of the entire page is black. In the lower half, there are several layers of thin, wavy, green lines that create a sense of motion and depth, resembling a stylized wave or a digital signal. The lines are more densely packed and brighter in the lower right corner, fading towards the top and left.

CONTENTS

Introduction	3
Key findings	4
Time Distribution and Amount Lost	6
Distribution of attacks per chain	9
Type of DeFi attacks	14
Type of attacks	18
Subcategories	18
Signature scheme and wallet protection	20
Use of flash loans	24
Root cause analysis	29
Direct contract exploitation	33
Price manipulation attacks	36
Governance attacks	39
Rug pull/scams	39
Compromised private key	42
Phishing	46
Traditional	46
Attacks per chains	47
Type of protocols	51
Governance	59
Type of Protocols per Chains	61
Type of protocols per type of attacks	65
Type functions	69
Type of Functions vs Chains	72
Type of Functions vs Types of Attacks	76
Type of Functions vs Protocols	79
Were they audited?	83
Audited Protocols by Chain	85
Audited Protocols by Type of Attack	89
Audited Protocols by Type of Protocol	93
Audited Protocols by Type of Function	97
Actionable takeaways	101

Author

Mar Gimenez-Aguilar, Lead Security Architect and Researcher.

INTRODUCTION

DeFi hacks are more common each day, causing losses of millions of dollars.

It is estimated that protocols lost \$3.9 billion in 2022¹. In 2023, funds stolen in DeFi platforms decreased by more than 50%; however, they are still considered a major threat to the industry.²

Halborn intends to provide in this report a comprehensive review of the top 100 hacks by loss in DeFi history, building on our previous report of the top 50 hacks until 2022³. As in the preceding release, we will analyze the time distribution, chain, cause, type of protocol and function (if applicable), and whether the protocol was previously audited. Remediation and advice to avoid future losses will also be presented.

¹ <https://cryptonews.com/news/web3-lost-nearly-4-billion-to-fraudsters-last-year-will-things-improve-in-2023.htm>

² <https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2024>

³ <https://www.halborn.com/reports/top-50-defi-hacks>

KEY FINDINGS

Some key findings from this study include:

- **DeFi hacks are still something to worry about, but they seem to have been less severe in 2023.** The total amount lost because of the top 100 largest DeFi hacks accumulates to \$7,352,064,089 USD. Although in previous years there seemed to be a tendency toward an increase in the number of hacks and their severity, for our sample in 2023, there were 6% fewer attacks than in 2022. Furthermore, the average value lost per attack is \$47 million USD less than in the previous year. It should be taken into account, however, that the total value locked (TVL), decreased since the middle of 2022 and continued this way through 2023, so this could also influence the attacks and value stolen.
- **Ethereum, BSC, and Polygon in the spotlight.** On the one hand, Ethereum and Binance Smart Chain are in the top 3 of chains by TVL and number of protocols, and they get the first and second spots both by number of attacks and value lost. On the other hand, Polygon occupies the third place by number of attacks and fourth by value lost. Furthermore, Polygon experienced more hacks than it should have based on its TVL.
- **Off-chain attacks seem to be an increasing threat, especially compromised private keys.** Although on-chain attacks like **smart contract exploitation**, **price manipulation**, or **governance** attacks are still the majority, off-chain attacks represent 29% of the total number of attacks and 34.6% of the funds stolen in general. Specifically, in 2023, off-chain attacks made up 56.5% of total attacks and accounted for 57.5% of the stolen amount for that year. Furthermore, **compromised private keys** are the second most common cause of attacks and loss (after **direct smart contract exploitation**), representing 27% of the attacks and 32.6% of the value lost. In 2023, 52.2% of the total attacks were due to a **compromised private key**, and they represent 55.7% of the losses for this year.
- **Audits of code and the whole ecosystem are necessary.** The majority of protocols attacked had contracts that were not audited. Furthermore, while audited protocols represent 20% of the sample, they only represent 14.3% of the value lost. However, some vulnerabilities leading to attacks, like **price manipulation**, are hard to find in audits if the whole ecosystem and how the protocol interacts with it are not considered. How off-chain elements interact with the protocol should be also taken into account.
- **Lack of use of multi-signature or MPC and cold wallets.** Only 21.1% of the attacked protocols used multi-sig or MPC schemes or wallets. However, the user needs to be careful when storing the private key, because, if a multi-sig or MPC wallet is attacked successfully, it can produce major losses. Therefore, it is also recommended to use cold wallets to increase the level of security, which, according to our data, 5.3% of the protocols used but only represent 2% of losses.
- **A lack of or faulty input verification or validation is the main cause of hacks and loss in direct exploitation of contracts.** Furthermore, it is also the main vulnerability by number and loss in general.

- **Reentrancy attacks, still relevant after all these years.** Reentrancy attacks have been present in almost all years, and the number has increased in 2023, with them causing 16.7% of the attacks that year.
- **Beware of faulty proof verification vulnerabilities.** Often associated with bridges (they caused 50.6% of the amount lost for this type of protocol), we have observed that the exploitation of this kind of vulnerability leads to major losses, as they represent only 4.3% of attacks but account for 25.7% of the total losses. This is especially relevant for **direct contract exploitation** of a protocol's smart contract, where this kind of vulnerability seems to account for 30.4% of the total losses.
- **Logic errors are the main cause of hacks and losses for direct contract exploitation in 2023.** In 2023, most of the hacks were enabled by logic errors (66.7%) and they caused 74.8% of the lost amount. Therefore, be especially cautious when designing a function and keep in mind all the requirements and how it would interact with the ecosystem in order to not incur these types of vulnerabilities.
- **Be careful with oracles.** A **flawed oracle** is the main reason why attackers are able to execute **price manipulation** attacks. It is also the main cause of losses in them. In 2023 alone, they were responsible for 49.1% of the losses caused by this type of attack.
- **Flash loans can be a means of attack.** Although the majority of the hacks studied do not use them, they are especially relevant in **price manipulation** and **governance** attacks. In fact, they are responsible for 62.1% of the funds lost in **price manipulation** attacks. Furthermore, there has been an increase in their use in 2023 compared to 2022, where flash loans were used in 62.5% of attacks versus 26.3% of the previous year. Therefore, consider flash loan attacks a possibility if your protocol allows swapping and exchange of assets or uses token quorum power in governance processes.
- **Lending protocols, Bridges and CEXs are still the most insecure types of protocols. Lending** protocols are the most attacked type of protocols, while **Bridges** accumulate the highest loss, followed by **CEXs**. Furthermore, the latter two types of protocols accumulate a very high number of attacks in proportion to the total of protocols available. This trend seems to be continuous over time, with CEXs being the most attacked protocol and bridges the second major cause of losses in 2023, while lending protocols were the first cause of losses that year.
- **Decentralized protocols seem to have fewer losses.** Although decentralized protocols represent 44% of the protocols hacked, they only account for 28.2% of the funds stolen.
- **Functions used to “withdraw”, “deposit”, “transferOwnership” or “verifyProof” should be carefully reviewed and secured.** While **withdraw**-like functions and **deposit** have been the most targeted, functions to verify a proof and transfer ownership of a contract are the ones that caused the biggest monetary losses.

TIME DISTRIBUTION AND AMOUNT LOST

DeFi protocols' use and development have increased through the years.

However, the biggest losses are not distributed in a constantly increasing way. **Figures 1 and 2** show the number of hacks per year, as a percentage of total hacks and in total. The earliest one corresponds to the DAO hack in June 2016. It can be seen that the number of hacks or other kinds of attacks is bigger in 2021 than in 2022 and in 2022 than in 2023.

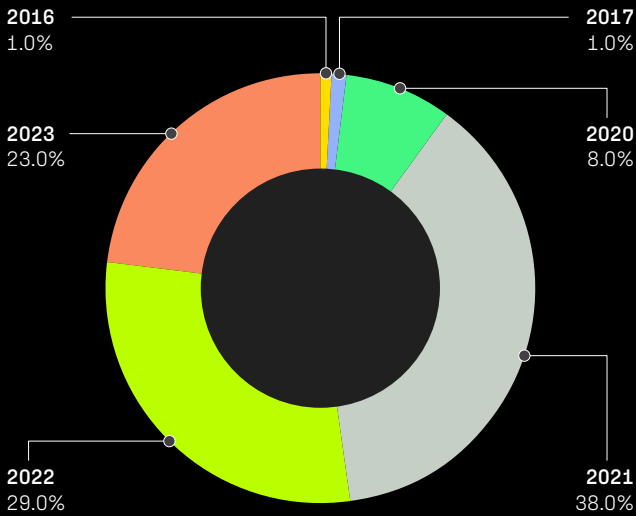


Figure 1: Number of hacks per year [percentage]

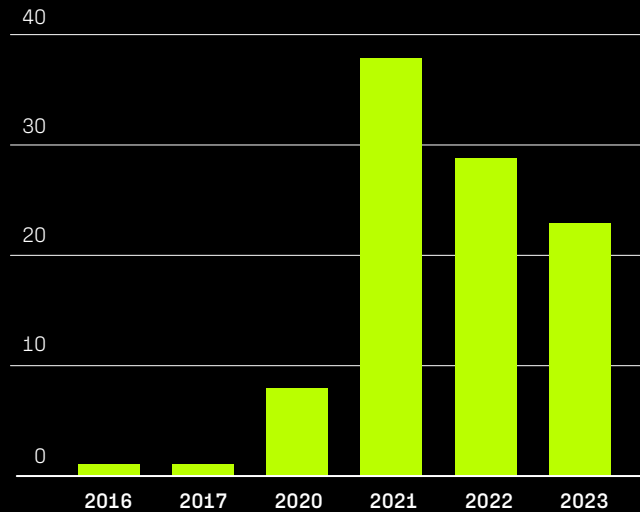


Figure 2: Number of hacks per year [count]

The total amount lost because of these attacks is approximately \$7,352,064,089 USD.

If we observe the distribution of this loss by year, there was an increase in funds stolen each year until 2022. In 2023, however, the amount lost decreased below that of 2021, decreasing from 44% of the share of total value lost (\$3,234,140,000 USD) to only 19.9% (\$1,464,537,951 USD). We can also observe this information in Figures 3 and 4.

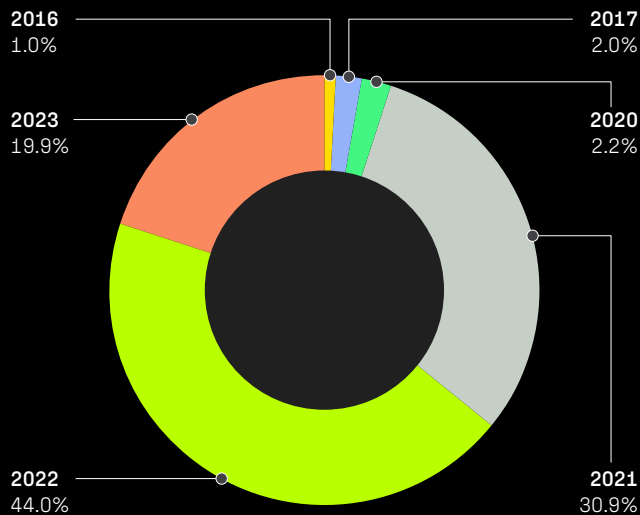


Figure 3: Loss caused by hacks per year [percentage]

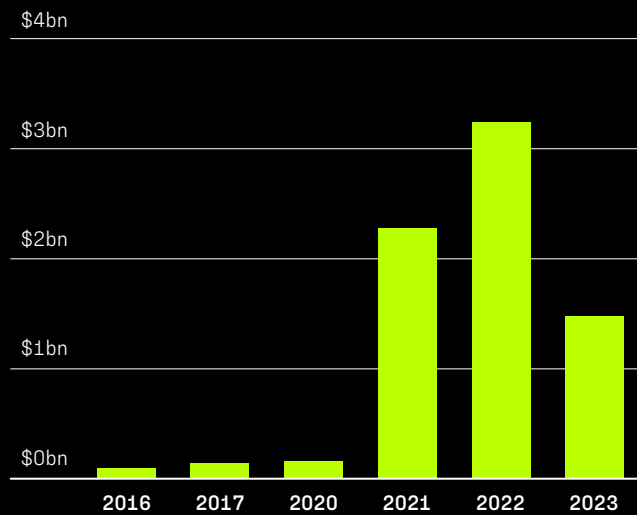


Figure 4: Loss caused by hacks per year [USD]

These could be explained by a few different causes:

- **There are fewer attacks in 2023 than in 2022.** Out of the top 100 attacks, 23 happened in 2023, while 29 in 2022 (see Figure 1).
- **The severity/impact of the attacks from 2023 is less than those in 2022.** There are fewer attacks in 2022 than in 2021 (see Figures 1 and 2); however, the amount lost is higher in the later year. As stated in the previous point, the number of attacks is again reduced in 2023. In this case, the loss is also reduced to almost half of the previous year's. This could indicate that the hacks in 2023 are less severe in the amount lost than those for 2022. Figure 5 shows that, indeed, the average loss per attack in 2023 is smaller in 2023 than in 2022 by 47, 846, 505 USD.
- **Total value locked (TVL), decreased since the middle of 2022 and continued this way through 2023** (Figure 6¹). Because of this, even if the number and severity of the hacks were the same, there is less money available to steal, so the amount lost by protocols will be less than in previous years.

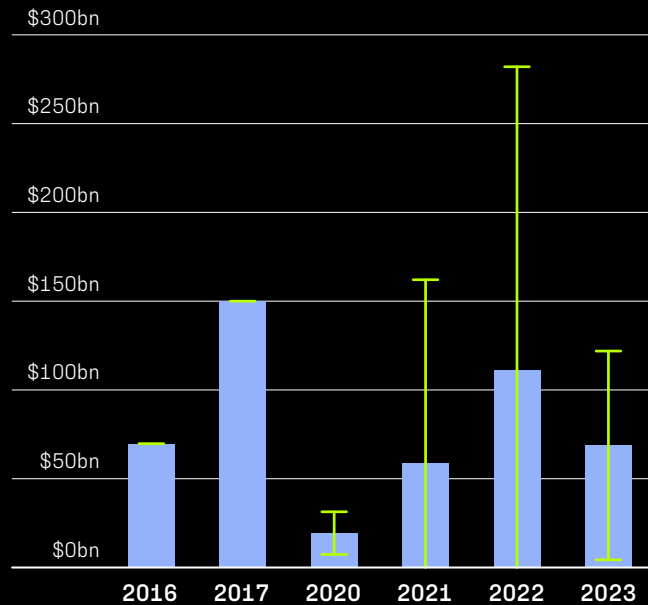


Figure 5: Average and standard deviation of the loss by year [USD]

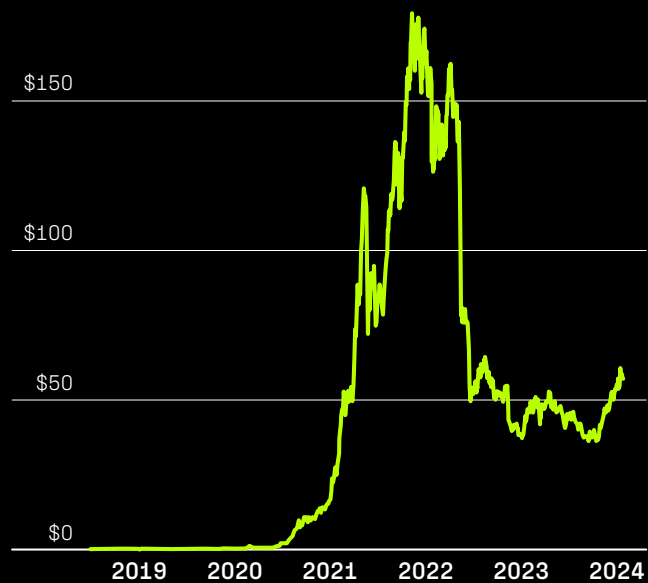


Figure 6: TVL in DeFi by year

¹ <https://defillama.com>

DISTRIBUTION OF ATTACKS PER CHAIN

Figures 7 and 8 show the distribution of the different hacks per chain.

If an attack has been carried out in more than one chain, it is counted in each one. It is noticeable that almost 45.7% are on the Ethereum network. Indeed, this chain has been identified as the biggest by TVL and the one with the biggest number of DeFi protocols. It is followed by Binance Smart Chain (BSC), which accounts for around 16.3% of the total hacks. However, that chain is the third by TVL, although it is the second by number of protocols (source: <https://defillama.com/chains>). The third by number of attacks is Polygon, which is the seventh by TVL and the fourth by number of protocols at the time of writing. Those attacks labeled as Multi, have affected more than 6 different chains.

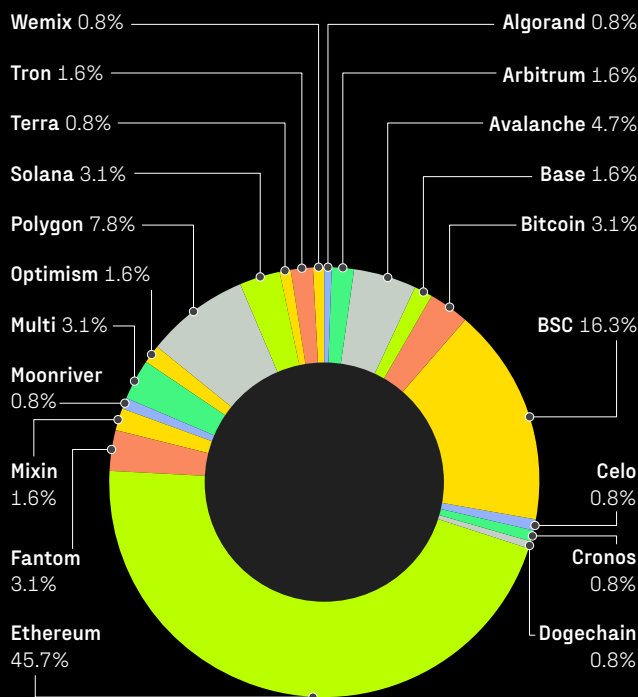


Figure 7: Number of attacks per chain [percentage]

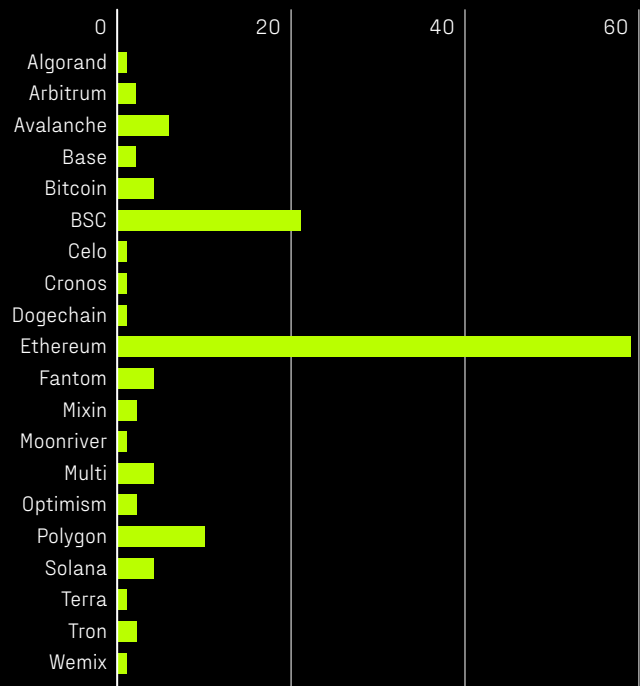


Figure 8: Number of attacks per chain [count]

It is also interesting to examine the monetary losses that these attacks have caused on each chain.

If we observe Figure 9 and Figure 10, they show the distribution of stolen funds by chain. Ethereum accounts for 54.2% of the funds stolen (\$3,984,645,462.00 USD), which makes sense because, as we have established before, it is the chain with the highest TVL and number of protocols. The second chain by loss is BSC, with 18.5% (\$1,359,649,033.00 USD). The third one is Solana, adding up to 6.8% of the total value lost, approximately \$497,800,000.00 USD. All the percentages for these chains are higher than their occurrence ones, which suggests that, in proportion, attacks on them cause more damage than the others.

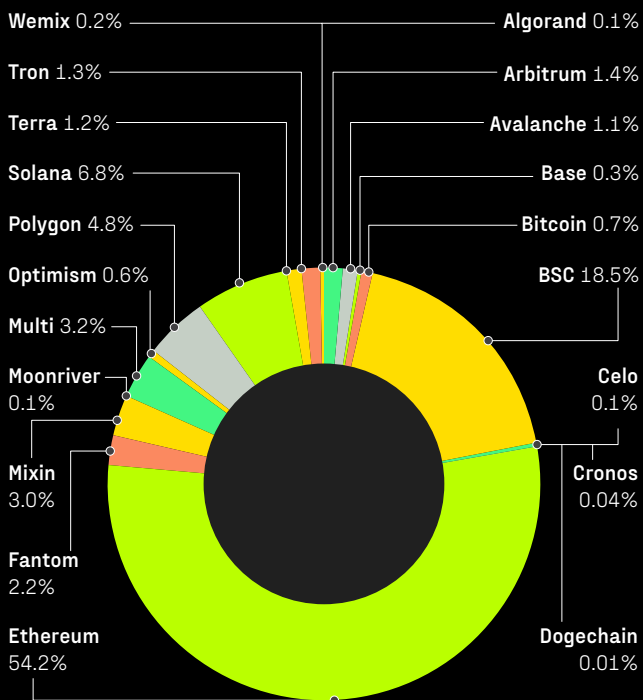


Figure 9: Loss caused by the attacks per chain [percentage]

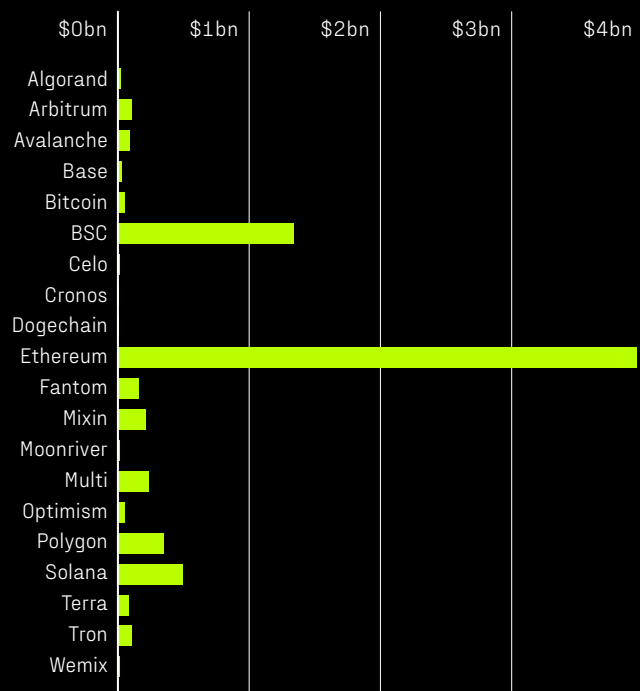


Figure 10: Loss caused by the attacks per chain [USD]

In Figure 11, we can see a comparison of how the attacks per chain and TVL in those chains are ordered (ranked from largest to smallest).

We want to examine if the value locked in each chain influences the number of hacks. For those that have an equal number of hacks, we followed the order by TVL. For a better understanding of last year's hacks, we also included the ranking for only 2023 (missing chains have not been attacked in this year). A value appearing under the blue line means that it is higher in the ranking by number of attacks than it is by TVL.

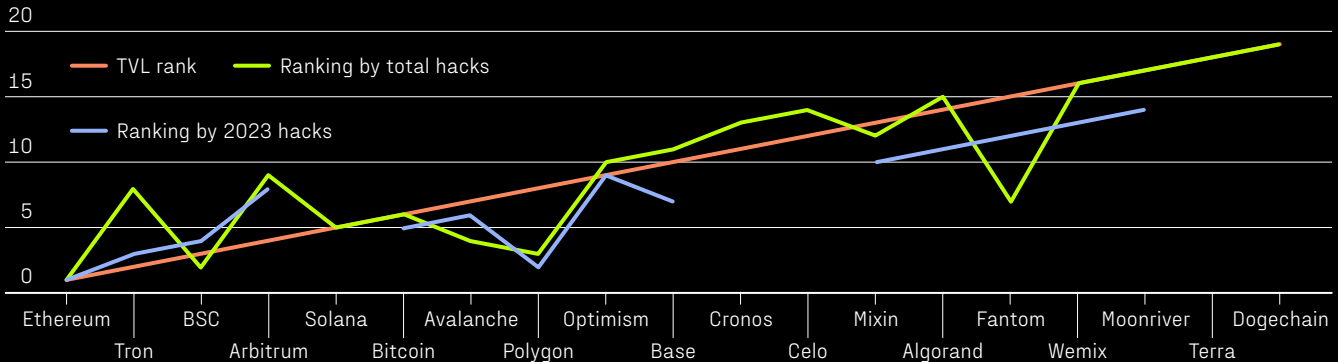


Figure 11: Order by TVL and number of attacks in total and only in 2023

We can observe that, in general, there seems to be some kind of relation between more attacked chains and their TVL. However, there are some cases we want to highlight. First, Tron and Arbitrum, despite having a high TVL, seem to not be the target of relatively few attacks in comparison. Secondly, there is a significant disparity between the ranking by Total Value Locked (TVL) and the frequency of attacks on the Fantom and Polygon chains: they appear to be targeted much more frequently than their TVL would suggest. In 2023, however, the difference in the Fantom chain is not as noticeable. Polygon is still a worrying case, and Base seems to be beginning to follow this trend.

11

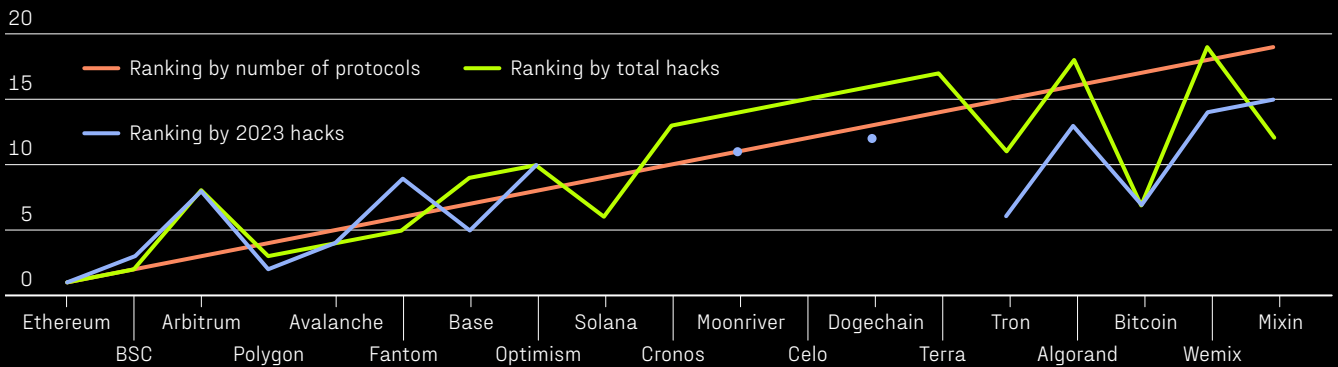


Figure 12: Order by number of protocols in chain and number of attacks in total and only in 2023

Another metric to study would be how much the chains are attacked in comparison with their number of protocols. It would make sense that, the higher this value is, the more targets a hacker has and, therefore, the more opportunity to steal funds. In Figure 12, we can see these values. As before, for those with an equal number of hacks, we ranked by the number of protocols. In this case, the relationship between these values is less close than for the previous chart (Figure 11), but it still seems to be somehow related. It is noticeable that, while Arbitrum, BSC, Base, Cronos, Moonriver, Celo, Dogechain, Terra, and Algorand seem to be somewhat less targeted, Polygon, Solana, Tron, Bitcoin, Wemix, and Mixin show the contrary. If we consider only attacks in 2023, Tron and Bitcoin have much more attacks than what would correspond to their number of protocols.

Figures 13 and 14 show the distribution of hacks by chain per year.

It should be noted that, for example, BSC and Solana were introduced in 2020, so some chains would not be available in earlier years. The trend seems to indicate that while Ethereum is still the one that accumulates the most attacks (probably because is also biggest by TVL and number of protocols), as new chains appeared and begin being used, the attacks seem to distribute among them, reducing the difference between Ethereum and the rest of the chains. New additions to the attacked chains in 2023 include Algorand or Moonriver chains, the former one released in 2017 and the latter in 2021.

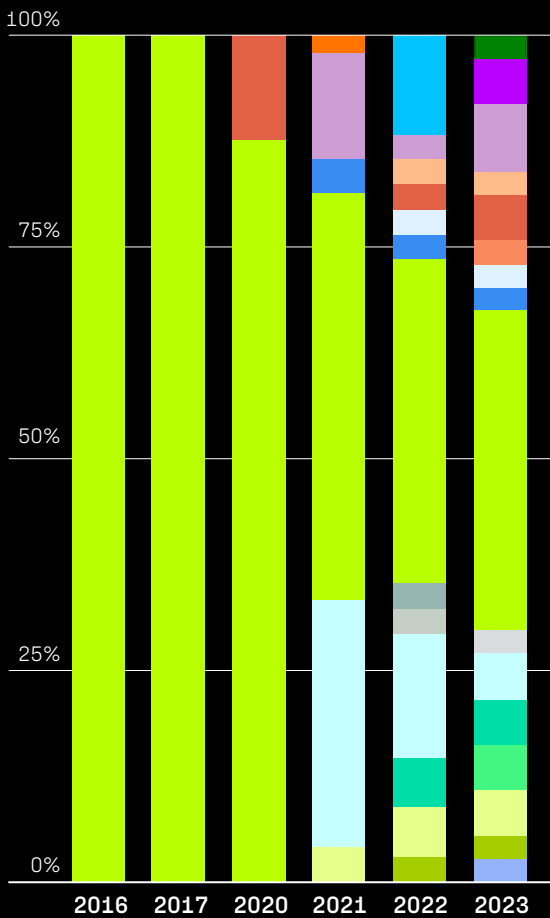


Figure 13: Number of attacks per year and chain [percentage]

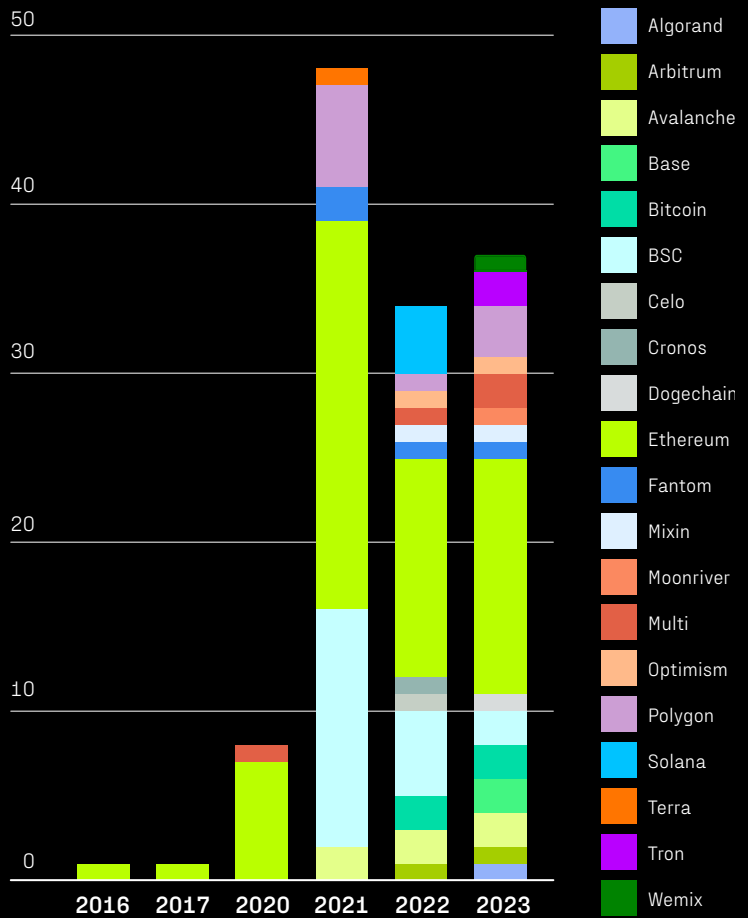


Figure 14: Number of attacks per year and chain [count]

The distribution of the lost value per chain over the years follows a similar pattern.

The proportion of the funds stolen in Ethereum decreases each year, giving space to other chains. In 2023, the Mixin chain stands out, as it accounts for 13.7% of the total value stolen for this year (around \$200,000,000.00 USD), while only being the target of a single attack (2.7%). (See Figures 15 and 16)

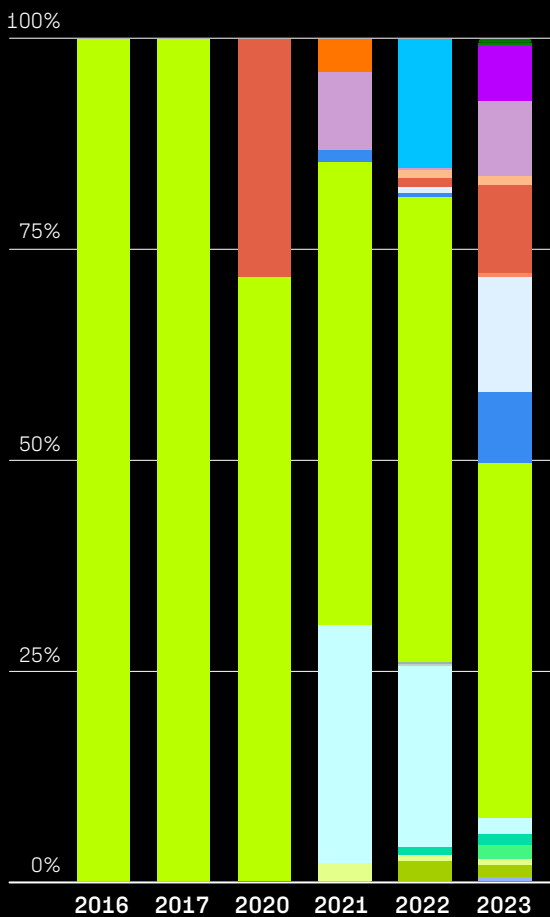


Figure 15: Loss caused by attacks per year and chain [percentage]

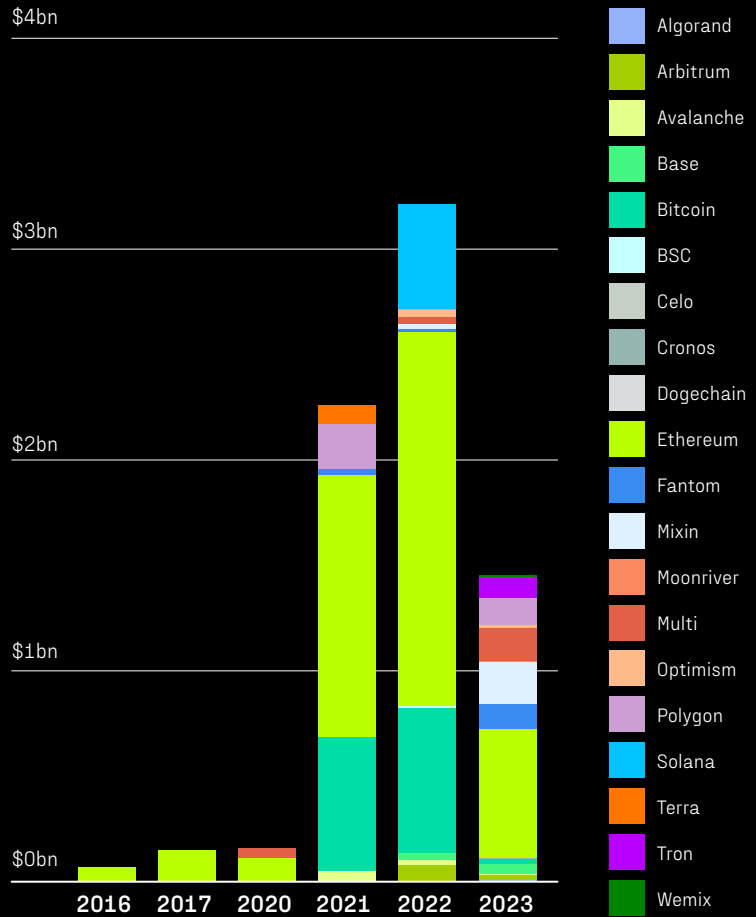


Figure 16: Loss caused by attacks per year and chain [USD]

TYPE OF DEFI ATTACKS

This section considers two main categories of attacks:

Off-chain attacks

Those hacks in which the main attack vector does not happen on chain, although results can be visible on it. For example, **compromised private keys** or **traditional** attacks.

On-chain attacks

Those attacks in which the main attack vector happens on the blockchain, for example the **exploitation of a contract** or a **price manipulation** attack. **Rug pulls and scams** are included in this category, as they need of a backdoor in the contract or by the protocol to hold accounts with privileges or access to the protocol's funds (enabled by the design of the protocol's on chain components).

These categories will have different sub-categories, namely:

Off-chain attacks

- **Compromised private keys:** This attack occurs when a private key is stolen or leaked, commonly via phishing attacks or by compromising the system in which the private key is stored. When an attacker is able to obtain the private key of an account by other methods (e.g. brute force on a weak cryptographic algorithm) it is also included in this category.
- **Traditional:** When the attack is carried out by other means not related with on-chain components, like gaining access to an API key with privileges capabilities on the protocol.
- **Phishing:** Phishing often occurs when an attacker tricks users into signing permissions (usually done by supplanting a legitimate protocol), allowing the attacker to spend tokens on users' behalf. It can also be called ice phishing when it is DeFi specific.

On-chain attacks

- **Price manipulation attack:** When an attacker artificially manipulates the price of a digital token, often by taking advantage of low or empty liquidity pools, exploiting a contract vulnerability that allows them to manipulate that price, or because of flawed oracles that do not reflect accurate asset prices. **Price manipulation** often requires a series of operations, like swaps or liquidity supply and removal, and involves two or more assets and a pool.
- **Direct contract exploitation:** When an attacker exploits a vulnerability in smart contract code, which typically grants access to various mechanisms of a protocol and unlawful receipt of funds or tokens. It can have more than one token involved and operations, but there is no pool and/or price manipulation.
- **Governance attack:** When a perpetrator exploits a blockchain venture governed through decentralized means by acquiring sufficient influence or voting power to execute a malicious proposal.
- **Rug pull/Deceptive Practice (Scams):** It refers to a situation where developers or project creators abruptly withdraw liquidity or funds from a DeFi protocol, leaving investors or users with significant losses. This typically occurs after users have invested funds or assets into the project, often lured by promises of high returns or other incentives. Projects in which the CEO has been arrested and accused of withdrawal of funds are also included in this category. Because this is often possible because of intentional backdoor functions left in the contract or by the existence of permissioned accounts or centralized accounts holding the project funds on chain (by how the protocol is designed on the blockchain), these attacks are included under the on-chain group.

For the main categories, we can observe (Figures 17 and 18) that the majority of attacks are on-chain ones (71% versus 29%).

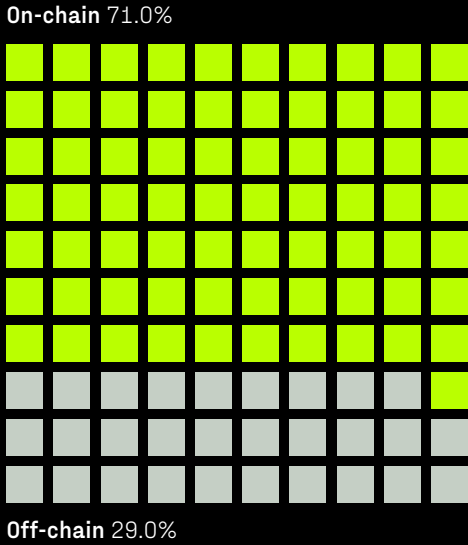


Figure 17: Number of off-chain and on-chain attacks [percentage]

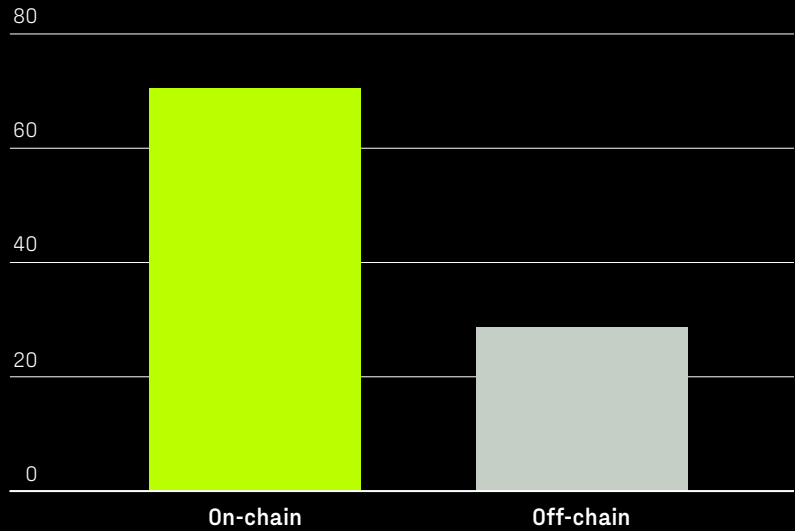


Figure 18: Number of off-chain and on-chain attacks [count]

However, if we observe the loss produced by each type of attack (Figures 19 and 20), we can see that the difference is slightly smaller, with off-chain attacks creating about 34.6% of losses despite accounting for 29% of attacks. While losses due to off-chain attacks accumulate to around \$2,546,025,089.00 USD, on-chain ones produced around \$4,806,039,000.00 USD.

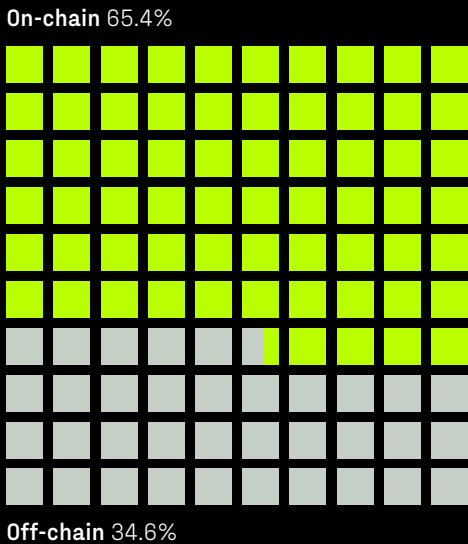


Figure 19: Loss caused by off-chain and on-chain attacks [percentage]

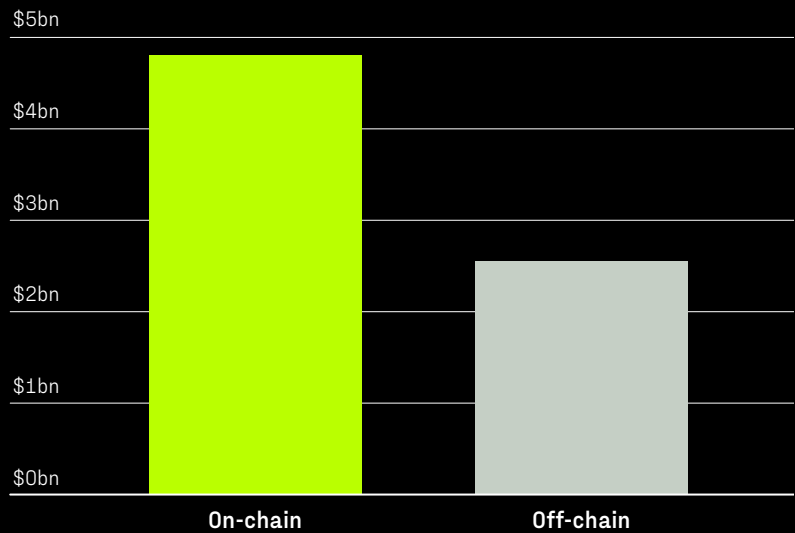


Figure 20: Loss caused by off-chain and on-chain attacks [USD]

By year, we can observe that the number of hacks due to off-chain elements have increased, causing around 56.5% of the hacks in 2023 (Figures 21 and 22).

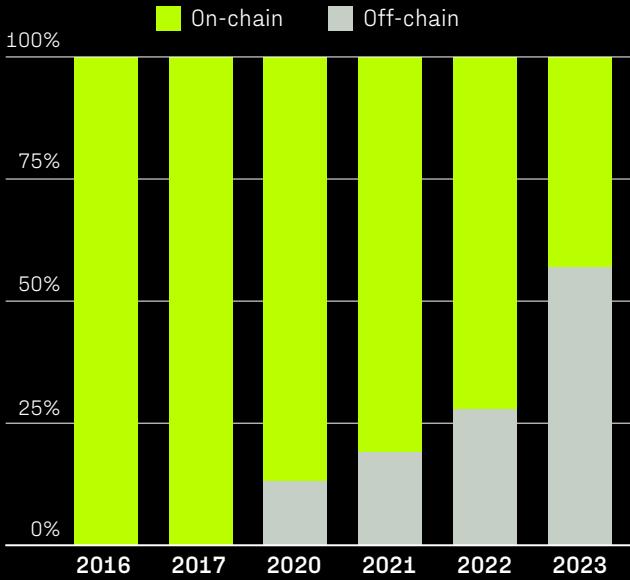


Figure 21: Number of off-chain and on-chain attacks per year [percentage]

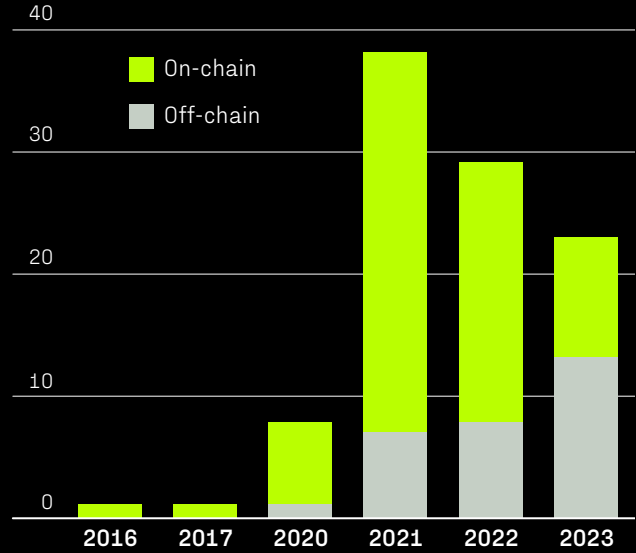


Figure 22: Number of off-chain and on-chain attacks per year [count]

By loss, we can see the same tendency in Figures 22 and 23. The percentage of funds lost by off-chain attacks vectors increases each year. The percentage in 2023 is slightly higher in this case, reaching around 57.5% of total losses (around \$842,448,951 USD).

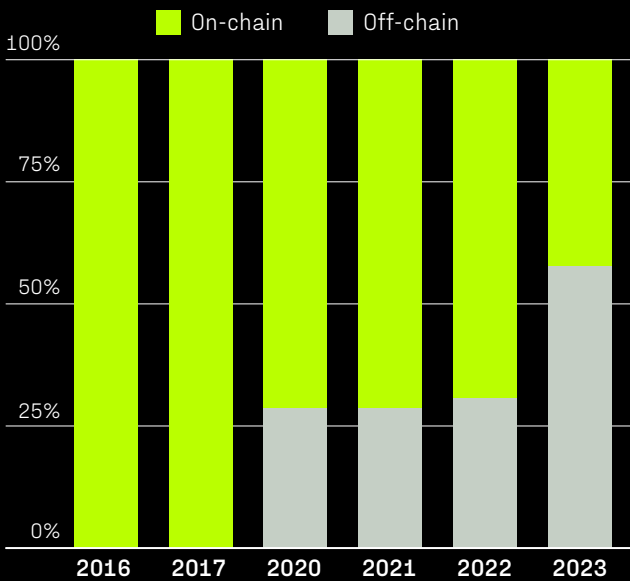


Figure 23: Loss caused by off-chain and on-chain attacks per year [percentage]

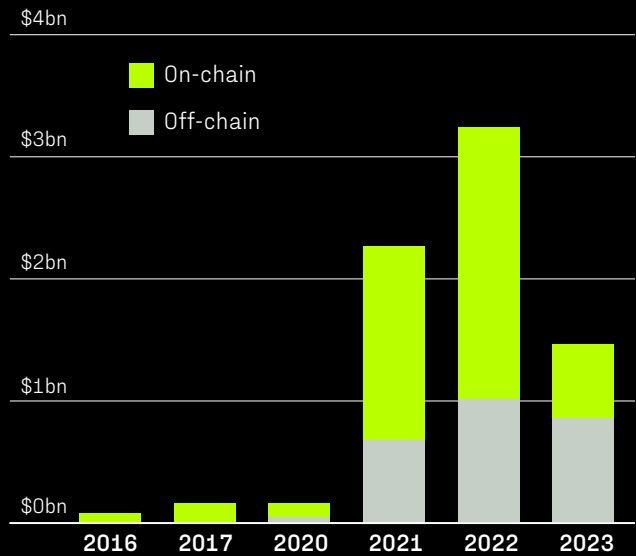


Figure 24: Loss caused by off-chain and on-chain attacks per year [USD]

Type of attacks

Subcategories

As stated before, these two main categories of attacks have been divided into different sub-categories.

Figures 25 and 26 show how these categories are divided by occurrence. We can see that the most common cause of hacks is **exploiting the smart contract** (33%), followed by those that were possible because of a **compromised private key** (27%) and **price manipulation** attacks (27%).

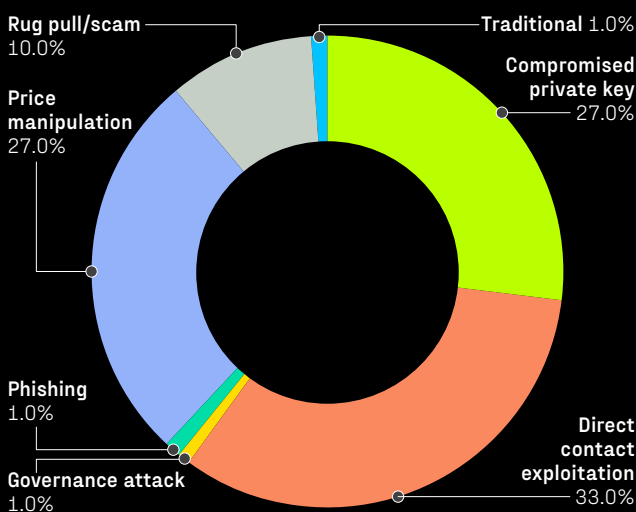


Figure 25: Number of attack sub-categories [percentage]

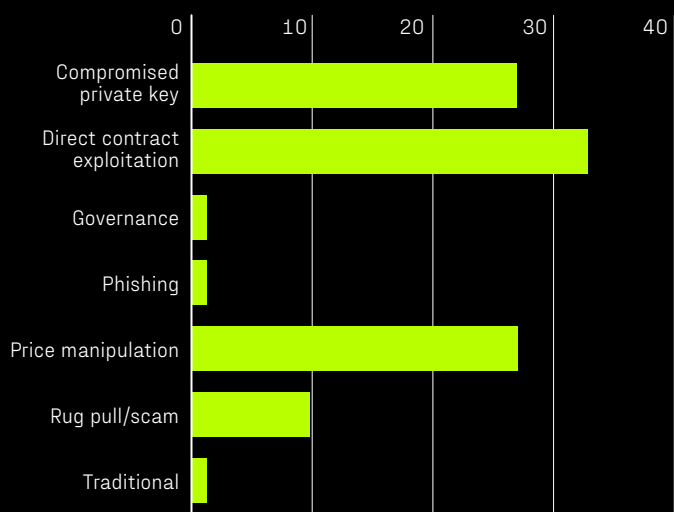


Figure 26: Number of attack sub-categories [count]

By loss, we can see that the differences with regard to which type of attacks are most destructive increases (Figures 27 and 28). **Direct smart contract exploitation** sums up to 40.7% (\$2,991,850,000 USD). **Compromised private keys** are the cause of 32.6% of the total value lost (\$2,400,025,089 USD) while **price manipulation** attacks, albeit keeping third place, only are responsible for 11.5% of the losses (\$846,623,000 USD).

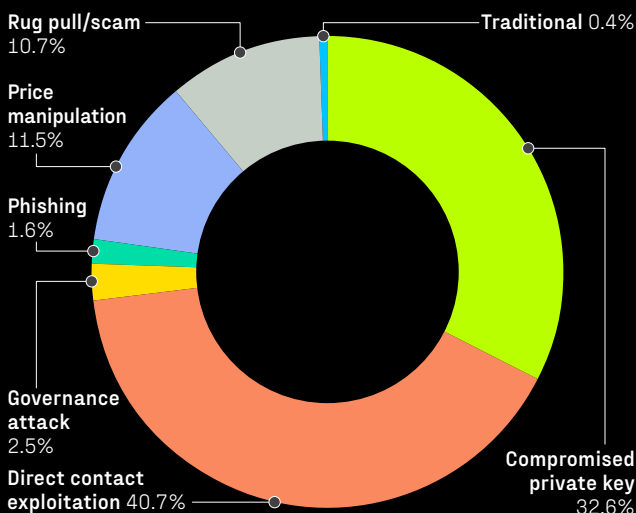


Figure 27: Loss caused by attack sub-categories [percentage]

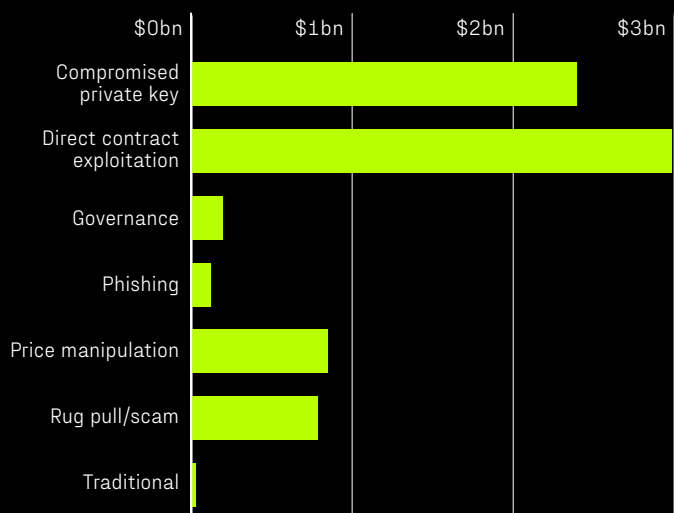


Figure 28: Loss caused by attack sub-categories [USD]

If we study the distribution of the number of hacks per type over the years (Figure 29 and 30), we can observe that, while at the beginning most of them were **smart contract exploitation**, from 2020 onwards, the types of attacks diversified.

Price manipulation attacks are the second most common type in 2020 (25% of them) and increase slightly in 2021 before decreasing through 2023, ending with around 21.7% of the total. However, **compromised private key** attacks have been increasing in popularity through the years, starting with 12.5% of the total in 2020 and growing to 52.2% in 2023. It should be noted that **rug pull and scams** are still a notable cause of attacks, maintaining around 10% of attacks throughout the years.

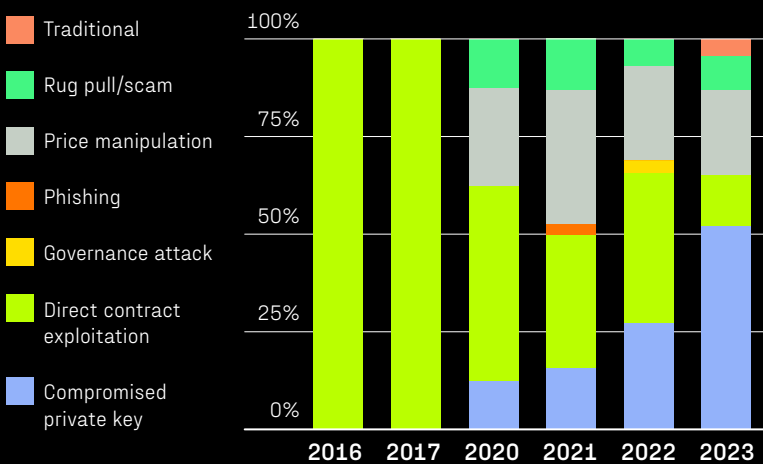


Figure 29: Number of attack sub-categories per year [percentage]

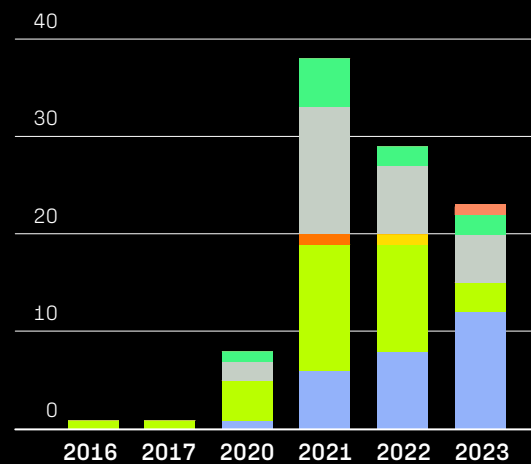


Figure 30: Number of attack sub-categories per year [count]

If we check how many funds were lost by type each year, we can observe Figures 31 and 32. In this case, **compromised private keys** show a concerning result, accounting for more than half (55.7%, \$816,448,951 USD) of the losses for 2023. **Direct contract exploitation** seems to be also increasing in severity, growing from 39% of the losses (\$62,100,000 USD) and 50% of incidents in 2020 to 18.8% of the losses (\$274,800,000.00 USD) in 2023 (with only 13% of the total attacks). **Price manipulation** attacks seem, however, to reduce in impact over the years, causing only 13.4% (\$196,823,000 USD) of the losses in 2023 (versus 21.7% of the number of attacks). **Rug pulls and scams**, in general, seem to be less dangerous in terms of loss versus occurrence.

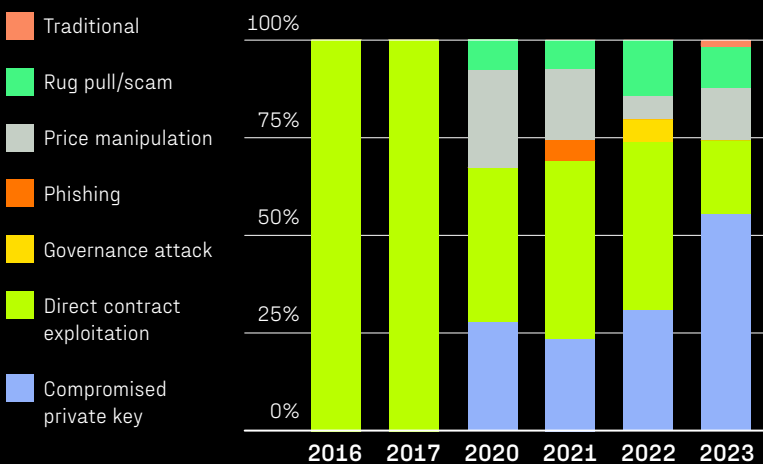


Figure 31: Loss caused by attack sub-categories per year [percentage]

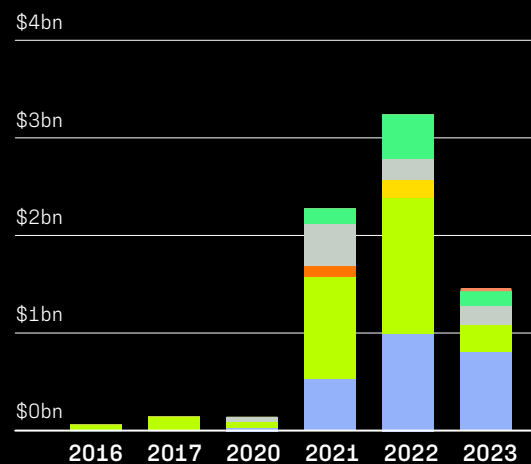


Figure 32: Loss caused by attack sub-categories per year [USD]

Signature scheme and wallet protection

In the previous section, it has been presented that the second most common cause an attack is a compromise of a project's private keys, which allows attackers to exploit the protocol.

In order to provide a better understanding of how these keys could have been exploited, two things will be analyzed. First, if the vulnerable key was part of a multi-signature or a multi-computation wallet or scheme. This entails that, in order to execute a transaction, it needs to have two or more signatures or all the parts of the signature. Multi-sig provides more security than single-signature transactions because more keys need to be compromised in order to damage the protocol. Likewise, in multi-computation wallets or schemes, the key is divided into different encrypted shares to be distributed among parties. According to our research, only 21.1% of attacks in which the private key has been compromised have used either a multi-signature or multi-computation wallet or scheme (Figures 33 and 34).

Even though 78.9% of the protocols attacked were not using multi-signature or MPC wallets or scheme, suggesting a higher risk, these measures do not fully shield against the financial damage caused by such attacks. In fact, attacks that breached protocols with these security features still resulted in significant losses, amounting to 43.7% of the total, or \$813,436,138 USD (refer to Figures 35 and 36). This data implies that while hacking into these types of wallets or schemes might be more challenging, if there is a higher amount of funds or higher level of management privileges the wallet or scheme has access to, successful breaches can lead to substantial damage.

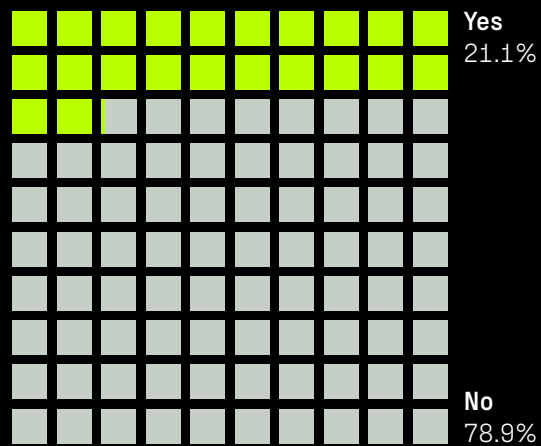


Figure 33: Number wallets/schemes using multi-signature or MPC [percentage]



Figure 34: Number wallets/Schemes using multi-signature or MPC [count]

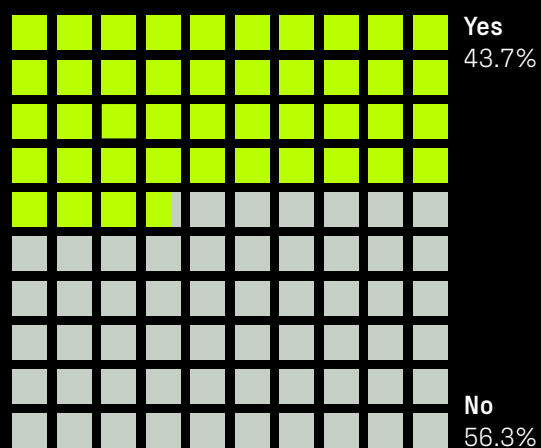


Figure 35: Loss caused by wallets using multi-signature or MPC [percentage]



Figure 36: Loss caused by wallets using multi-signature or MPC [USD]

Through the years, it seems that this kind of wallet or scheme was used by 16.7% of the protocols whose keys were compromised in 2021.

This number increased to 28.6% in 2022 but decreased to 20% in 2023, as can be observed in Figures 37 and 38.

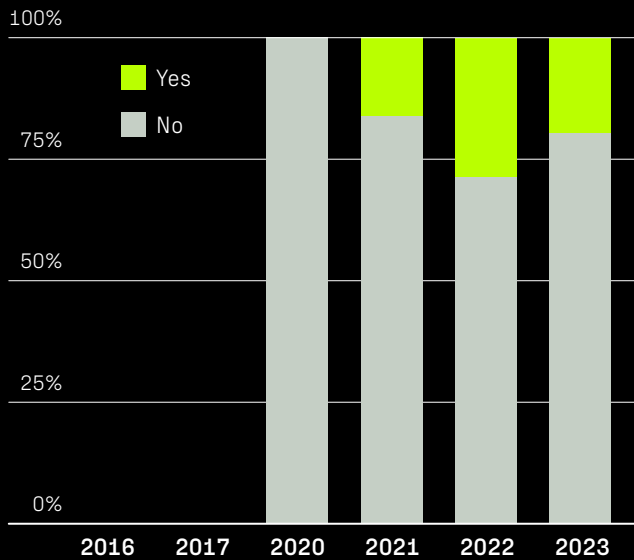


Figure 37: Number wallets using multi-signature or MPC per year [percentage]

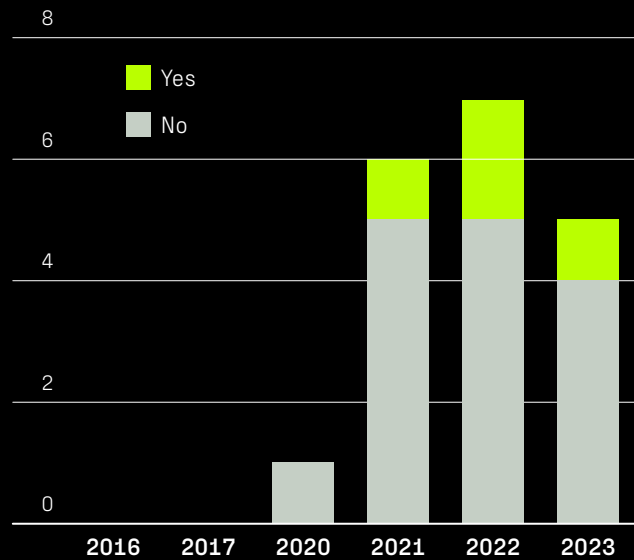


Figure 38: Number wallets using multi-signature or MPC per year [count]

If we check the evolution of losses (Figures 39 and 40) , we can see that there is not a clear pattern, as there is almost no loss in 2021 for multi-sig or MPC wallets or schemes (only 1.5%, \$7,936,138 USD) that increases into a lot of funds lost in 2022 (72.9%, \$724,000,000 USD) to decrease again in 2023 (28.2%, \$81,500,000 USD). The huge spike in 2022 is caused by the Ronin Bridge hack, in which their multi-signature was hacked, producing a loss of \$624,000,000 USD.

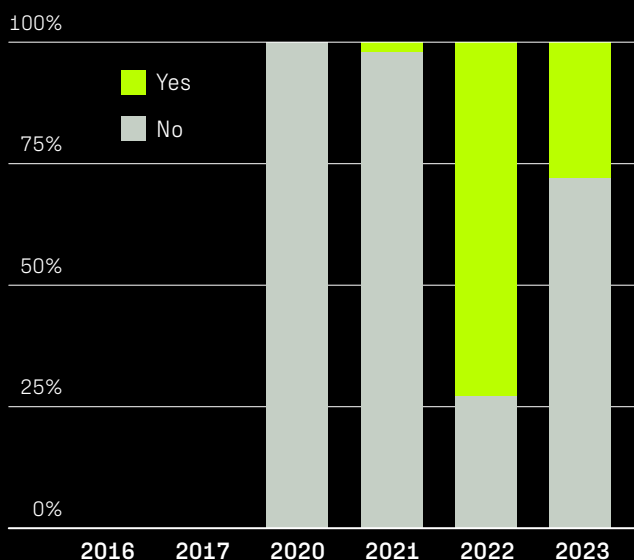


Figure 39: Loss caused by wallets using multi-signature or MPC per year [percentage]

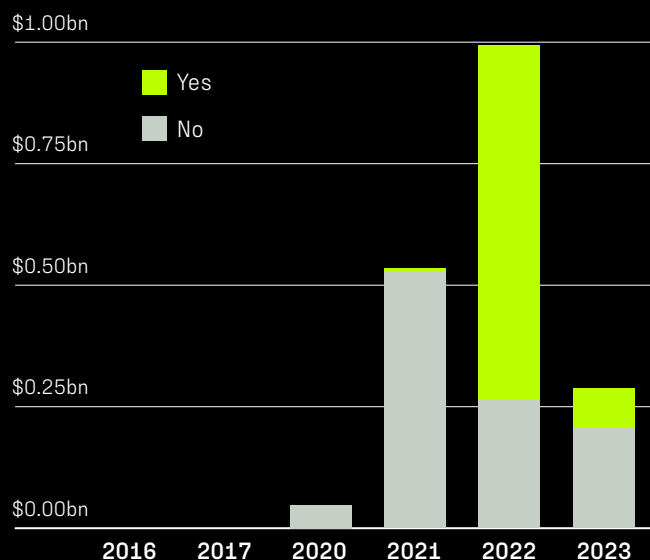


Figure 40: Loss caused by wallets using multi-signature or MPC per year [USD]

Another security measure is to use cold wallets or storage instead of hot wallets.

Hot wallets are connected to the Internet, while cold wallets utilize private keys kept offline. A benefit of the first type of wallets is ease of use; however, they are less secure than cold wallets. This is because, in order to steal from a cold wallet, the attacker would usually require physical access to it, as well as any associated password to unlock access to the funds.

In the analyzed sample (Figures 41 and 42), the vast majority of the private keys compromised were stored in a hot wallet (94.7%), and only one was actually in a cold wallet.

Comparing the loss produced by each type of wallet, Figure 43 and Figure 44 show that 98% of the total value lost (\$1,825,756,089 USD) corresponds to hot wallets. The percentage of value lost for cold wallets is lower than the occurrence (5.3% of the attacked wallets versus 2% of the total losses), which seems to suggest that attacks on cold wallets are less profitable. However, the sample presented is too small to reach a clear conclusion and this result could also be due to the compromised key being less critical to the protocol or with less funds available.

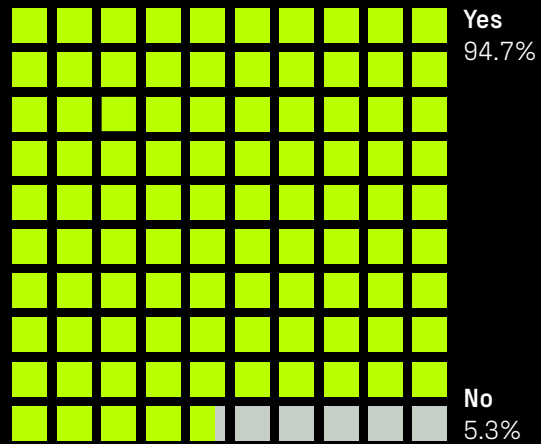


Figure 41: Usage of hot wallets [percentage]



Figure 42: Usage of hot wallets [count]

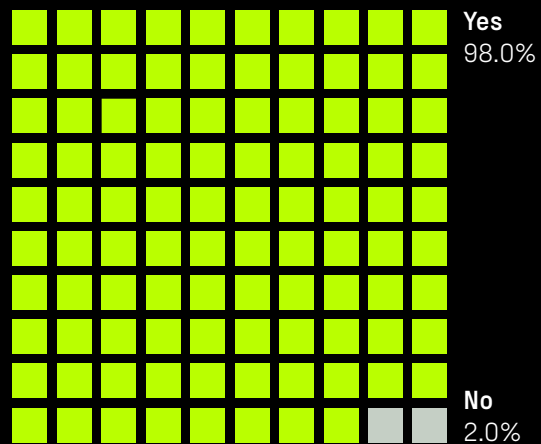


Figure 43: Loss caused by the usage of hot wallets [percentage]



Figure 44: Loss caused by the usage of hot wallets [USD]

By year, we can observe in **Figure 45** and **46**, that only in **2022** one of the attacks was on a private key stored in a cold wallet.

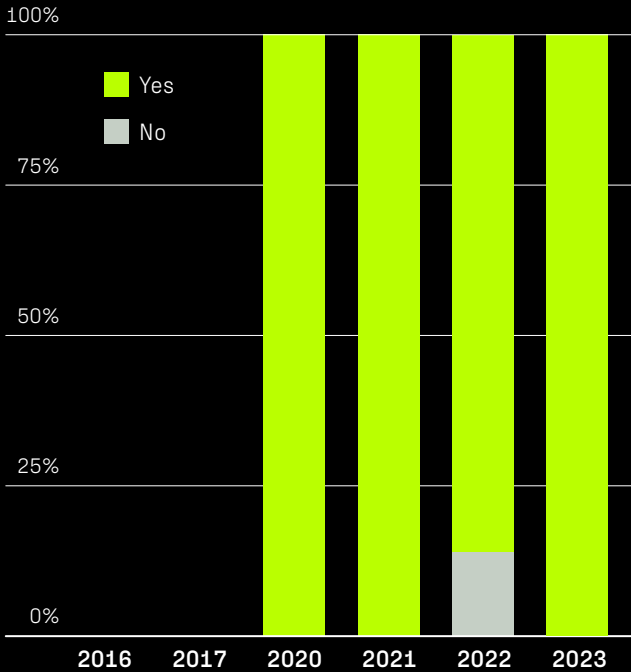


Figure 45: Usage of hot wallets by year [percentage]

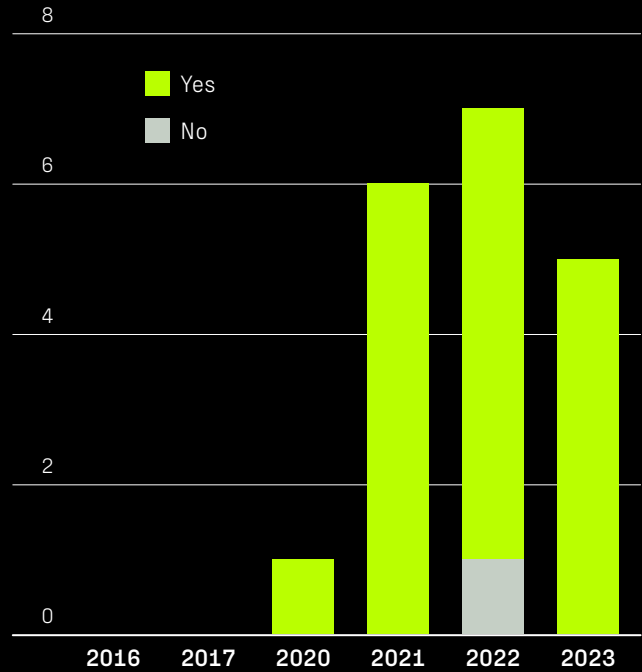


Figure 46: Usage of hot wallets by year [count]

Comparing loss by year (Figure 47 and Figure 48) doesn't really provide much extra information, it just reiterates what previous charts already suggested: the damage produced by a hack on a cold wallet seems to be smaller than for one of a hot wallet. Again, the sample is very small and this discrepancy can be due to other causes mentioned before.

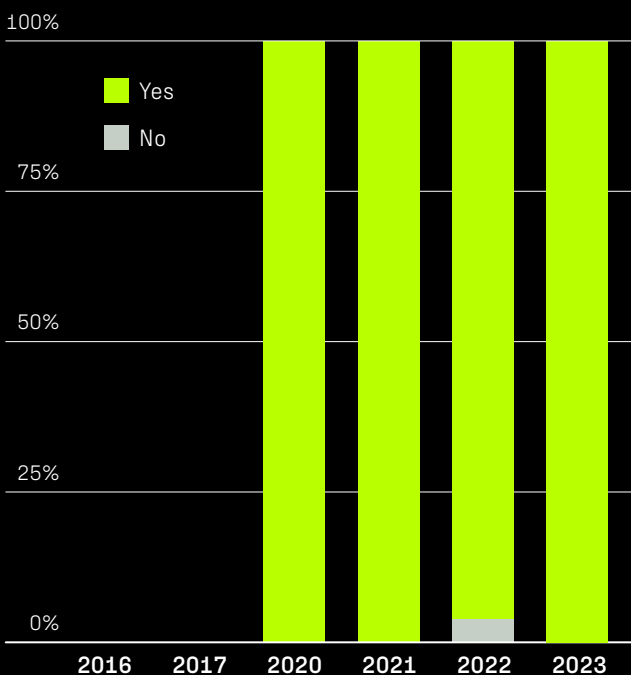


Figure 47: Loss caused by the usage of hot wallets by year [percentage]

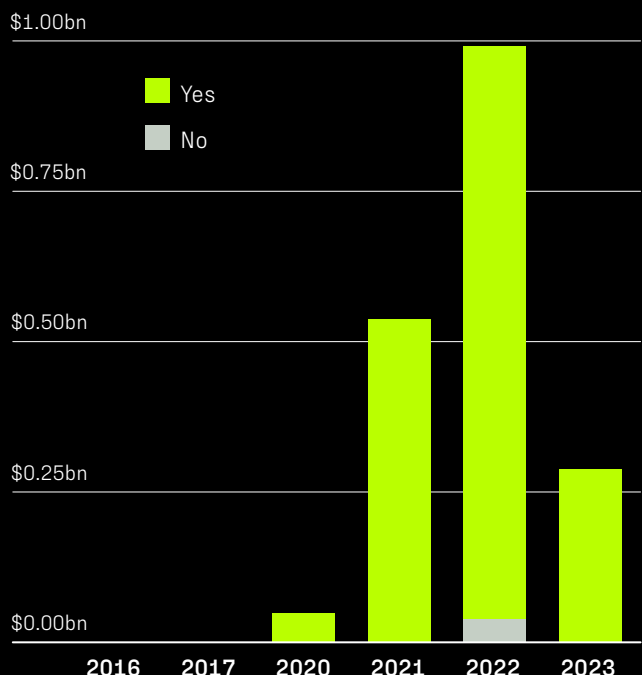


Figure 48: Loss caused by the usage of hot wallets by year [USD]

Use of flash loans

A flash loan is a type of loan where a user borrows assets with no upfront collateral and returns the borrowed assets within the same blockchain transaction.

It is known that they can and have been used as a method to execute attacks, but to what degree?

In general, we can see in Figures 49 and 50, that the majority of attacks that could involve the use of flash loans, as they interact with smart contracts, do not use them. However, this percentage is really close: 60% of the attacks do not make use of flash loans versus 40% of attacks that do.

Does the use of flash loans provoke higher losses? Figure 51 and Figure 52 try to answer that question. We can observe that attacks involving flash loans do not seem to produce bigger losses than those that do not use them, as they represent only 29.5% of the loss, approximately \$1,004,723,000 USD compared to 40% of the total attacks that have used them.

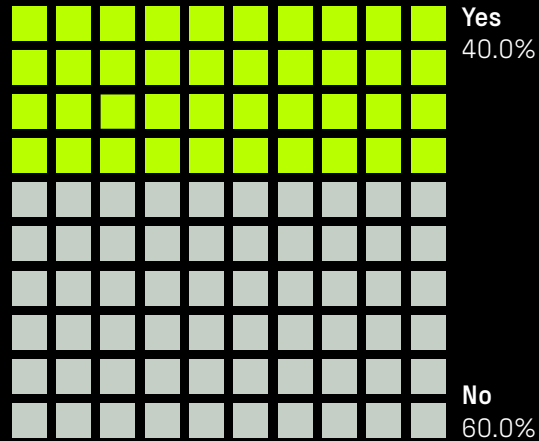


Figure 49: Usage of flash loans [percentage]



Figure 50: Usage of flash loans [count]

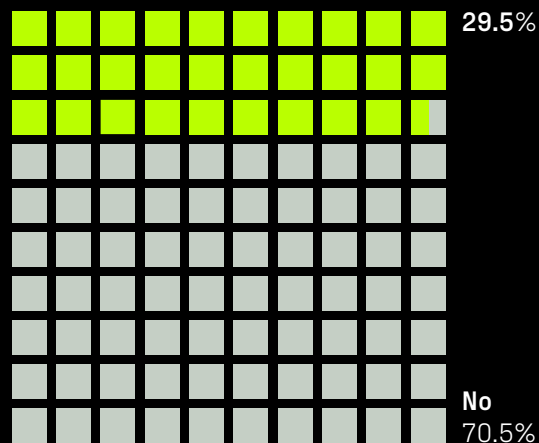


Figure 51: Loss caused by the usage of flash loans [percentage]

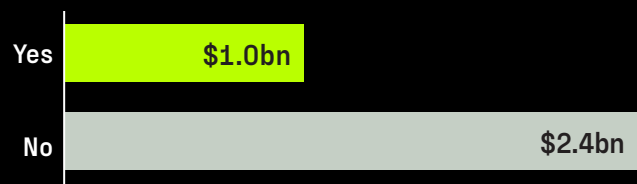


Figure 52: Loss caused by the usage of flash loans [USD]

If we examine the evolution of the use of this kind of mechanism through the years, we can observe that the use of flash loans seems to decline until 2022. (Figures 53 and 54)

Nevertheless, there has been an increase in their use in 2023, when flash loans were used in 62.5% of attacks.

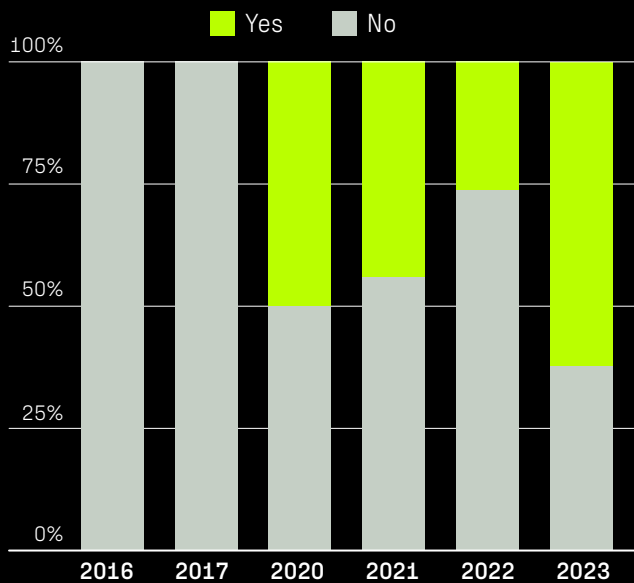


Figure 53: Usage of flash loans per year [percentage]

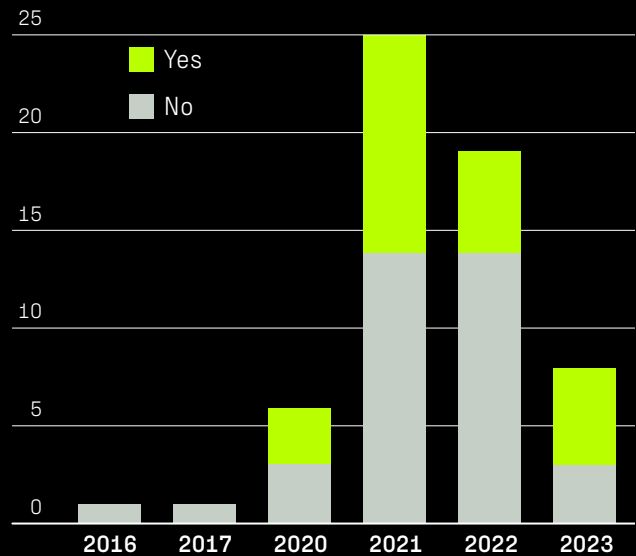


Figure 54: Usage of flash loans per year [count]

In terms of losses, the use of flash loans seems to follow the same trend as in the previous charts (Figures 53 and 54), decreasing until 2022 to later increase drastically in 2023, accounting for 54.1% of the total loss (\$233,300,000 USD, see Figures 55 and 56). However, this percentage is slightly lower than their rate of occurrence, so it might indicate that they are not as severe as it might seem.

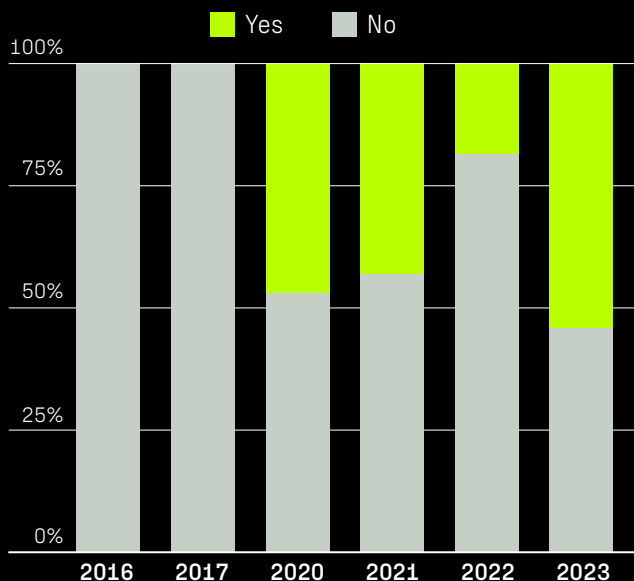


Figure 55: Loss caused by the usage of flash loans per year [percentage]

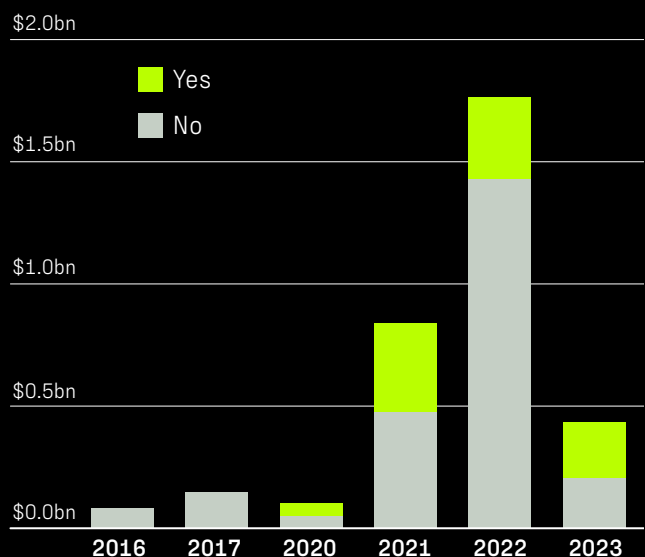


Figure 56: Loss caused by the usage of flash loans per year [USD]

Figures 57 and 58 depict the use of flash loans by type of attack sub-categories.

We can observe that for **compromised private keys, phishing, rug pulls** and **traditional hacks**, there is no use of flash loans. In **direct contract exploitation**, only 25% of the attacks use flash loans. However, the majority of **price manipulation** attacks (55.6%) and all of the **governance** attacks in our sample utilize them, although the number of **governance** attacks in our sample is really small and could not be representative of a trend in the use of this technology.

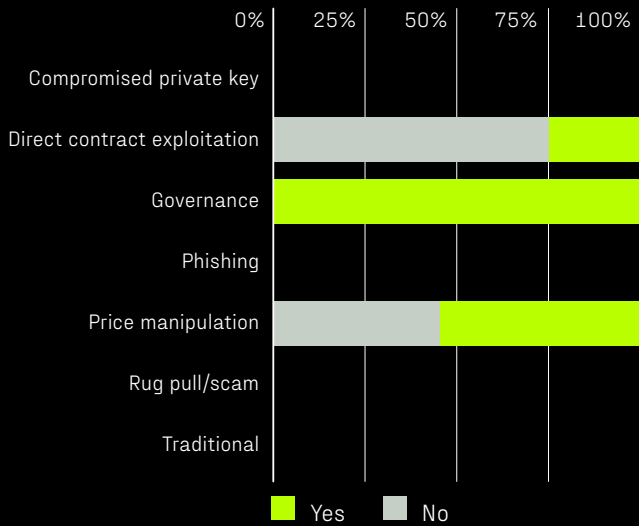


Figure 57: Usage of flash loans per type of attack sub-categories [percentage]

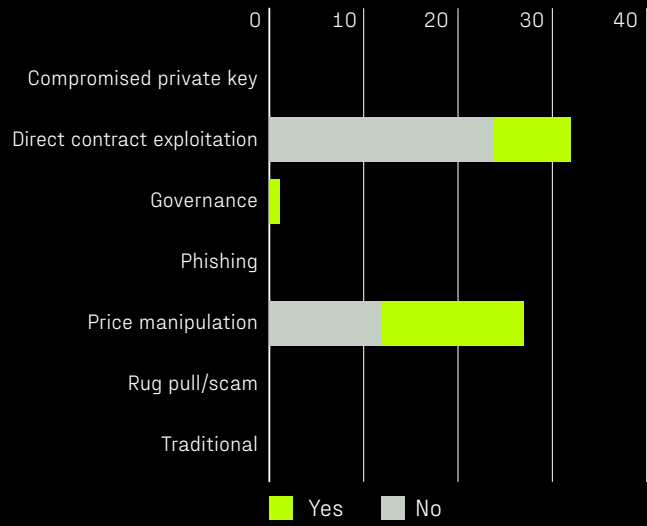


Figure 58: Usage of flash loans per type of attack sub-categories [count]

Comparing the losses caused by the attacks that employ flash loans, Figure 59 and Figure 60 show that, in the case of **direct contract exploitation**, they represent only 12.5% of the losses (\$297,800,000 USD), which is less than the percentage of attacks that use them. For **price manipulation** attacks, however, they represent 62.1% of the funds lost, which is around \$525,923,000 USD. This number is slightly higher than in the previous charts (Figures 57 and 58). This data could indicate that, in **price manipulation** attacks, the use of flash loans could lead to bigger losses while in **direct contract exploitation** they do not significantly impact the amount stolen.

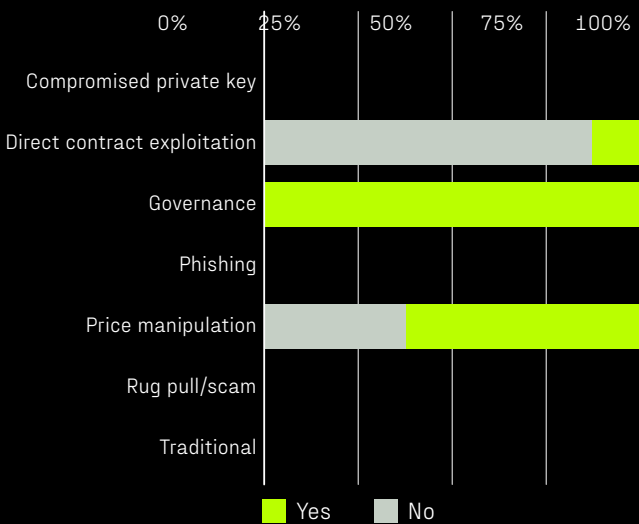


Figure 59: Loss caused by the usage of flash loans per type of attack sub-categories [percentage]

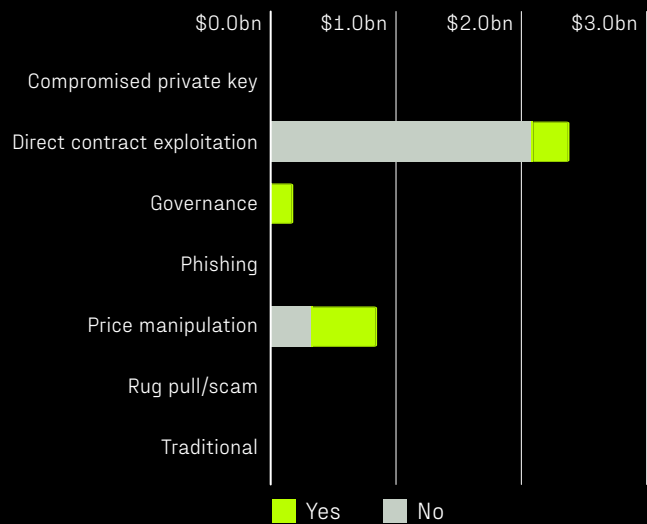


Figure 60: Loss caused by the usage of flash loans per type of attack sub-categories [USD]

Over the years, we can observe in **Figures 61 and 62** that the use of flash loans seemed to be in decline until 2022 since its first appearance in 2020, specifically in **price manipulation attacks**.

However, there has been a huge increase in their use in 2023, with them appearing in 66.7% of the **direct contract exploitation** attacks and 60% of the **price manipulation** ones, adding up to a total of 62.5% of the attacks.

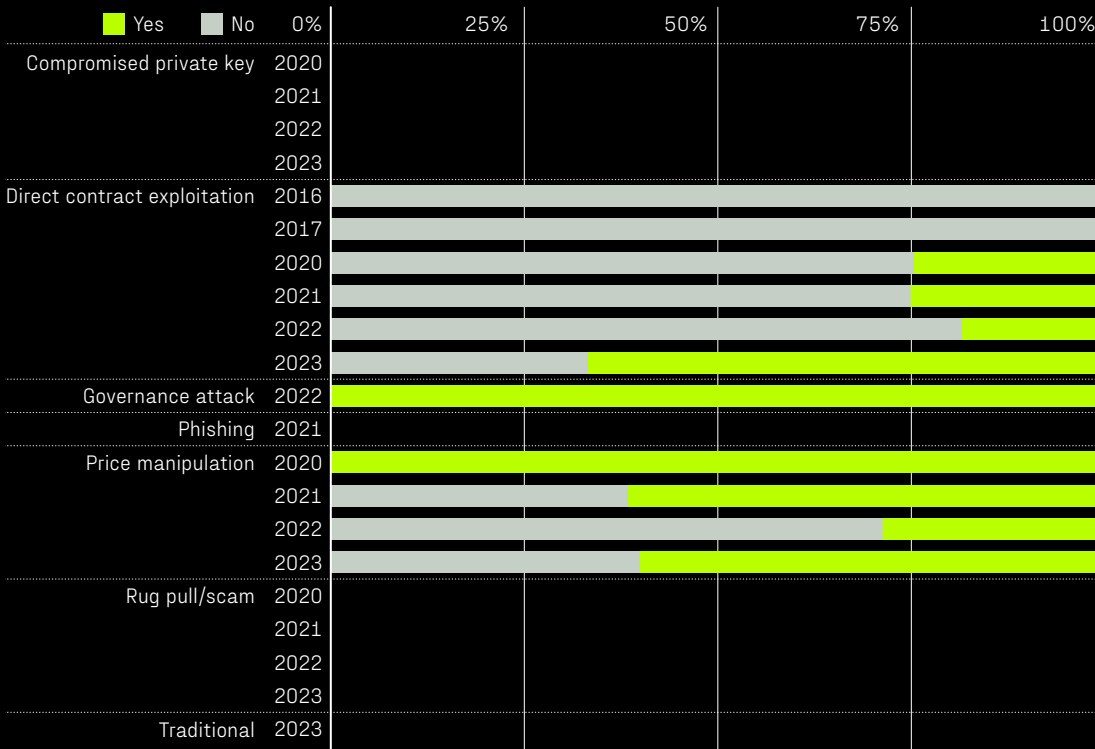


Figure 61: Usage of flash loans per type of attack sub-categories and year [percentage]

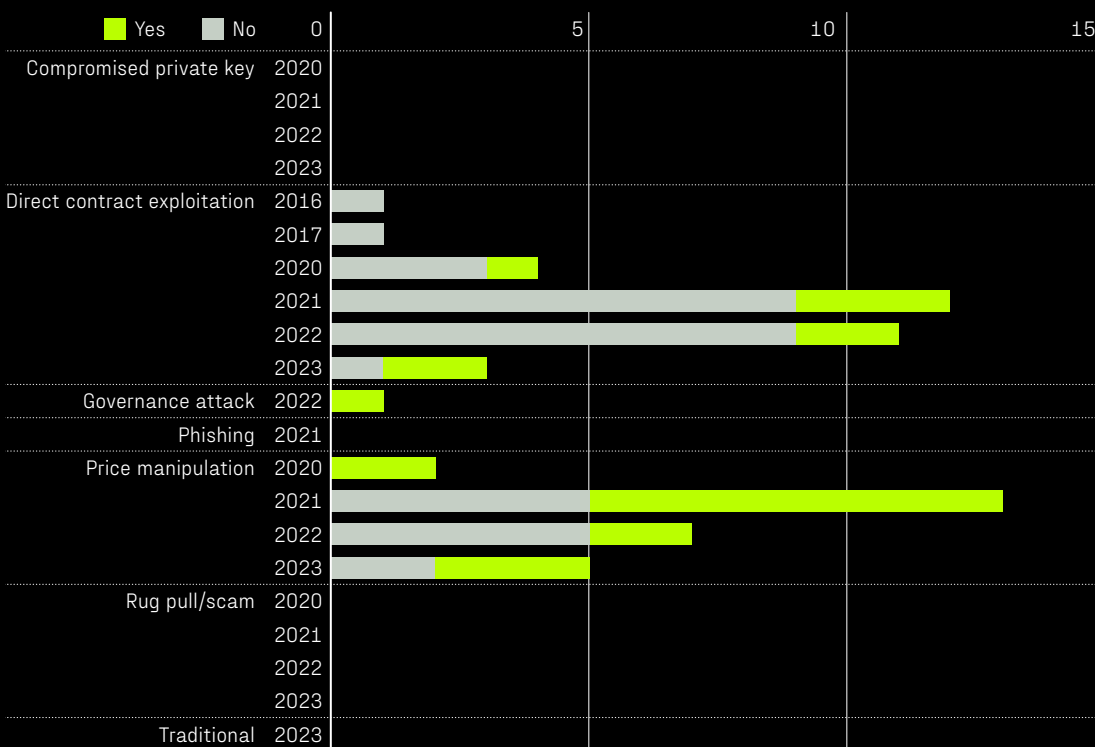


Figure 62: Usage of flash loans per type of attack sub-categories and year [count]

If we observe how the losses evolved through the years for each sub-type of attacks (Figures 63 and 64), it is interesting that, while historically price manipulation attacks are the ones that accumulate higher losses due to flash loans attacks, this trend seems to be changing through the years, changing from 100% in 2020 (\$40,000,000 USD) and 70% in 2021 (\$289,700,000 USD) to only 34.5% in 2023 (\$67,823,000 USD).

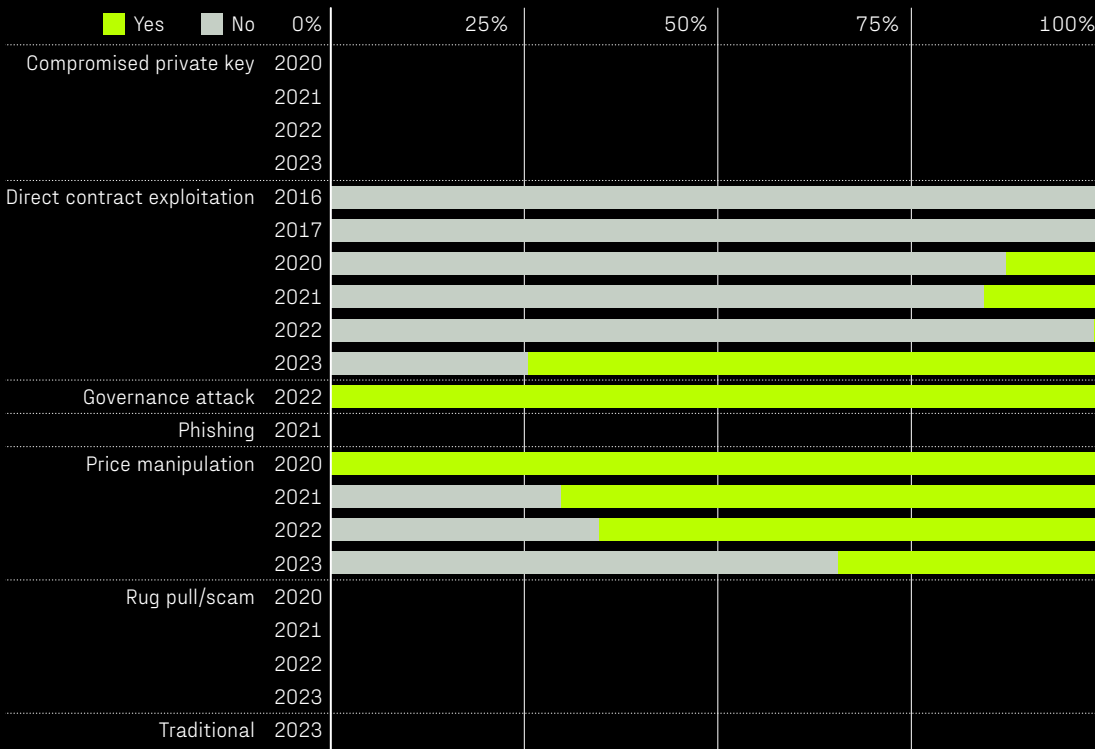


Figure 63: Loss caused by the usage of flash loans per type of attack sub-categories and year [percentage]



Figure 64: Loss caused by the usage of flash loans per type of attack sub-categories and year [USD]

Root cause analysis

Each type of attack could have been possible because of different causes.

First, we want to explain those causes related to exploiting a smart contract vulnerability. These can be the cause of **direct contract exploitation**, **governance** and some **price manipulation** attacks.

We will consider the following:

- **Math error:** Math errors are when an error in a mathematical formula or in the calculation process occurs, for example, rounding mistakes.
- **Lack of/faulty input verification/validation:** A contract is exploited in this category when there is a lack or faulty verification or validation of some input argument for a function call, for example, not checking that two assets supplied are not the same or the zero address.
- **Reentrancy:** This is one of the most common attacks in smart contracts. It consists of an attacker calling a function recursively in order to damage the protocol, often by stealing funds.
- **Faulty proof verification:** Especially relevant in bridges and other cross-chain protocols, it occurs when there is a faulty verification proof on one chain which allows the attacker to falsify actions on the other paired chain. For example, by an incorrect implementation of the signature verification algorithm.
- **Faulty initialization:** Especially relevant for proxy contracts. Occurs when a contract is left uninitialized, or it is initialized with the wrong arguments.
- **Configuration error:** When some contract configuration parameters are wrong in the code, and therefore introduce a vulnerability. For example, using the wrong address for a token contract.
- **EVM based:** Exploiting a contract by taking advantage of how the EVM works. For example, by repeatedly making the contract to deploy other contracts until a certain contract address is generated or taking advantage of how the ABI decoder works.
- **Lack of/faulty access control:** The caller's ability to execute the function is not properly set or checked. For example, a function that should be executed only by certain roles is left free for anyone to be called.
- **Flawed proposal execution mechanism:** When in governance, the proposal execution process is flawed. For example, allowing an attacker to directly execute malicious proposals without a community review after completing the voting.
- **Logic error:** Any other kind of programming error that results in contract exploitation.

Figures 65 and 66 show the distribution of these vulnerabilities among the previous sub-categories.

It can be seen that the most common attack cause is a **lack of or faulty input verification or validation**, accounting for 25.5% of the attacks. The second most common cause is a **logic error** in the contracts, with 17% of the total. The third one is **reentrancy** with 14.9%. The rest appear in lesser percentages, with **governance-related vulnerabilities** and **configuration errors** being the least common.

Calculating the loss produced by each sub-category could be difficult because, although normally an attack can be attributed to a single type of vulnerability, there is one case, namely Alpha Finance, in which the attacker actually took advantage of different vulnerabilities to carry out the attack. In this case, the losses produced have been assigned to each of them to better represent lost funds by type. However, because of this the sum of the total values is higher because of the repeated amounts.

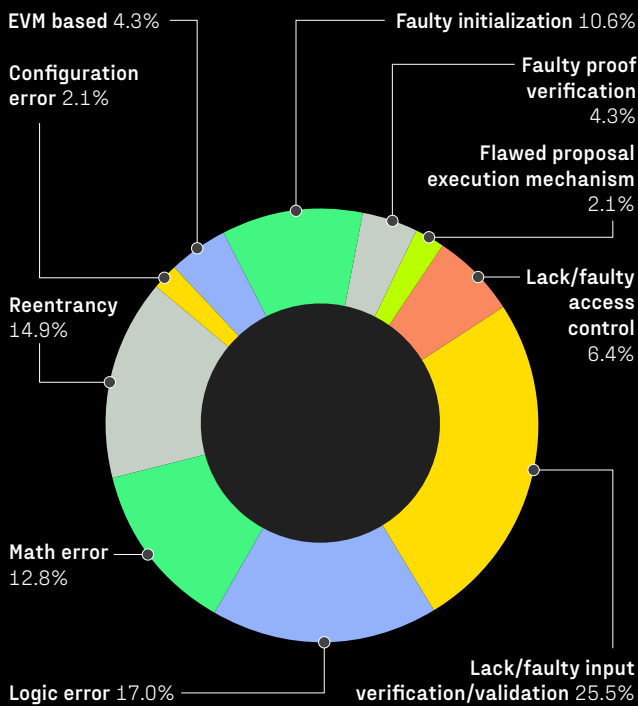


Figure 65: Number of type of vulnerabilities in contracts [percentage]

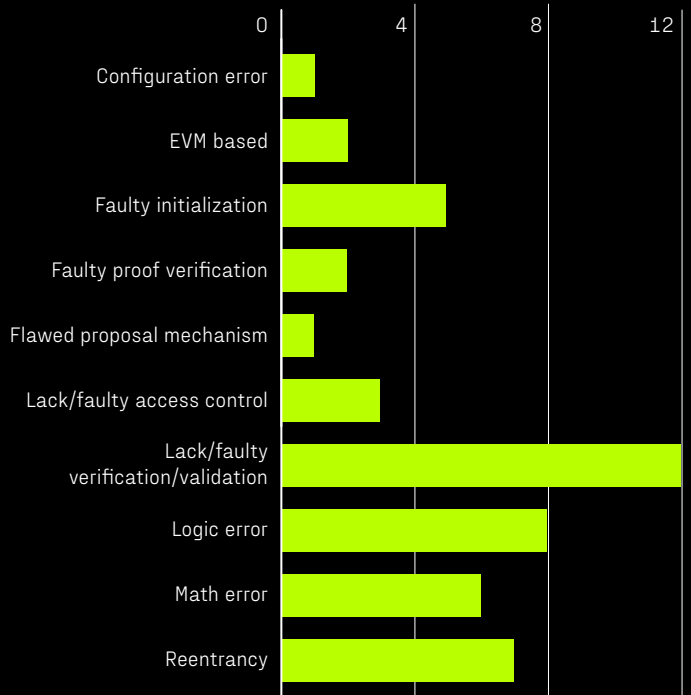


Figure 66: Number of type of vulnerabilities in contracts [count]

Figure 67 and Figure 68 show how losses are distributed by type of vulnerability.

We can observe that the vulnerability that causes the most monetary losses seems to also be a **lack of or faulty input verification or validation**, with 28% of the total (around \$993,800,000 USD). The second, however, is **faulty proof verification**, which amounts to 25.7% of the losses (\$912,000,000 USD) but only 4.3% of occurrences. This could suggest that a successful exploit of this kind of vulnerability usually leads to larger losses than many of the other categories and are, in general, severe. The third place is for **math errors** leading to \$381,546,000 USD in losses. The percentage of occurrence is pretty similar in these cases. The second and third places by occurrence, **logic errors** and **reentrancy**, actually produce relatively less loss than their occurrence.

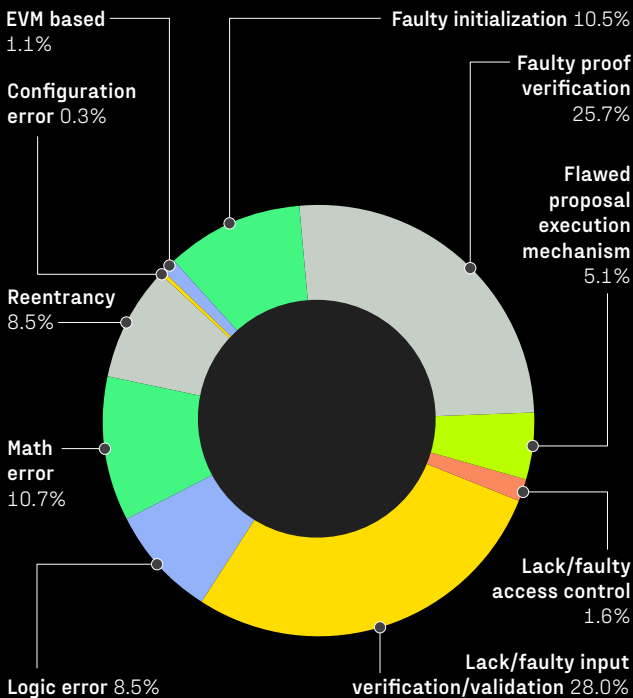


Figure 67: Loss caused by type of vulnerabilities in contracts [percentage]

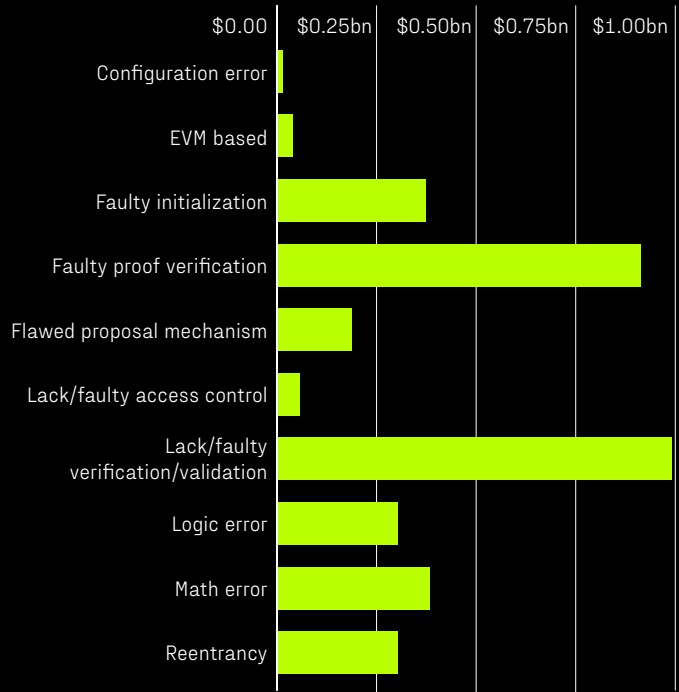


Figure 68: Loss caused by type of vulnerabilities in contracts [USD]

As time passes, it can be observed that the main cause of attacks related to contracts has changed (Figures 69 and 70).

In the beginning, the main cause was the **reentrancy** exploit of The DAO, followed by the **faulty initialization** vulnerability of the Parity multi-sig wallet. In 2020, **lack of/faulty input verification/validation** and **reentrancy** share the same percentage. From 2021 onward, causes diversify a bit more, but **lack of/faulty input verification/validation** accounts for the majority of incidents until 2023, in which the main cause is some kind of logic error in the code.

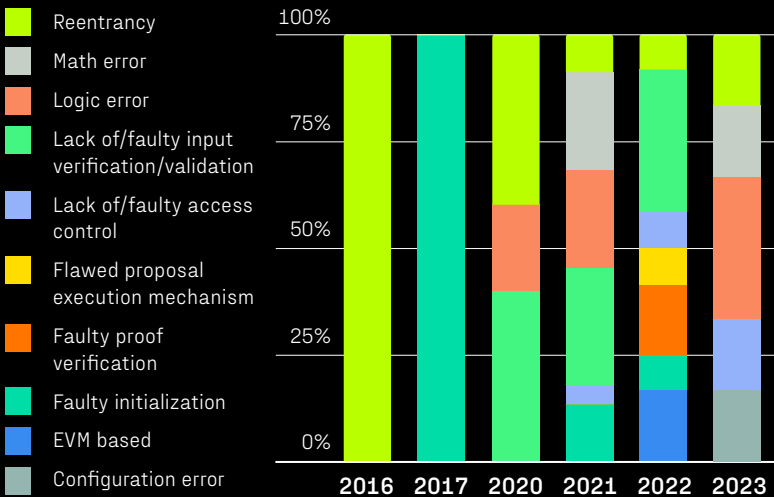


Figure 69: Number of type of vulnerabilities in contracts per year [percentage]

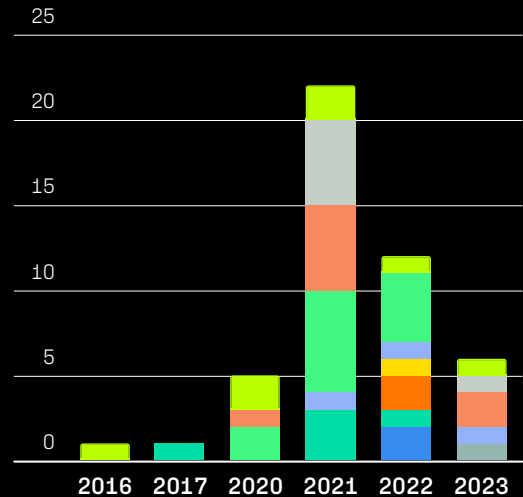


Figure 70: Number of type of vulnerabilities in contracts per year [count]

Figures 71 and Figure 72 show how much was stolen per year because of each vulnerability. As mentioned before, for hacks involving more than one type of vulnerability, the amount is repeated. We can see how, until 2020, the distribution of the losses is proportional to their rates of occurrence. In 2021, we observe how **lack of/faulty input verification/validation** vulnerabilities actually produced 61.9% of the losses for that year (\$802,100,000 USD) while making up 33.3% of attacks. For 2022, the same happens with **faulty proof verification** vulnerabilities, making up 57.9% of losses (around \$912,000,000 USD) versus 16.7% of attacks. This seems to support the theory that exploiting this type of vulnerability generally leads to major losses for the protocol. In 2023, **logic errors** also caused more loss than their occurrence percentage, 52.7% (\$205,500,000 USD) versus 33.3%.

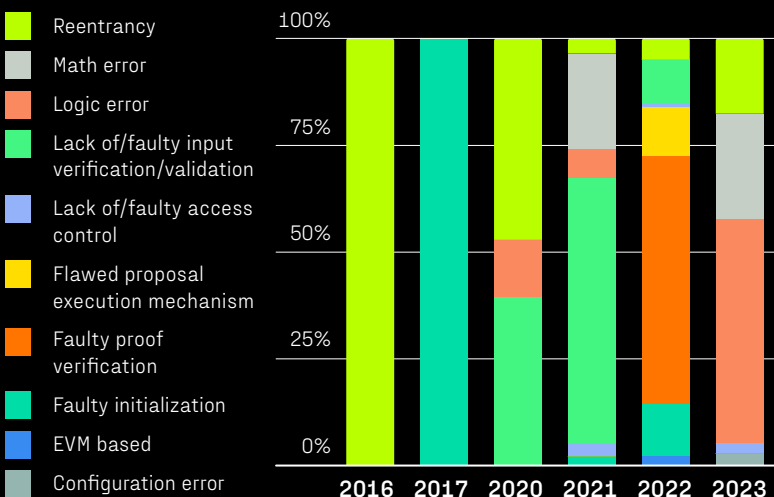


Figure 71: Loss caused by type of vulnerabilities in contracts per year [percentage]

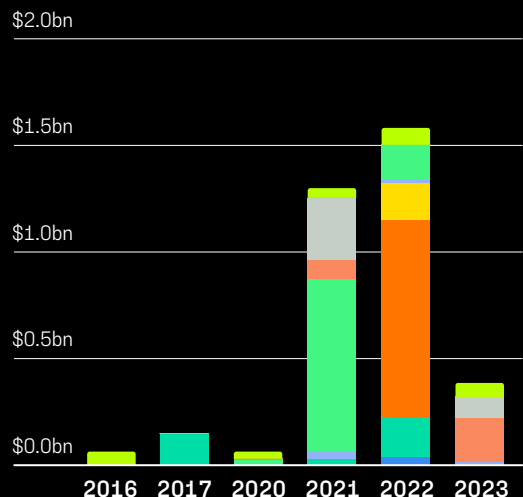


Figure 72: Loss caused by type of vulnerabilities in contracts per year [USD]

Once we have established these common causes, each sub-category will be studied separately.

Direct contract exploitation

Direct contract exploitation in the studied sample is possible because of the vulnerabilities previously explained.

If we observe those that are most commonly used, we are left with those shown in Figures 73 and 74. We can still see that the most common cause of attack is a **lack of/faulty input verification/validation** (26.5%). However, for **direct contract exploitation**, **reentrancy** is the second most common one with 20.6% of the total, followed by **logic errors**.

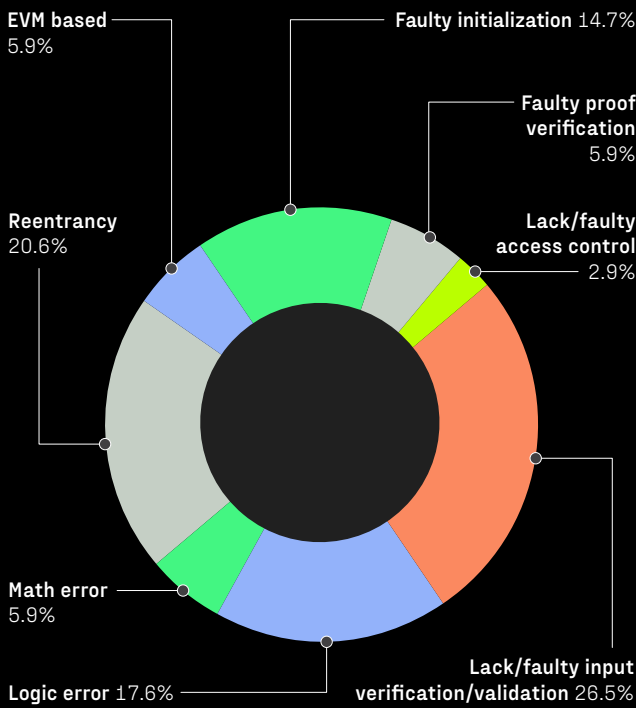


Figure 73: Number of type of vulnerabilities in direct contract exploitation [percentage]

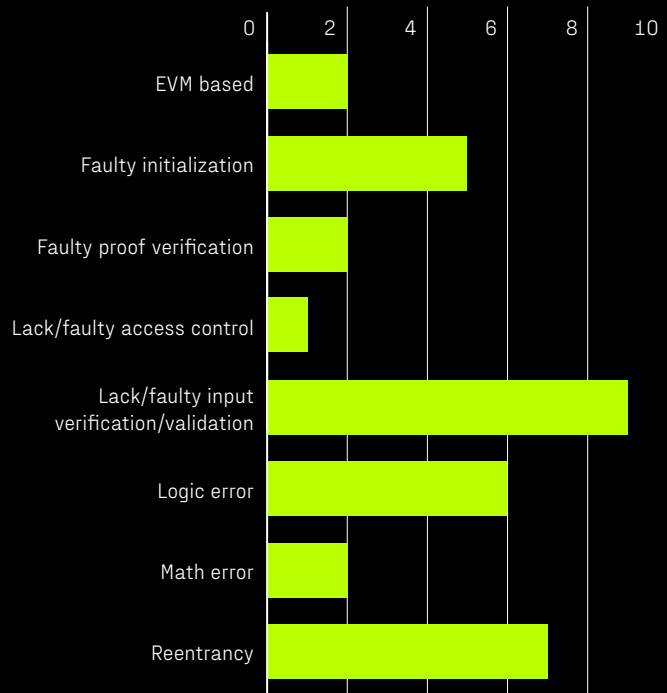


Figure 74: Number of type of vulnerabilities in direct contract exploitation [count]

If we examine the amount lost due to each vulnerability, we can see in **Figures 75 and 76**, that **faulty proof verification** accounts for **30.4%** of the total amount lost (\$912,000,000 USD), against **5.9%** of the occurrences, suggesting again that this kind of attack can lead to **severe damage when directly exploiting a contract.**

The second most common is a **lack of or a faulty input verification or validation**, with 30% of the losses (\$900,900,000 USD) which is really close to its occurrence percentage. The third most common vulnerability is **faulty initialization**, adding up to 12.4% of losses (\$372,950,000 USD), also close to its occurrence rate.

If we look at the evolution through the years (Figures 77 and 78), we can see how reentrancy seemed to be very prominent, only to decrease by 2022 to 9.1% and then increase again in 2023. Lack of/faulty input verification/validation was a threat mostly in 2020 and 2022. In 2023, most of the hacks were enabled by logic errors at 66.7%.

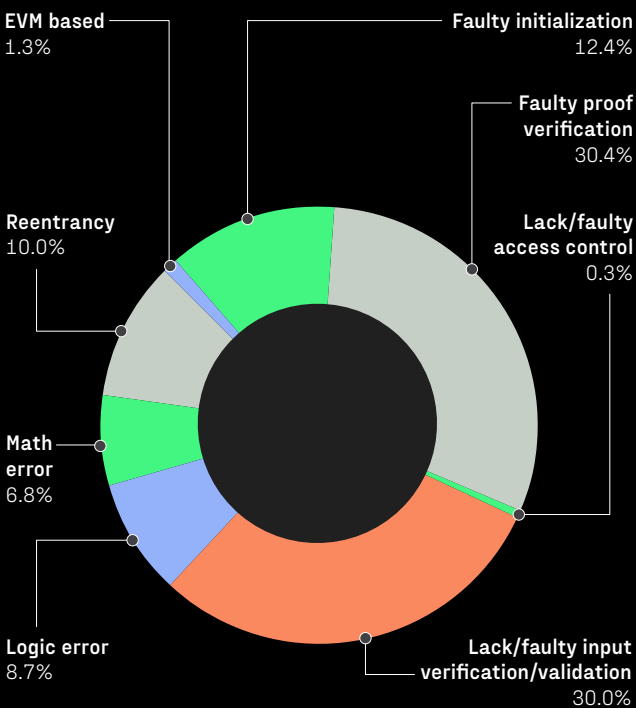


Figure 75: Loss caused by type of vulnerabilities in direct contract exploitation [percentage]

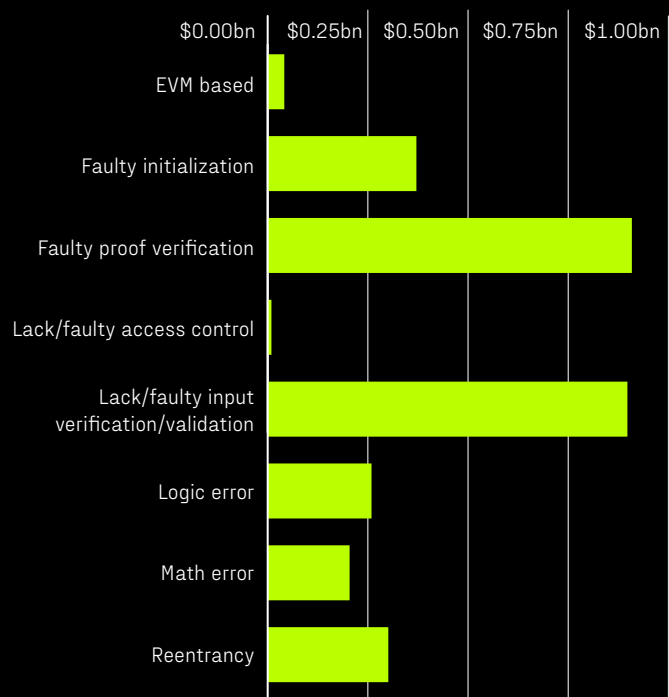


Figure 76: Loss caused by type of vulnerabilities in direct contract exploitation [USD]

If we look at the evolution through the years (Figures 77 and 78), we can see how **reentrancy** seemed to be very prominent, only to decrease by 2022 to 9.1% and then increase again in 2023.

Lack of/faulty input verification/validation was a threat mostly in 2020 and 2022. In 2023, most of the hacks were enabled by **logic errors** at 66.7%.

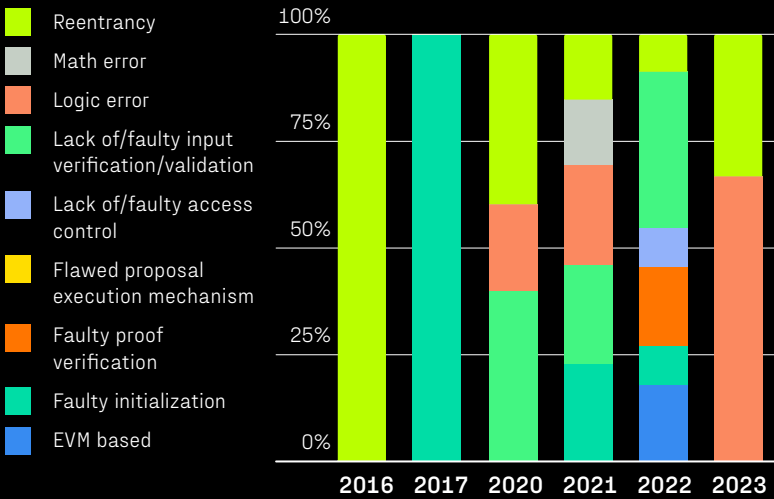


Figure 77: Number of type of vulnerabilities in direct contract exploitation per year [percentage]

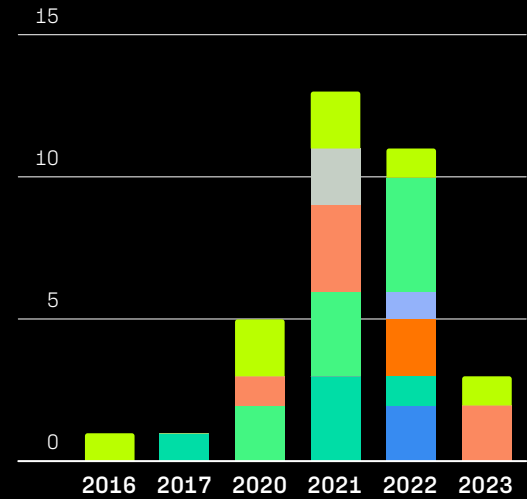


Figure 78: Number of type of vulnerabilities in direct contract exploitation per year [count]

We can see in Figures 79 and 80 that, until 2020, the distribution of losses is very similar to vulnerabilities' rates of occurrence. In 2021, we observe how a **lack of/faulty input verification/validation** caused 68.1% of the losses for that year (\$709,200,000 USD) compared to the 23.1% rate of occurrence. In 2022, something similar happened with **faulty proof verification**, accounting for 65.4% of losses (around \$912,000,000 USD) versus 18.2% of occurrences. This seems to support the theory that exploiting this type of vulnerability generally leads to major losses for the protocol. In 2023, loss proportions are similar to those for occurrence, being the losses caused by **logic errors** slightly higher than their rate of occurrence (74.8% versus 66.7%).

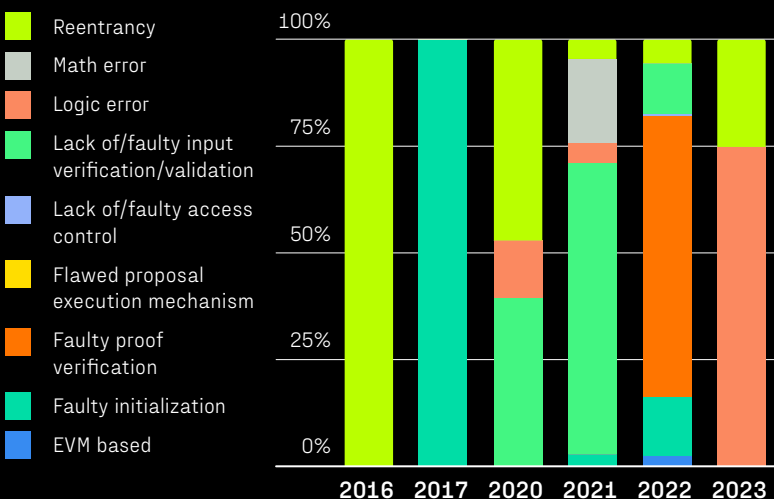


Figure 79: Loss caused by type of vulnerabilities in direct contract exploitation per year [percentage]

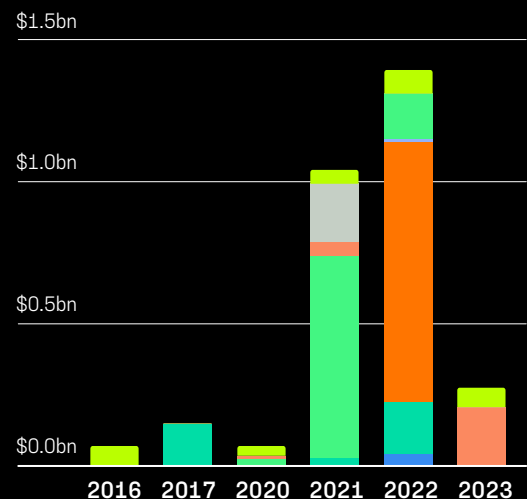


Figure 80: Loss caused by type of vulnerabilities in direct contract exploitation per year [USD]

Price manipulation attacks

Price manipulation attacks can be possible because an attacker takes advantage of a vulnerability in a smart contract.

It can also have other causes. In this research we will consider the following:

- Flawed oracle:** Oracles provide a way to access existing data sources, systems, and complex computations outside. A flawed oracle exists when the party is negligent or malicious or can be easily exploited or manipulated. This can happen, for example, when the oracle’s data source is compromised. Thus, having as many different data sources as possible is desirable, because it is more difficult to compromise all of them. A good oracle would also protect itself from external tampering and single points of failure via decentralization and provide incentives to the user to report in a faithful way.

- Low liquidity in pool:** The cause of these attacks is a low number of tokens in a pool. Pools with low liquidity are vulnerable to **price manipulation** attacks primarily due to the ease with which a relatively small amount of capital can significantly impact the price of the assets within the pool.

Figures 81 and 82 show the main causes of **price manipulation** attacks. We can observe that the majority of them are possible because of a **flawed oracle** (50%). After that, taking advantage of a contract vulnerability (especially **math errors**) is the second most common. Hacks due to low **liquidity in the pool accounts** for 12.5% of the total.

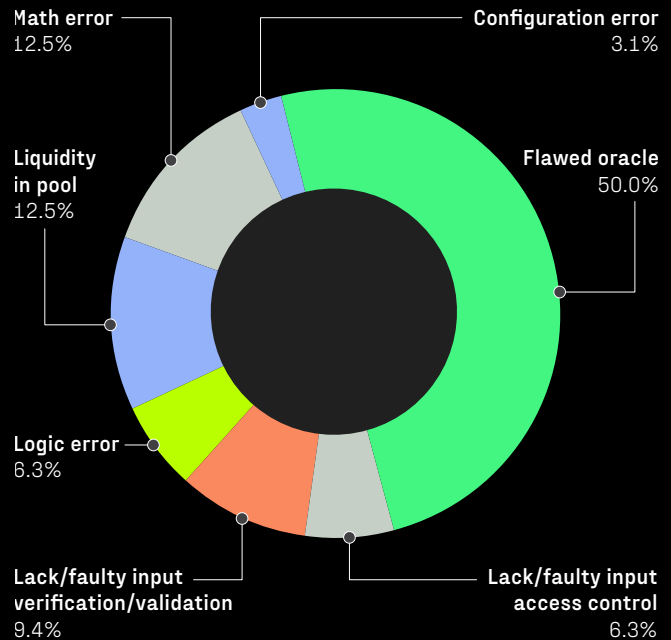


Figure 81: Number of type of vulnerabilities in price manipulation attacks [percentage]

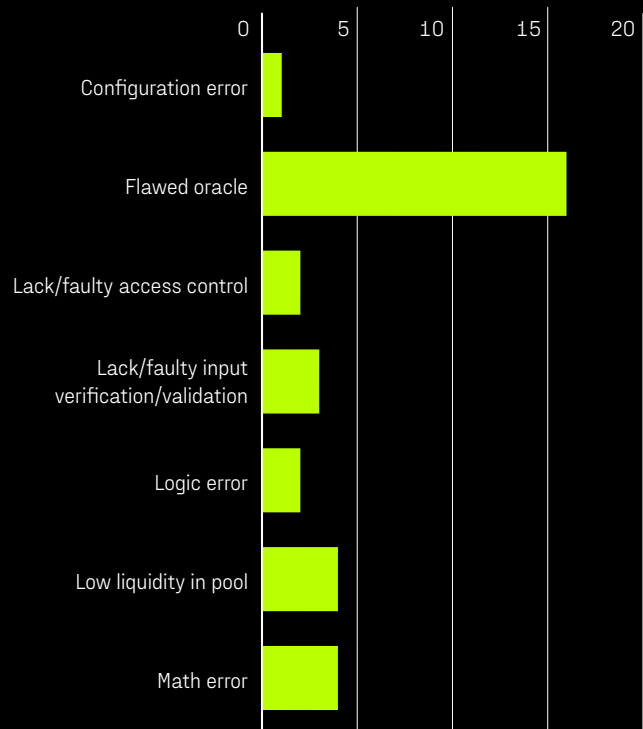


Figure 82: Number of type of vulnerabilities in price manipulation attacks [count]

If we observe the distribution by loss (Figures 83 and 84), we can see that the primary cause of losses is also the most common by occurrence.

This is a **flawed oracle**, causing 49.3% of the lost value (\$525,000,000 USD). The second most common cause of losses is also **math errors**, with 16.7% (\$177,546,000 USD). The third cause of major losses is **low liquidity in a pool**, with 16% of the total losses (\$169,900,000 USD). In general, these percentages are close to their rates of occurrence.

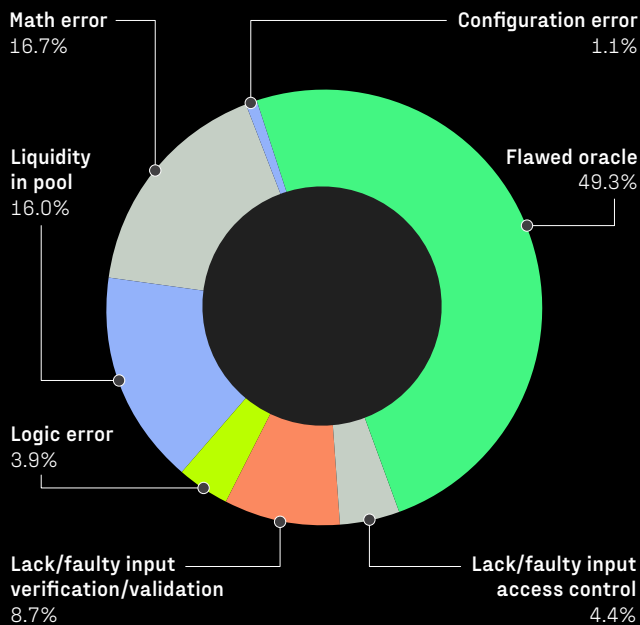


Figure 83: Loss caused by type of vulnerabilities in price manipulation attacks [percentage]

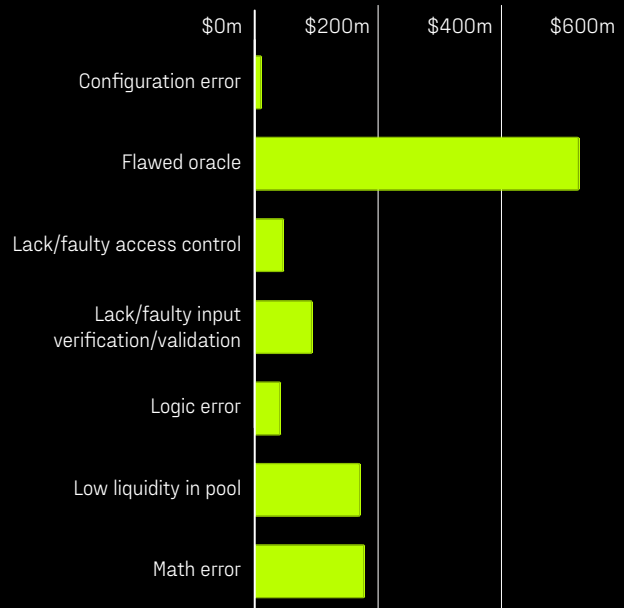


Figure 84: Loss caused by type of vulnerabilities in price manipulation attacks [USD]

Through the years, as we can observe in Figures 85 and 86, that, until 2023, the main cause of price manipulation attacks was the use of a flawed oracle by the protocol.

In 2023, however, it seems that causes related to exploiting a contract are the most used. This year's cause of hacks is divided equally among taking advantage of a **math error, a flawed oracle, low liquidity in a pool, configuration error, and a lack of/faulty access control** to a function.

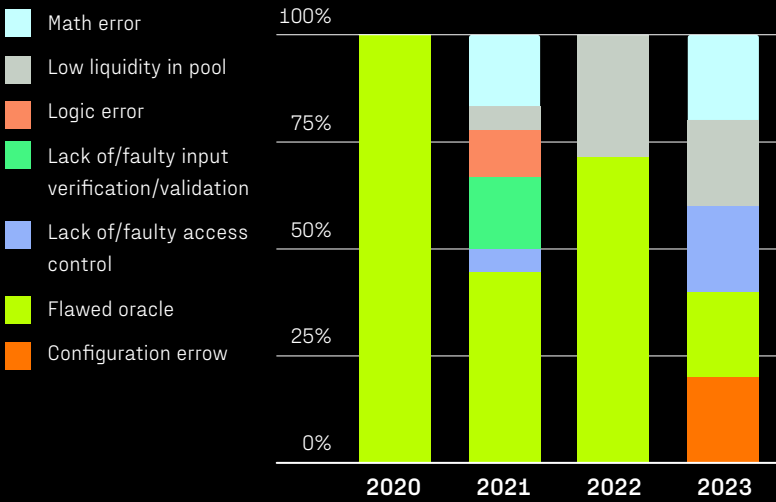


Figure 85: Number of type of vulnerabilities in price manipulation attacks per year [percentage]

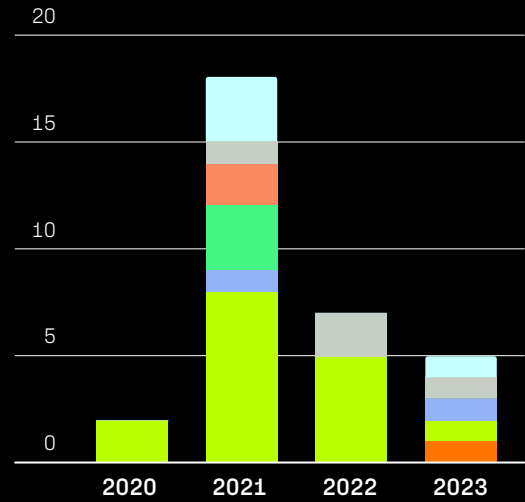


Figure 86: Types of vulnerabilities in price manipulation attacks per year [count]

We can see that losses are very similar in percentage to their occurrence until 2022 (Figures 87 and 88), in which the majority of the loss is caused by **low liquidity in the pool** (62.9%, \$123,400,000 USD) instead of a **flawed oracle**. In 2023, we can observe that the majority of losses are caused by the latter, accounting for 49.1% (\$120,000,000.00 USD).

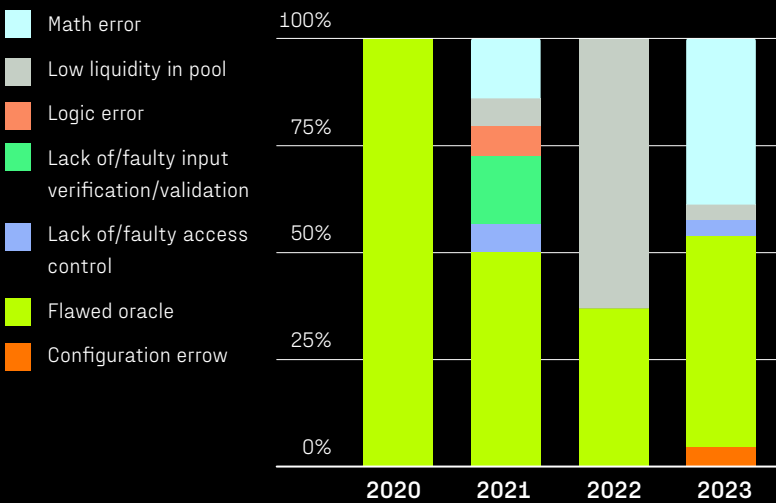


Figure 87: Loss caused by type of vulnerabilities in price manipulation attacks per year [percentage]

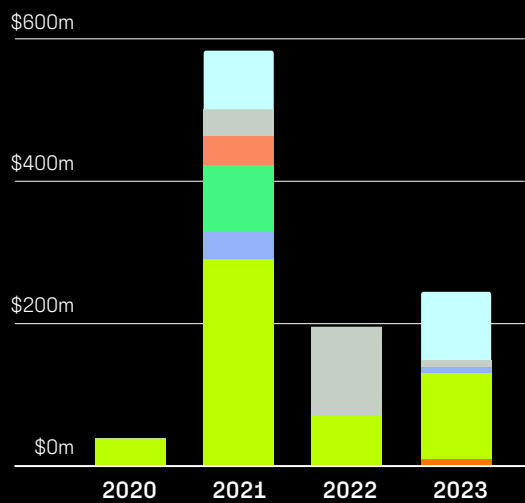


Figure 88: Loss caused by type of vulnerabilities in price manipulation attacks per year [USD]

Governance attacks

In our sample, we only have one protocol whose hack was possible because of a **governance** attack, with a **flawed proposal** execution mechanism being the main cause of it.

Rug pull/scams

Projects can **rug pull or scam** their users in different ways. We have observed the following main methods:

- **Privileged owner account:** The project was able to rug pull because they have the keys to an account with admin or other kind of higher permissions set on a project's hot wallet. The project used those accounts to perform actions in already deployed contracts or retrieve funds.
- **Upgrade to a malicious contract:** The project exited by upgrading the protocol to a malicious contract that allowed them to perform malicious actions.
- **Malicious library:** The project used a malicious library that was not verified to steal the funds.
- **Challenge key knowledge:** The project has knowledge of some secret (key) required by the code that allowed them to perform the theft of funds.

In **Figures 89 and 90**, it can be seen that the main way in which a project executes a rug pull is because they control a **privileged owner account (70%)**. Other causes are mostly project-specific.

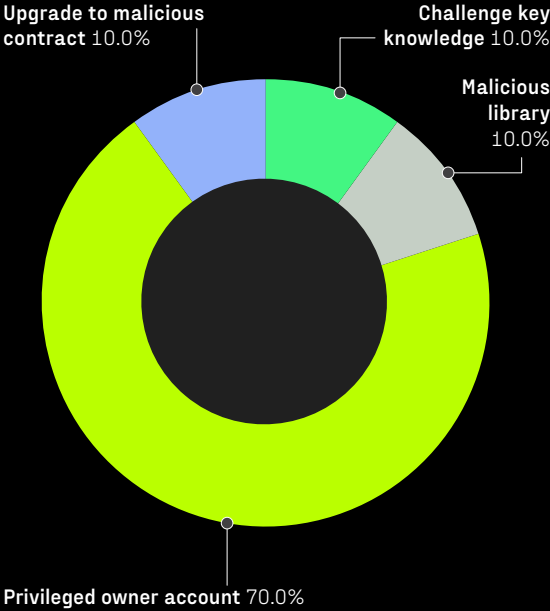


Figure 89: Number of methods used in rug pulls/scams [percentage]

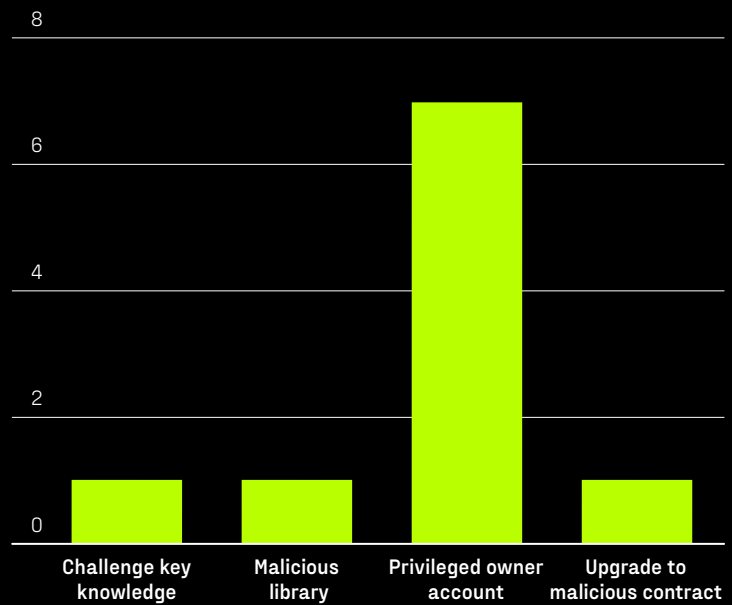


Figure 90: Number of methods used in rug pulls/scams [count]

With regard to losses, we can observe in Figures 91 and 92 that the distribution is quite similar, with a **privileged owner account** being the main attack vector, 90.2% (\$709,466,000 USD).

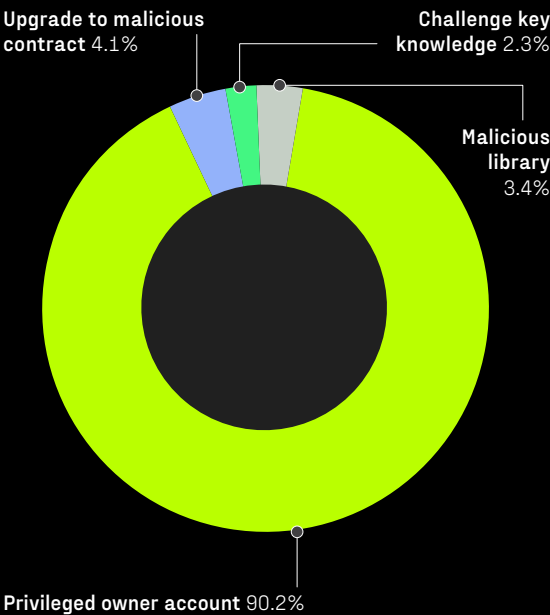


Figure 91: Loss caused by methods used in rug pulls/scams [percentage]

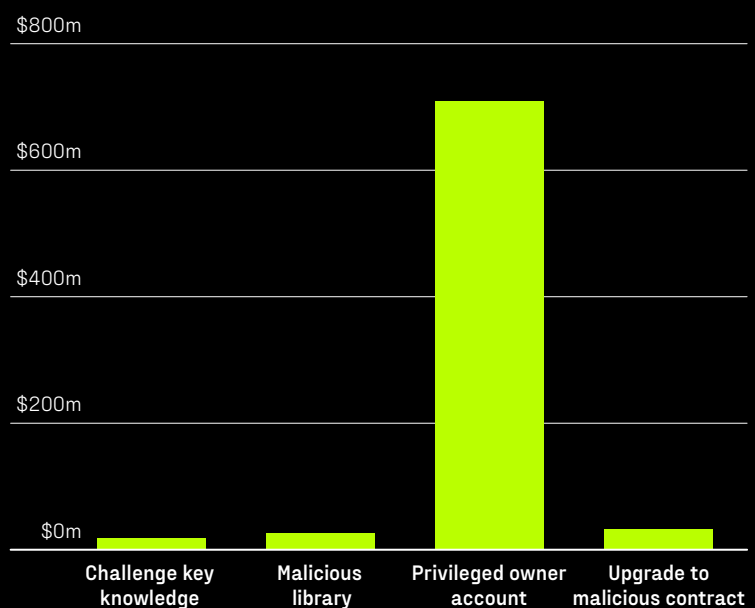


Figure 92: Loss caused by methods used in rug pulls/scams [USD]

Through the years, we can see that having a **privileged owner account** has been the main way projects rug pulled, although, in 2021, they used the other methods too **(Figures 93 and 94)**

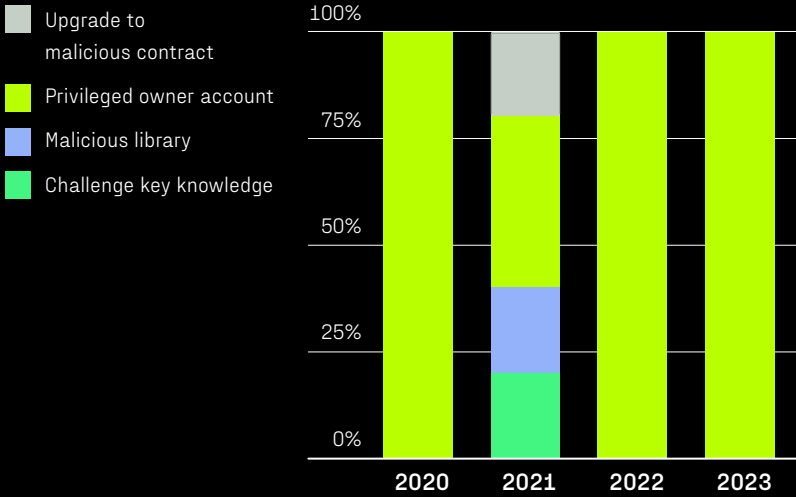


Figure 93: Number of methods used in rug pulls/scams per year [percentage]

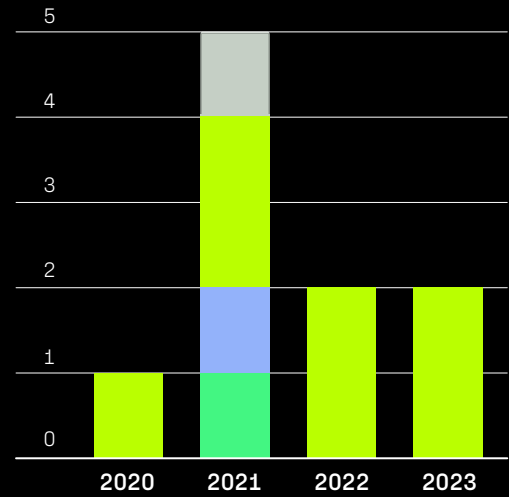


Figure 94: Number of methods used in rug pulls/scams per year [count]

The distribution by loss is pretty similar to the one by occurrence, although the loss produced by a privileged owner account in 2021 is slightly bigger than its occurrence, with 53% of the loss (\$87,000,000 USD) against 40% of occurrence (Figure 95 and 96).

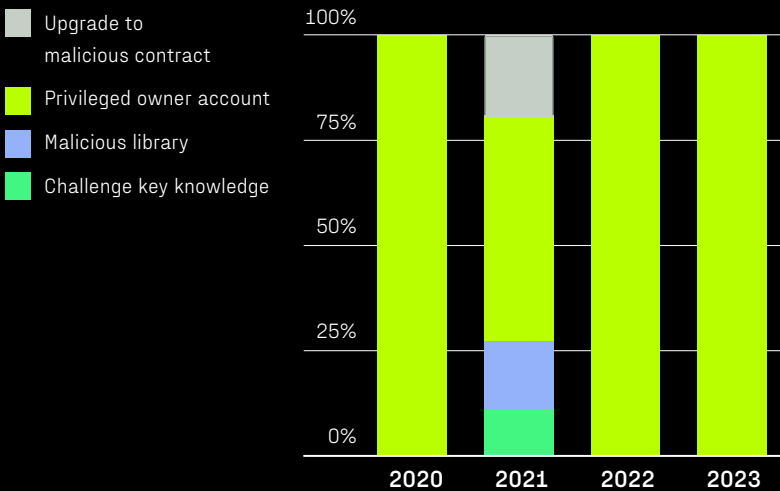


Figure 95: Loss caused by methods used in rug pulls/scams per year [percentage]

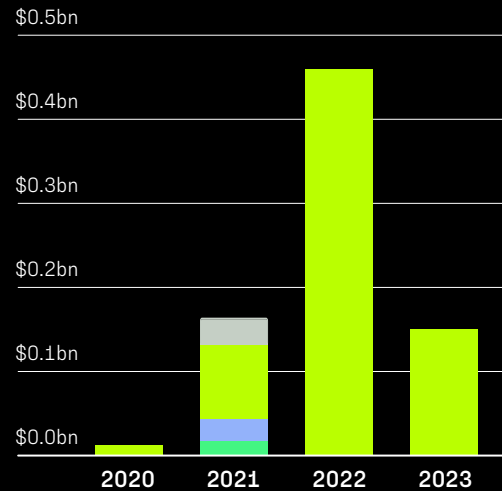


Figure 96: Loss caused by methods used in rug pulls/scams per year [USD]

Compromised private key

Private keys can be compromised in different ways:

- **2FA bypass:** Bypass of a protocol's two-factor authentication to retrieve the key.
- **Flawed key generation:** Bad implementation of the algorithm used to generate the key, which decreases the difficulty of obtaining it.
- **Compromised server:** The server where the key or mnemonic was stored was compromised, and the attacker was able to retrieve it.
- **MITM attack:** The attacker was able to obtain the key because of a Man in the Middle attack.
- **Social engineering attack/Phishing:** The attack was possible because of a social engineering or phishing attack, for example, disguising a malicious mail as a legit one to one of the project's developers.
- **Third-party hack:** A third party, outside the project's or users' control, was compromised, and that led to the theft of the key. For example, a breach of a third-party database.
- **Unknown:** The method used to steal the key is not known.

Figures 97 and 98 show how private keys are commonly stolen.

We can observe that, in the majority of cases (65.4%), how these keys were stolen is not known. After that, some **flaw in the key generation algorithm, a third party hack or a social engineering or phishing attack** is the most common way for hackers to obtain a private key, with 7.7% of the total for each one.

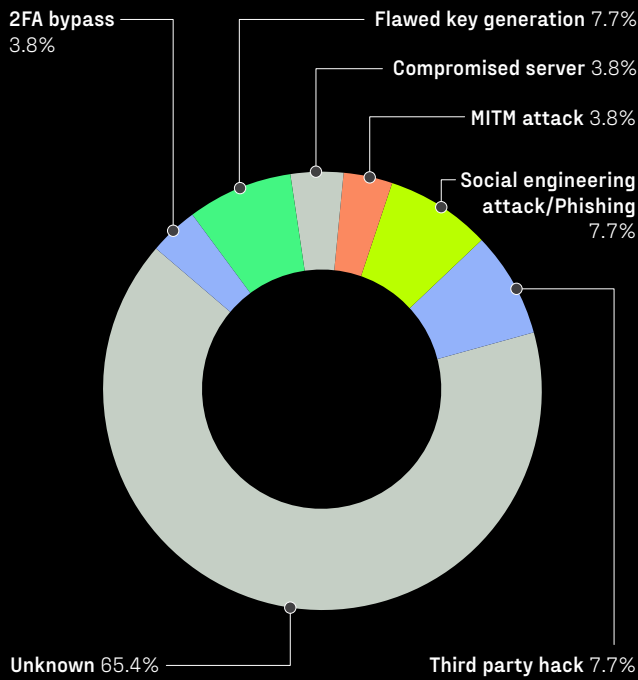


Figure 97: Number of reasons causing a compromised private key [percentage]

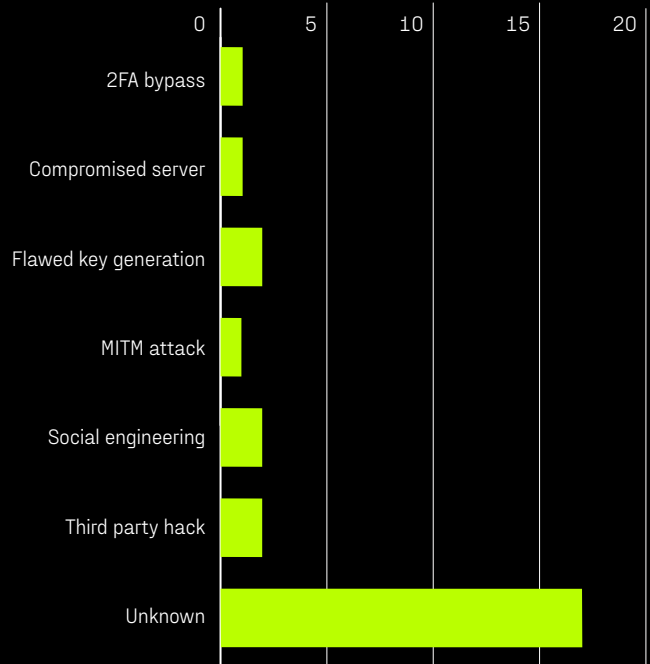


Figure 98: Number of reasons causing a compromised private key [count]

If we observe the distribution of the lost funds by root cause, **Figures 99 and 100**, **unknown** causes produce slightly lower losses (50.3%, \$1,202,188,951 USD) than predicted by their rates of occurrence but still take first place.

Social engineering, however, results in major losses (28.4%, \$679,000,000 USD) when compared with the number of times it has been the root cause. **Third-party hacks** also produce slightly higher losses, accounting for 9.9% of the total or around \$237,000,000 USD.

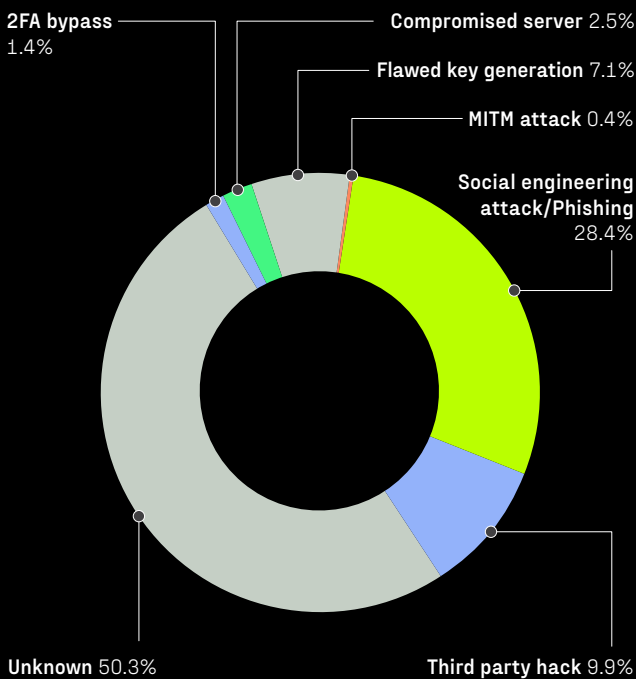


Figure 99: Loss caused by reasons leading to a compromised private key [percentage]

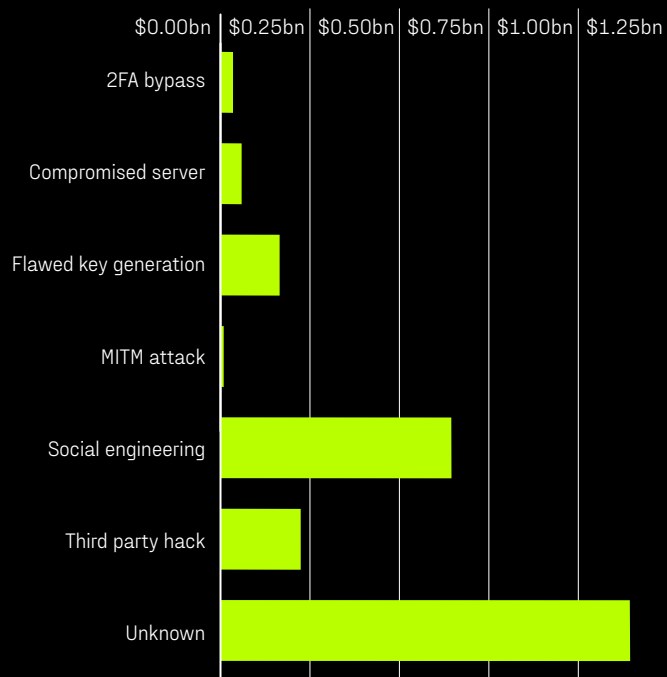


Figure 100: Loss caused by reasons leading to a compromised private key [USD]

Through the years, we can observe that, in the majority of the cases, how these keys were compromised is unknown.

However, in 2021 and 2022, this percentage decreases to make space for other causes previously mentioned. (Figures 101 and 102).

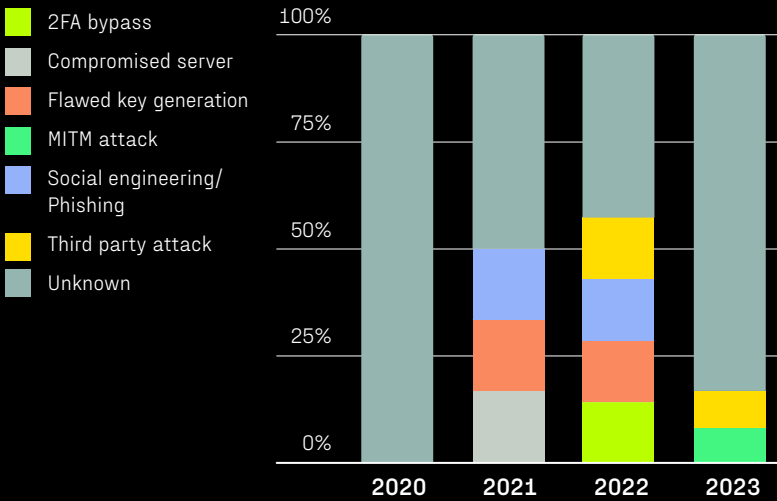


Figure 101: Number of reasons causing a compromised private key per year [percentage]

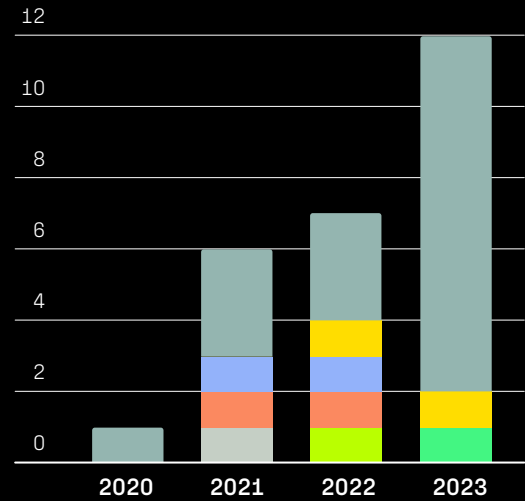


Figure 102: Number of reasons causing a compromised private key per year [count]

If we compare the distribution by loss and years in Figures 103 and 104, we can observe that, in 2021, **unknown** causes produce a higher ratio of stolen funds compared to their rates of occurrence (77.2%, \$413,700,000 USD). In 2023, however, it resulted in less loss, 74.4% (\$607,248,951 USD). In 2022, **social engineering** produced a much higher ratio of losses than their occurrence (62.8%, \$624,000,000 USD)

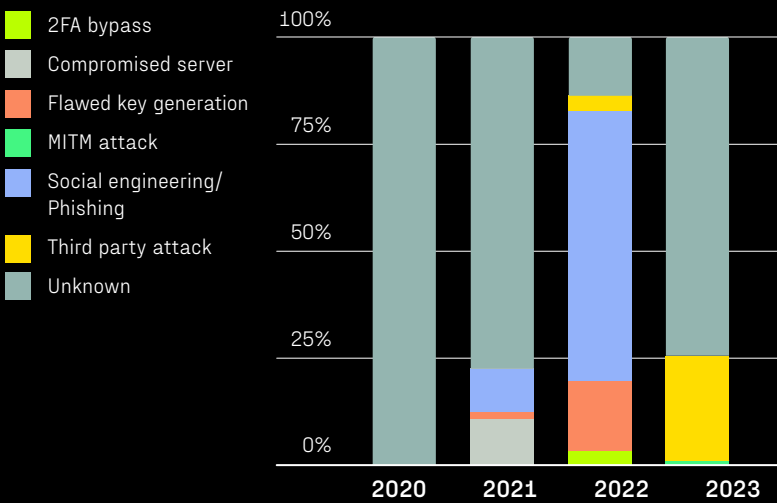


Figure 103: Loss caused by reasons leading to a compromised private key per year [percentage]

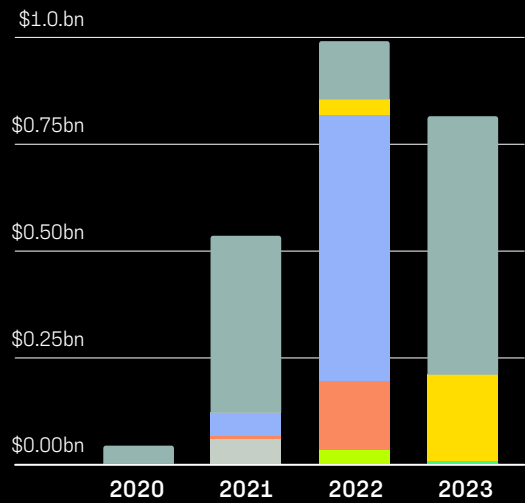


Figure 104: Loss caused by reasons leading to a compromised private key per year [USD]

Phishing

The main cause of the only attack resulting from a phishing scheme against users was a protocol's compromised API key, which allowed the attacker to inject a malicious script into custom routes.

This malicious script was triggered when users attempted to perform transactions on the project's web page. The script included additional unlimited spend approvals for the attacker's address within the user's transactions.

Traditional

In this case the cause was a compromised API key.

The API is related to the project's trading capabilities. As a result, an attacker with access to these API keys could also access the company's blockchain wallets and perform transactions on its behalf.

Attacks per chains

In order to see how different chains are being attacked, this report studies the relation between the different types of attacks defined previously and the chains previously examined.

We can observe in Figures 105 and 106, that the primary cause of attacks in Algorand, Bitcoin, Cronos, Tron, and Wemix is a **compromised private key**. It also accounts for half of the attacks on the Mixin and Polygon chain and is equal to **direct contract exploitation** on Ethereum. **Direct contract exploitation** is the primary cause of hacks in BSC, Solana, and Terra. **Price manipulation** attacks account for the majority of the hacks in Avalanche and Celo and for half of those on Arbitrum, Base, Mixin, and Optimism. **Rug pulls and scams** are the cause of all attacks on MoonRiver and DogeChain and half on Base. Fantom is a peculiar case because the attacks are divided equally between **compromised private keys, direct contract exploitation, price manipulation** attacks, and **rug pulls**.



Figure 105: Number of type of attack per chain [percentage]

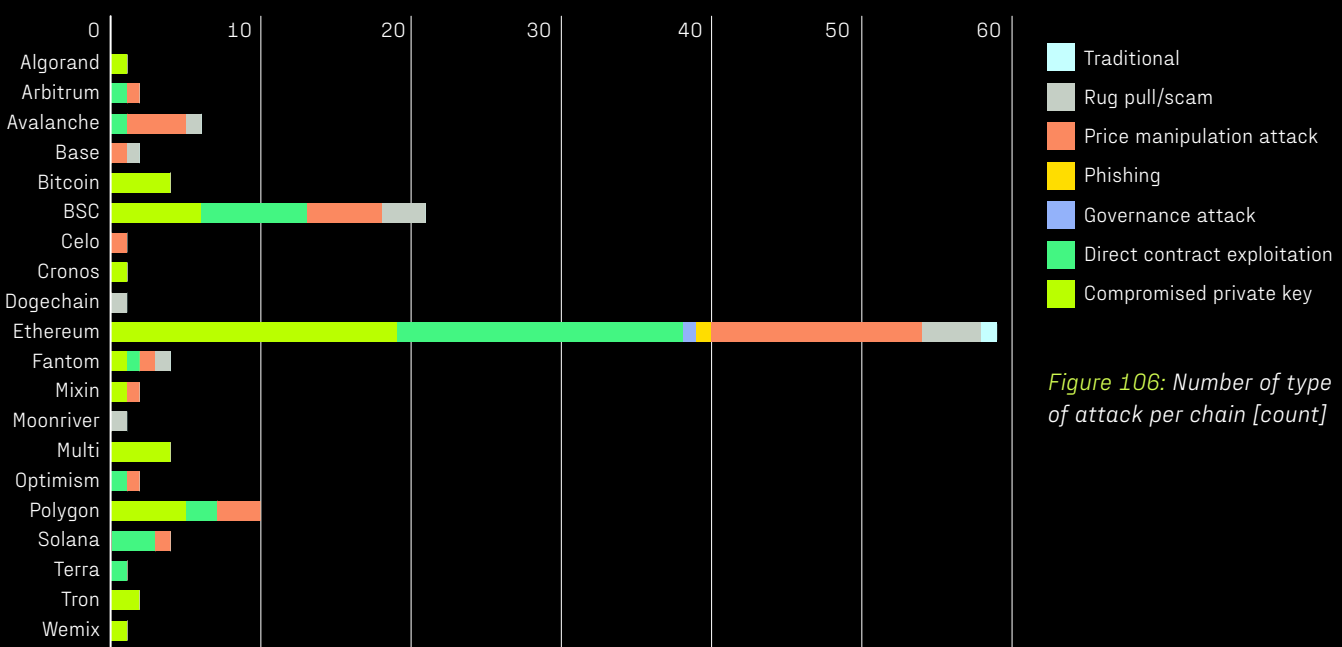


Figure 106: Number of type of attack per chain [count]

With regard to losses, we can observe that, in general, they follow a similar distribution to that of occurrence with some exceptions (Figures 107 and 108).

For example, in Arbitrum the amount lost because of **direct contract exploitation** (80%, \$80,000,000 USD) is higher than its rate of occurrence. Something similar occurs in BSC, with it accounting for 75% of the losses (\$1,020,000,000 USD) and Optimism (67%, \$30,500,000 USD). In Base, **rug pulls** led to 92% of the value lost (\$23,000,000 USD), and the same was true for Fantom (73.4%, \$120,000,000 USD).



Figure 107: Loss caused by type of attack per chain [percentage]

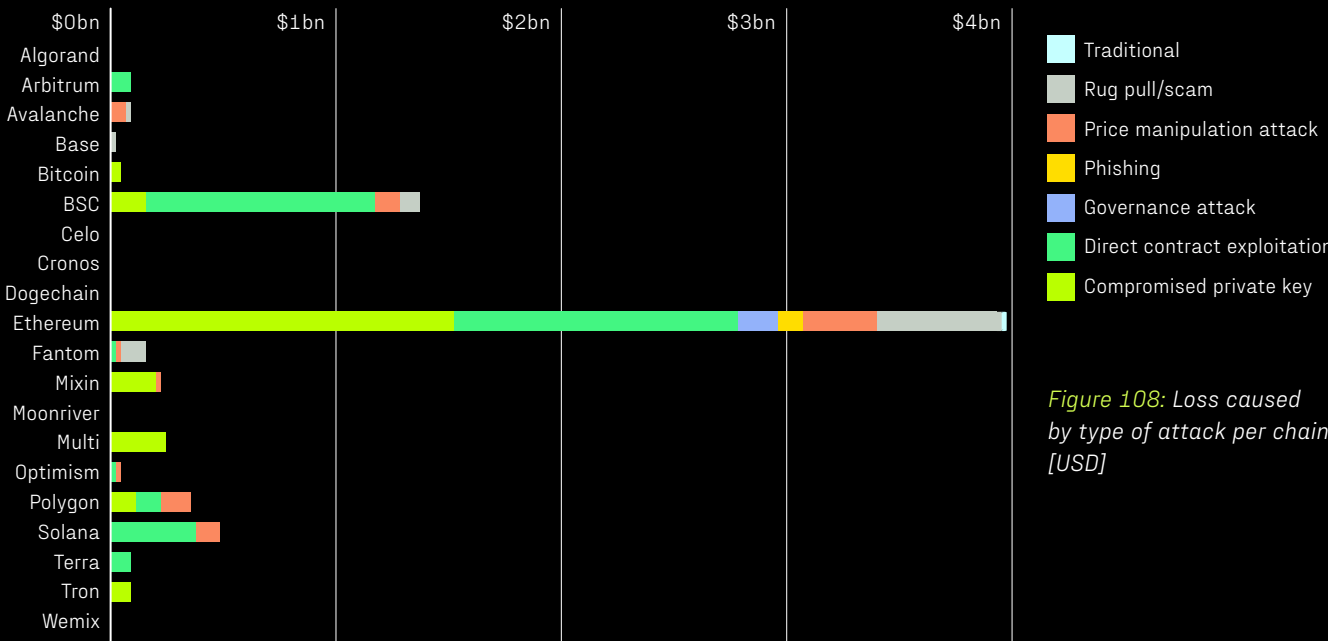


Figure 108: Loss caused by type of attack per chain [USD]

Through the years, it can be seen how some attacks are most prominent in some chains and how they evolve.

For example, attacks on Avalanche started mainly as **price manipulation** attacks and evolved to mostly **direct contract exploitation**. In BSC and Ethereum, a **compromised private key** seems to be an increasing threat. Polygon, however, seems to be the subject of more **price manipulation attacks** in recent years. (Figures 109 and 110)

■ Traditional
 ■ Rug pull/scam
 ■ Price manipulation attack
 ■ Phishing
 ■ Governance attack
 ■ Direct contract exploitation
 ■ Compromised private key

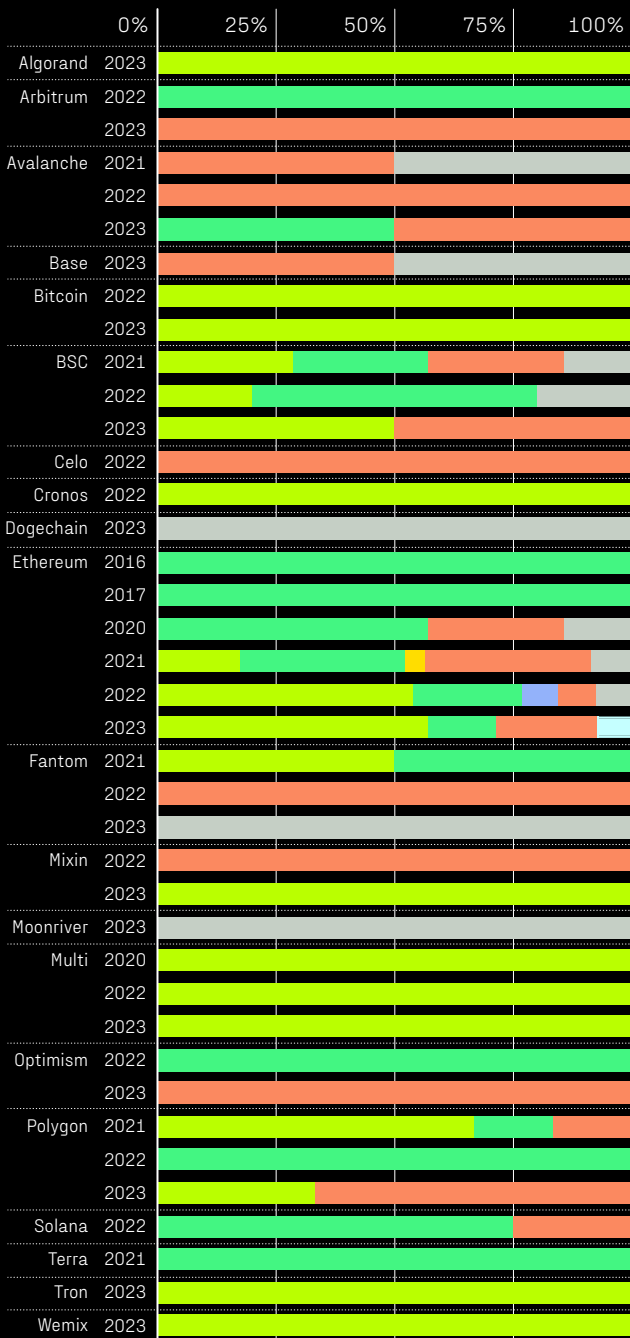


Figure 109: Number of type of attack per chain and year [percentage]

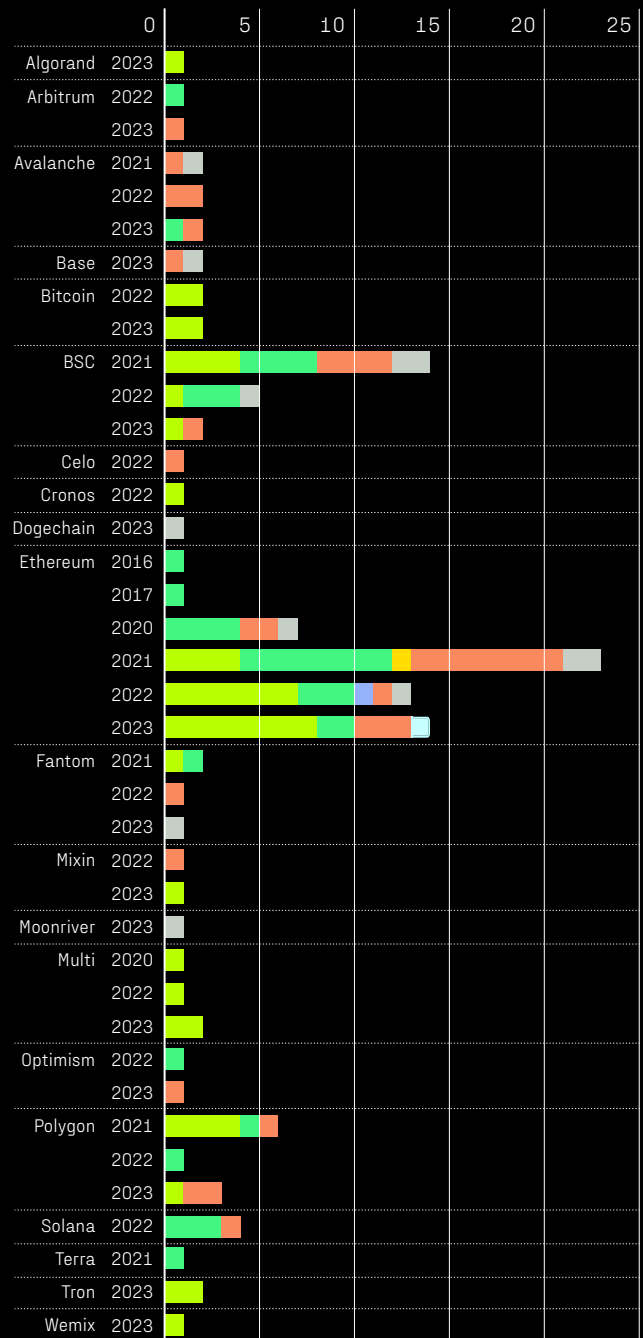


Figure 110: Number of type of attack per chain and year [count]

We can observe in Figures 111 and 112, that in general, losses follow a similar pattern to that observed in the previous charts (Figures 109 and 110).

However, there are some cases that are a bit different. For example, the value lost in Avalanche in 2021 due to price manipulation attacks is a bit higher than its rate of occurrence, 65.3% (\$34,000,000 USD) versus 50%. In BSC in 2022, direct contract exploitation accounted for 98.2% of the lost funds (\$682,000,000 USD) against a rate of occurrence of 60%. Finally, for Polygon in 2021, a compromised private key actually dealt more damage than predicted by its rate of occurrence, with 51.4% (\$110,539,954 USD) of losses and less in 2023, 5.9% (\$7,700,000 USD).

■ Traditional
 ■ Rug pull/scam
 ■ Price manipulation attack
 ■ Phishing
 ■ Governance attack
 ■ Direct contract exploitation
 ■ Compromised private key

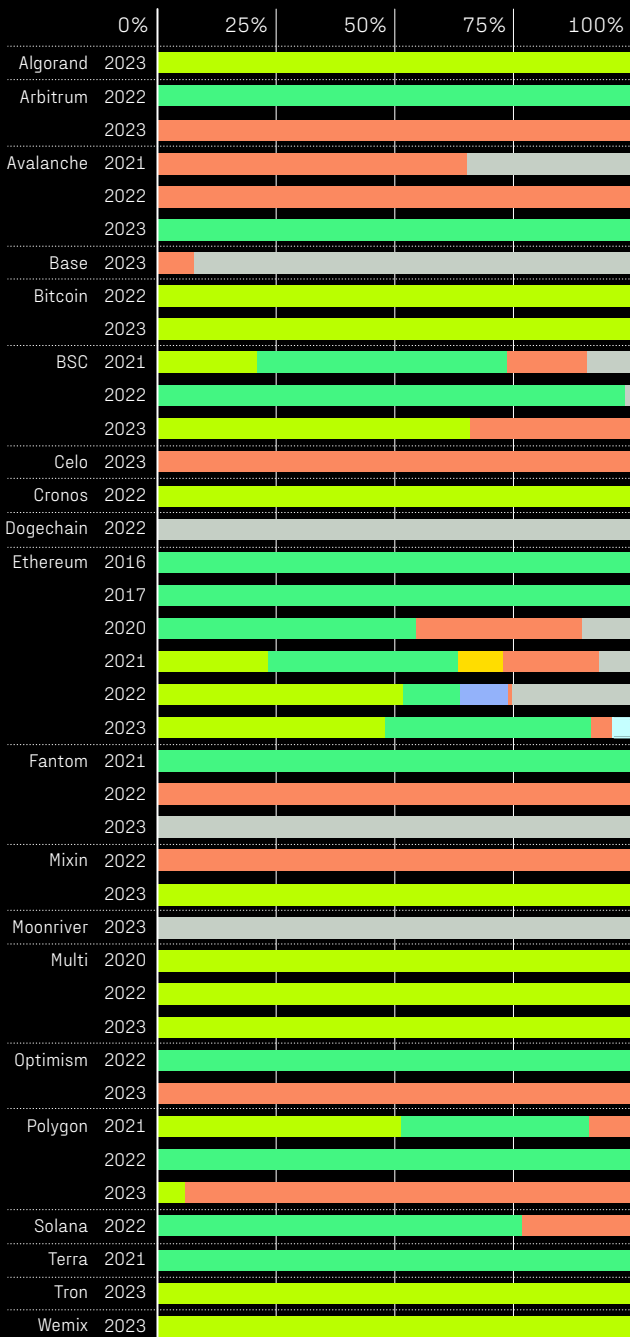


Figure 111: Loss caused by type of attack per chain and year [percentage]

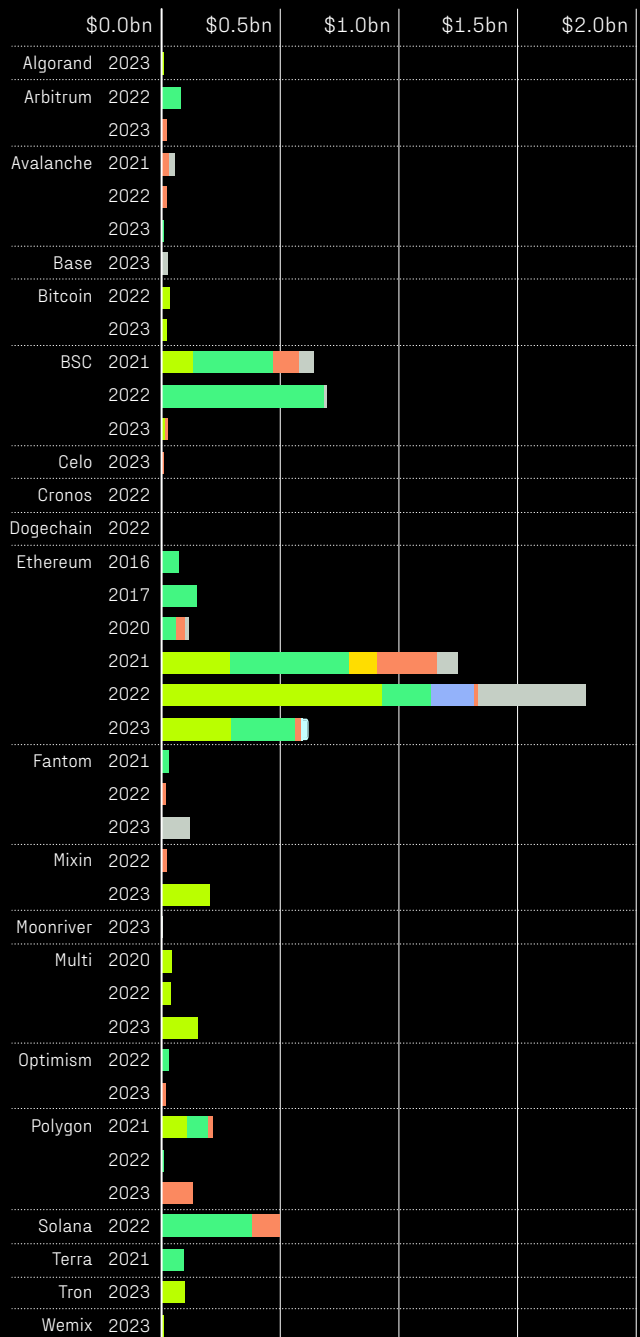


Figure 112: Loss caused by type of attack per chain and year [USD]

TYPE OF PROTOCOLS

In the DeFi space, there are a myriad of different protocols offering diverse services.

With the goal of knowing which protocol could be more vulnerable to attacks, we have analyzed the nature of each of the victims. For our classification, in order to preserve some coherence among different sources, we will follow DeFillama's protocol categories and nomenclature⁵, which are the following:

DEXes⁶ (Decentralized Exchanges)

Protocols enabling users to trade cryptocurrencies in a decentralized manner without the need for an intermediary.

Yield

Protocols that reward users for staking or providing liquidity to their platform.

Lending

Protocols that facilitate the borrowing and lending of digital assets between users.

Derivatives

Protocols that allow users to engage in contracts whose value is derived from the performance of underlying assets, such as futures and options, without directly owning the asset.

Services

Protocols offering various services to users.

Liquid Staking

Protocols that allow users to earn staking rewards without locking up their tokens, providing a liquid certificate representing the staked position that can be traded

Yield Aggregator

Platforms that optimize yield farming strategies across various protocols to maximize returns for users.

Reserve Currency

Protocols modeled after OHM that back their native tokens with a reserve of valuable assets.

CDP (Collateralized Debt Positions)

Protocols that issue stablecoins backed by user-provided collateral through a lending process.

Algo-Stables

Protocols that issue stablecoins with values stabilized through algorithmic mechanisms.

Farm

Protocols that offer users protocol tokens in return for locking up funds, often to provide liquidity.

Bridge

Protocols designed to transfer tokens between different blockchain networks, enhancing interoperability.

Indexes

Protocols that track or create a performance metric for a collection of related digital assets.

Options

Protocols that offer contracts providing the right to purchase an asset at a predetermined price

⁵ Downloaded from <https://defillama.com/docs/api>

⁶ We will use the term 'DEXes' throughout this report instead of 'Dexes', which is DeFiLlama's nomenclature, since this term is more commonly used.

Launchpad

Protocols designed to support the launch of new projects and tokens, often providing early access to participants.

SoFi (Social Finance)

Networks that combine finance with social elements, fostering community and financial interactions.

Gaming

Protocols incorporating gaming mechanics.

Synthetics

Protocols that create digital assets whose value is derived from another asset, mimicking its price movements.

Prediction Market

Protocols where users can speculate on the outcomes of future events, with rewards for accurate predictions.

CEX (Centralized Exchanges)

Protocols where users can buy, sell, and trade cryptocurrencies with the facilitation of a central authority, offering high liquidity and fiat currency support.

NFT Marketplace

Protocols where users can trade NFTs, including buying, selling, and renting.

Chain

Refers to blockchain networks and platforms that support the creation and execution of decentralized applications, smart contracts, and transactions.

Liquidity manager

Protocols that specialize in optimizing liquidity positions within AMMs that offer concentrated liquidity.

NFT Lending

Protocols that allow users to take out loans using non-fungible tokens (NFTs) as collateral.

Cross Chain

Protocols that facilitate interoperability and communication between different blockchain networks.

Insurance

Protocols offering financial protection against specific risks in the digital asset space.

Staking Pool

Platforms where users can stake their assets to secure a network, earning rewards without receiving a tradable receipt token to use in other Defi apps like with Liquid Staking projects.

Leveraged Farming

Protocols that allow users to employ leverage (borrowed funds) to increase their position in yield farming activities, potentially amplifying returns.

Payments

Protocols that enable users to conduct cryptocurrency transactions, including payments, receipts, and transfers.

DEX Aggregator

Services that pull liquidity from multiple decentralized exchanges to ensure users receive the best possible trading terms, such as price and slippage.

Privacy

Protocols focused on concealing transaction details to enhance user privacy.

Options Vault

Protocols offering automated options strategies, where users can deposit their assets to gain exposure to options trading with the aim of earning higher yields.

Uncollateralized Lending

Protocols that offer loans without requiring collateral from the borrower, relying instead on credit assessments or community trust.

Oracles

Protocols that provide a bridge for external (off-chain) data to interact with smart contracts on the blockchain.

Decentralized Stablecoin

Digital currencies designed to maintain a stable value, typically pegged to a fiat currency or basket of assets, without centralized control.

RWA Lending

Platforms that facilitate the lending and borrowing of funds using real-world assets as collateral, integrating traditional asset classes into the blockchain ecosystem.

Liquid Restaking

Initiatives where the liquid staking token (LST) inherently acts as a Liquid Restaking Token (LRT), fully supporting restaking activities.

Wallets

Digital applications or devices that store public and private keys for cryptocurrencies, enabling users to send, receive, and manage their digital assets securely.

Restaking

Platforms that allow users to stake their assets in multiple protocols simultaneously, maximizing their staking rewards and participation in network security.

We have added the category **Other Currency**, to include memecoins and shitcoins as they appear in our studied sample but do not fit a category in DeFillama's classification.

In **Figures 113** and **114**, we can see that the majority of protocols that have been attacked were **Lending protocols**, which account for 14% of the total.

The next most hacked protocols have been **DEXes**, with 13%. **Bridges**, **CEXs**, and **Yield Aggregators** share the third position with 12%.

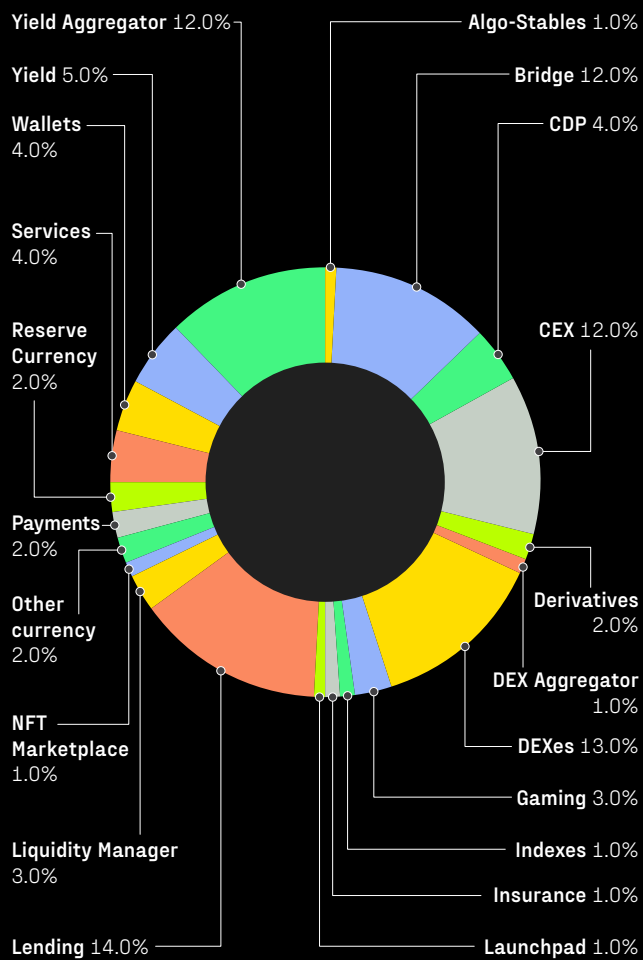


Figure 113: Number of type of protocol [percentage]

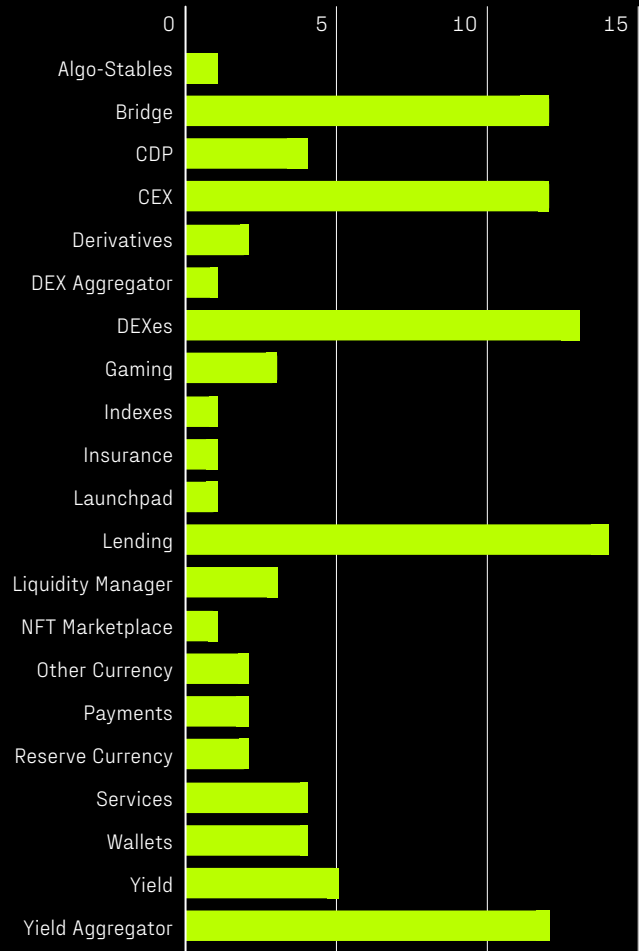


Figure 114: Number of type of protocol [count]

By loss, however, we can see in **Figures 115 and 116** that **Bridges** are the protocol that occupies first place, representing **38.6% of the total loss (\$2,841,002,138.00 USD)**.

Next, **CEXs** are the protocol with the second highest loss, 14.5% (\$1,064,609,000.00 USD). The third one is **Lending** protocols, with 13% of the total funds stolen (\$953,000,000.00 USD).

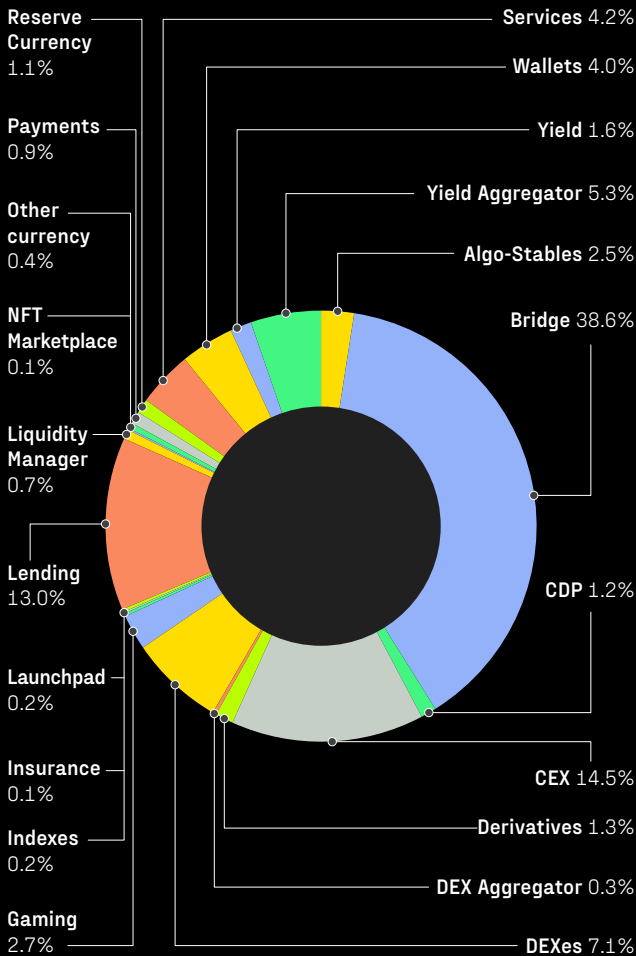


Figure 115: Loss caused by type of protocol [percentage]

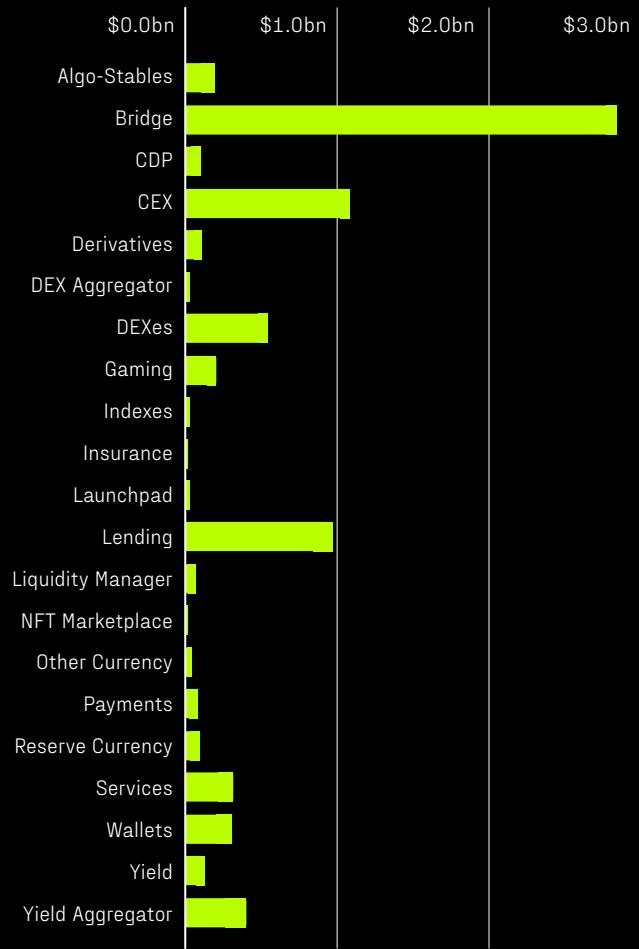


Figure 116: Loss caused by type of protocol [USD]

Comparing the percentage of type of protocol attacked versus the approximate percentage of the type of protocol in the sample retrieved from DeFillama (Figure 117), we can see that, while DEXes are the most popular protocols by quantity, they are comparatively one of the least hacked ones in comparison.

Something similar happens with Yield protocols. Bridges, DEX Aggregators, and CEXs are fewer in number in the DeFi space but have been the targets of a larger percentage of attacks in comparison. Thus, it seems these protocols are more vulnerable to attacks or more desirable for hackers. Lending protocols and Gaming seem to be also somehow vulnerable but to a lesser extent.

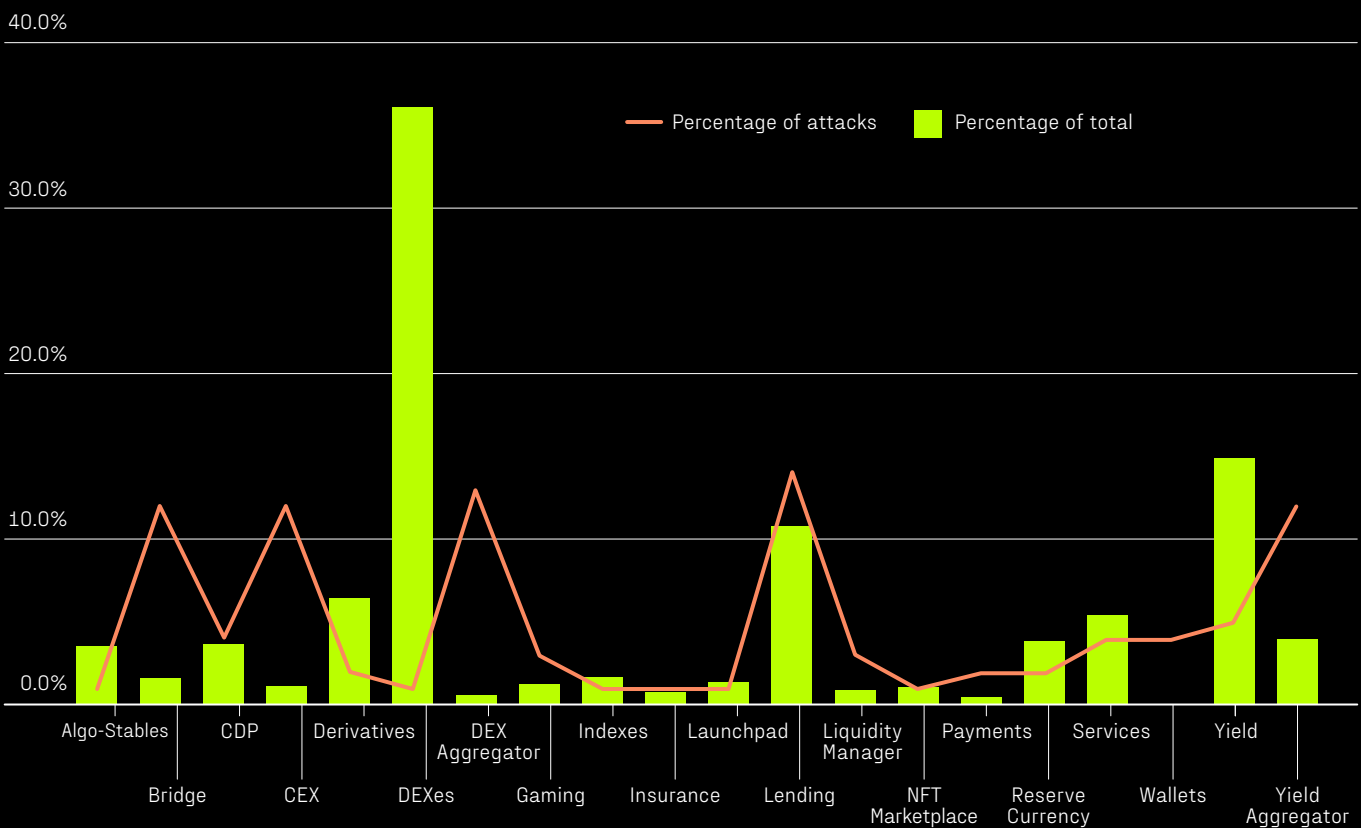


Figure 117: Percentage of type of protocol attacked versus percentage of type of protocol in sample

If we observe which types of protocols have been hacked more by year (Figures 118 and 119), we can observe that the protocol attacked in 2016 was a **Service**, while the one in 2017 was a **Wallet**.

In 2020, the most attacked protocols were **Yield Aggregators**, with 50% of the total. In 2021, however, **DEXes** took the first place with 18.4% of the total. In 2022, **Bridges** were the most common with 20.7% of the total. Finally, in 2023, **CEXs** have been the main target, accounting for 21.7% of the total.

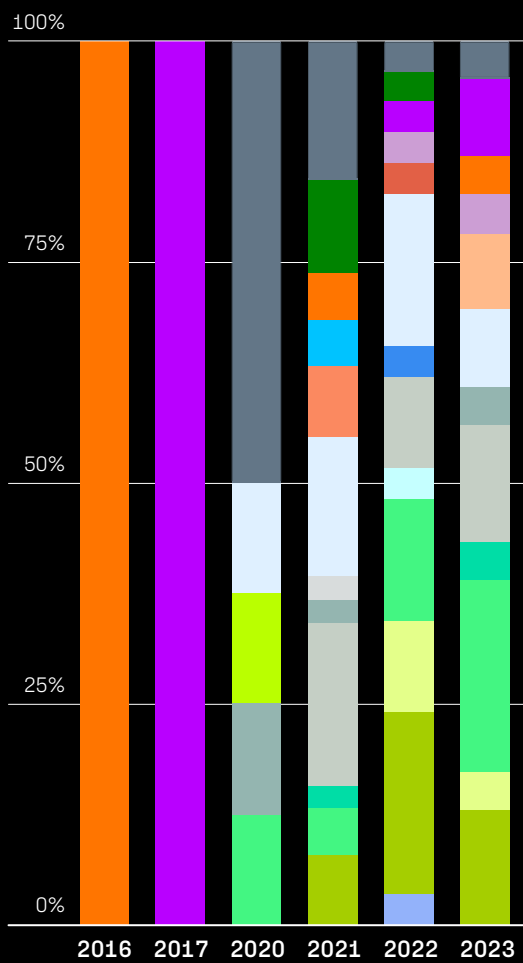


Figure 118: Number of type of protocol per year [percentage]

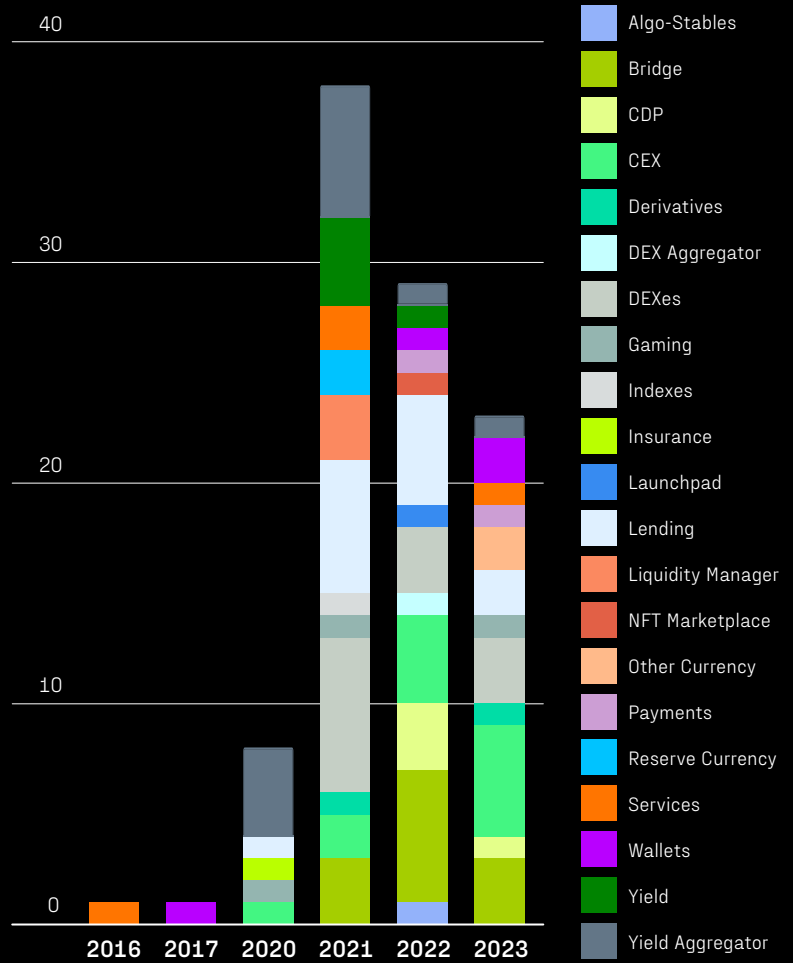


Figure 119: Number of type of protocol per year [count]

We can also check the distribution of the losses by year in Figures 120 and 121.

We are able to observe that **Bridges**, in general, seem to be the main cause of losses in 2021 and 2022. They are also the second most common cause in 2023, despite their low occurrence, accounting for 27.6% of the lost value in 2021 (\$626,936,138 USD), 58.9% in 2022 (\$1,906,000,000 USD) and 21% in 2023 (\$308,066,000 USD). The remaining types of protocols seem to follow in general a similar loss/occurrence distribution.

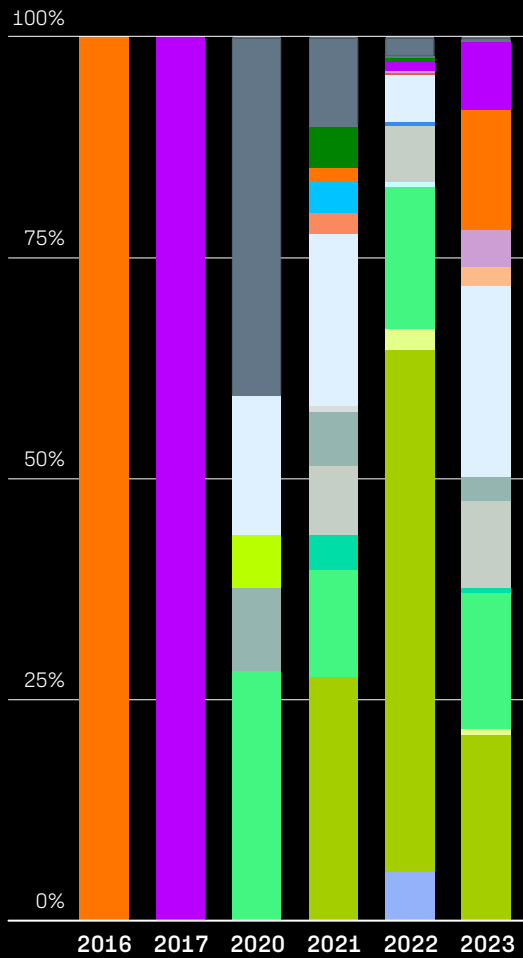


Figure 120: Loss caused by type of protocol per year [percentage]

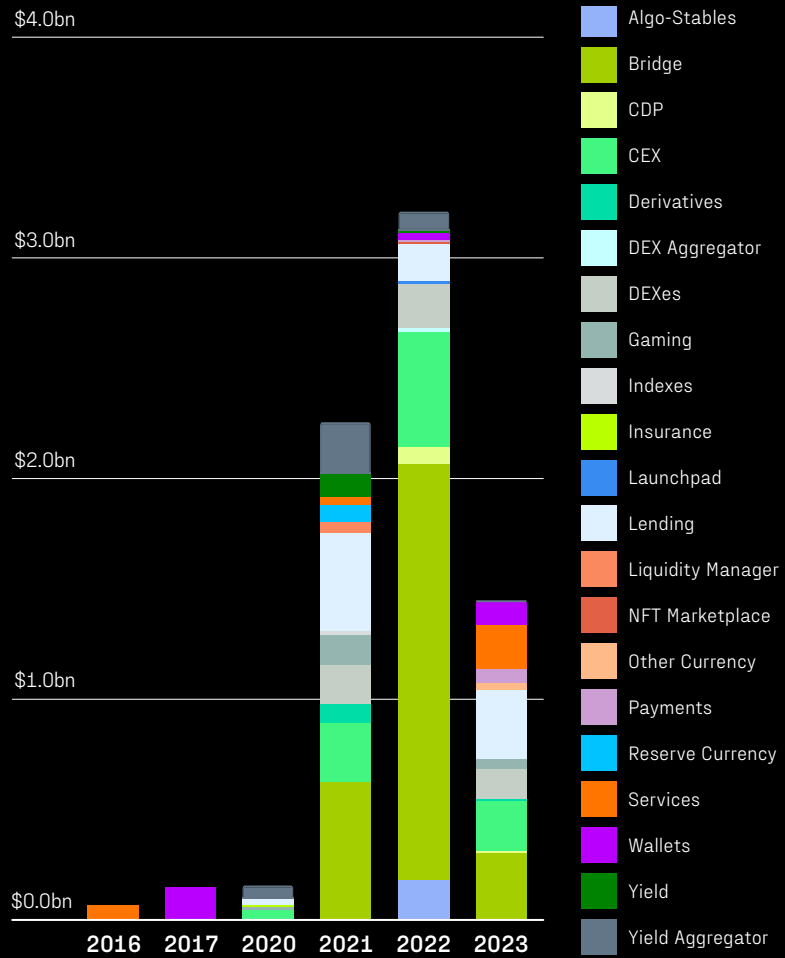


Figure 121: Loss caused by type of protocol per year [USD]

Governance

Are centralized organizations more vulnerable than decentralized ones?

We have analyzed the type of governance for each attacked protocol in order to answer that question. Our research shows that, while centralized organizations have been attacked more, the difference between them and decentralized ones is really small, at 12% (Figures 122 and 123).

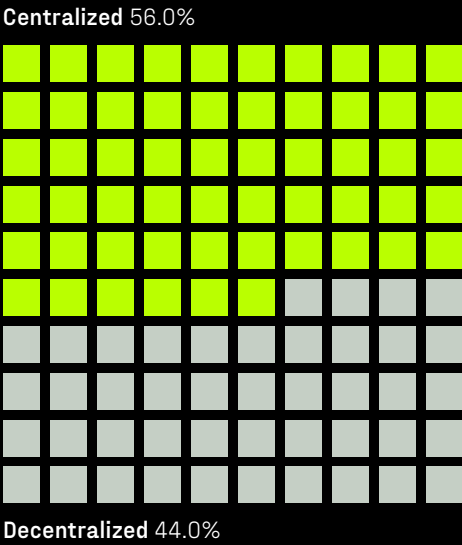


Figure 122: Usage of type of governance [percentage]

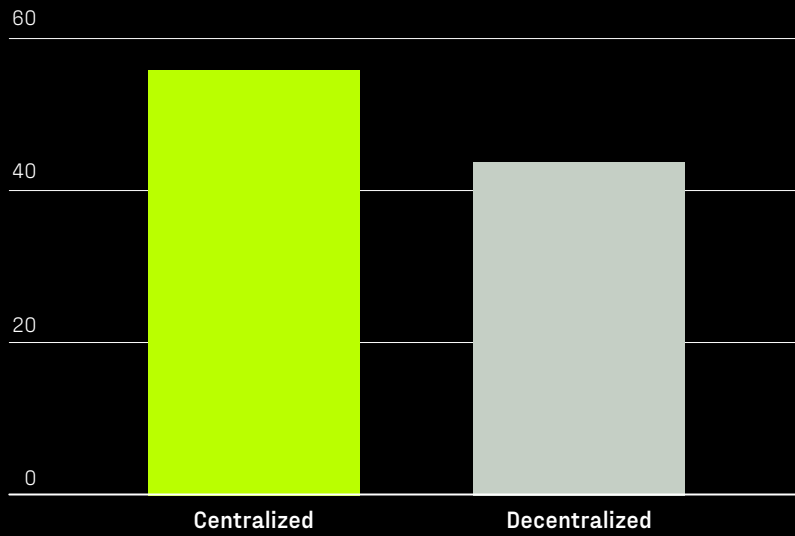


Figure 123: Usage of type of governance [count]

However, if we observe the losses for each type of governance, we can see that decentralized protocols only represent 28.2% of the total losses (\$2,072,475,138.00 USD), while centralized ones represent the rest (\$5,279,588,951.00 USD) (Figures 124 and 125). This suggests that attacking centralized protocols is more profitable for the hackers than targeting decentralized ones.

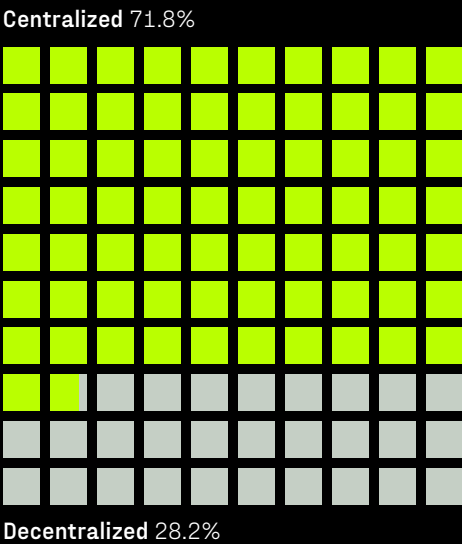


Figure 124: Loss per type of governance [percentage]

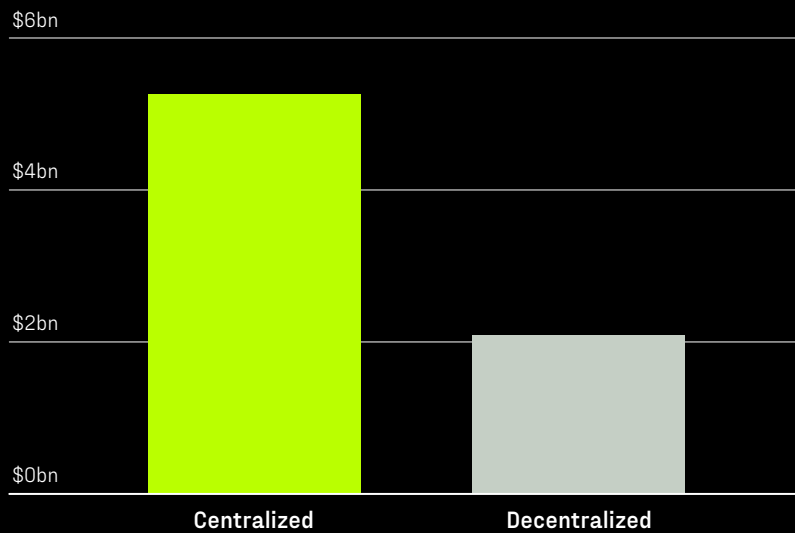


Figure 125: Loss per type of governance [USD]

By year, attacks on decentralized protocols seem to have lessened slightly, from 37.5% in 2020 to 30.4% in 2023 (Figures 126 and 127).

However, they increased in 2021, up to 57.9%.

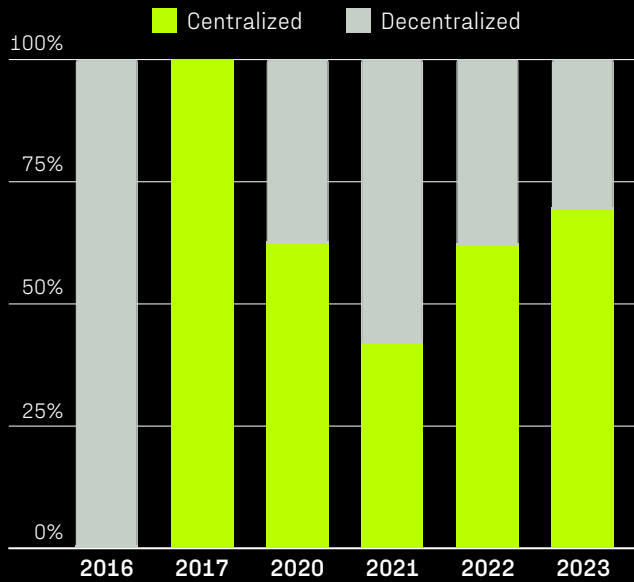


Figure 126: Usage of type of governance per year [percentage]

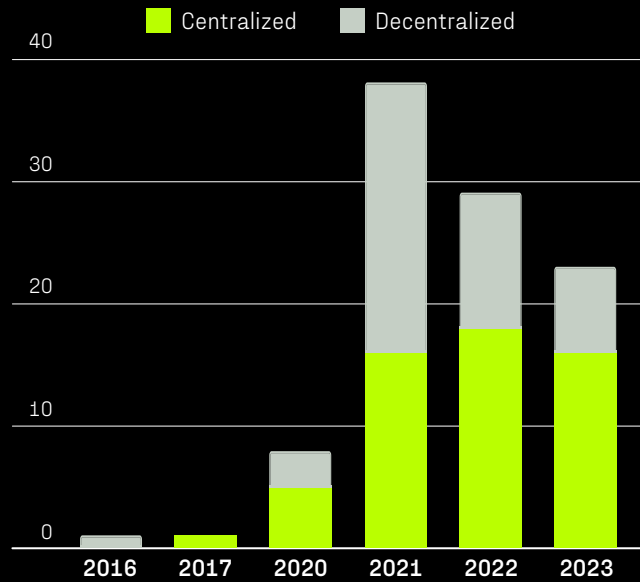


Figure 127: Usage of type of governance per year [count]

Regarding losses, we can observe in Figures 128 and 129, that the opposite occurs. There is less loss by decentralized protocols in 2021 (35.3%, \$802,386,138.00 USD) and in 2022 (15.1%, \$487,900,000 USD) and more in 2023 (44.7%, \$654,189,000 USD). Still, the percentages are fairly similar.

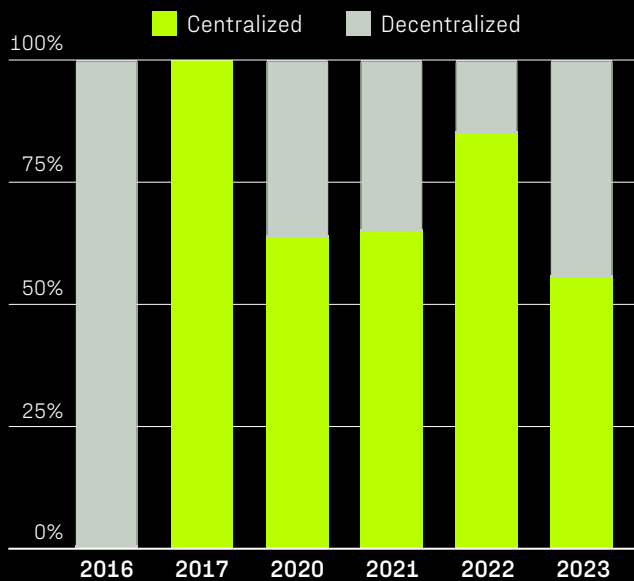


Figure 128: Loss per type of governance per year [percentage]

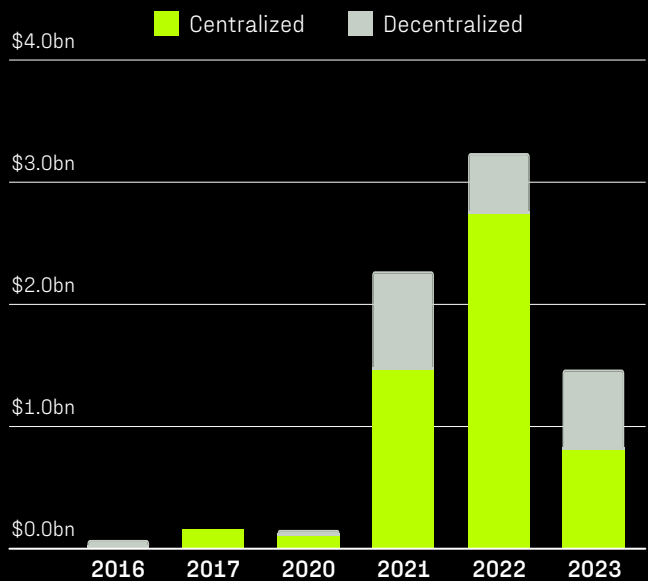


Figure 129: Loss per type of governance per year [USD]

Type of Protocols per Chains

Figures 130 and 131 show which types of protocols have been attacked more per chain.

For Algorand, it has been a **Wallet**. Arbitrum is divided between **DEXes** and **Yield Aggregators** equally. Avalanche’s most targeted type of protocol has been **Lending**. Base is equally divided between **Other currency** and **DEXes**. On Bitcoin, the attacked protocols have been **CEXs**. On BSC, **DEXes** and **Bridges** are the two most common ones. In Celo, it is **Lending** protocols. On Cronos, it was an **NFT Marketplace**. Ethereum has more variety, **CEXs** lead with 16.9% of the total, followed by **Yield Aggregators** with 15.3%. The attacked protocol in Dogechain was a **Bridge**. In Fantom, there were mostly **Bridges** attacked. Mixin is divided between **Services** and **Lending** protocols. Moonriver’s attacked protocol was a **Bridge**. Optimism’s hacked protocols were **DEXes**. In Polygon, the most attacked protocols were **Lending** protocols with 30% of the total, followed by **Gaming** and **DEXes** (20%). Solana is divided equally between **Lending**, **DEXes**, **CDP**, and **Bridge**. For Terra it was **Derivatives**. On Tron, they were **CEXs** and **Payments**. Finally in Wemix, it was a **CEX**.

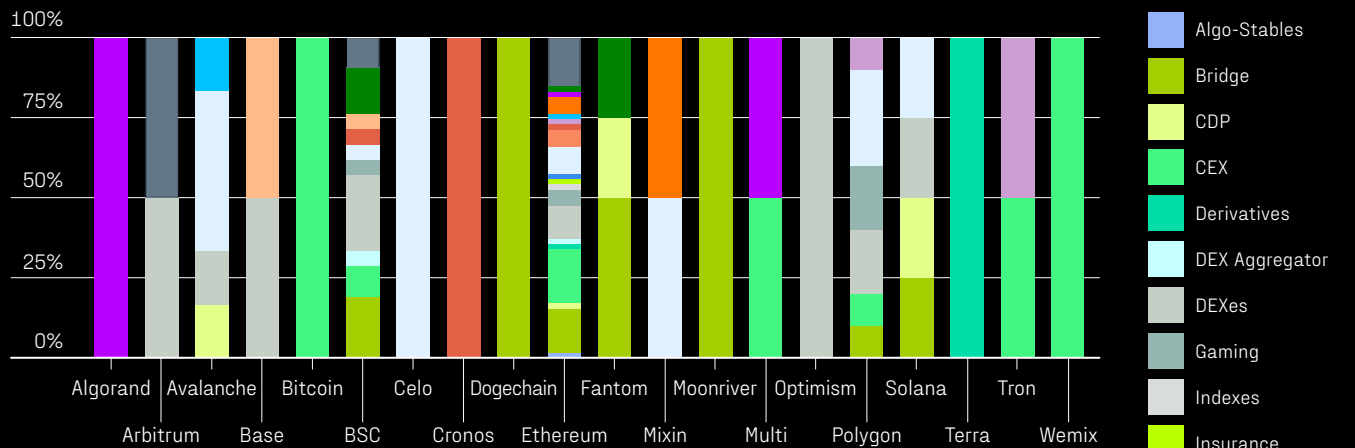


Figure 130: Number of type of protocols per chain [percentage]

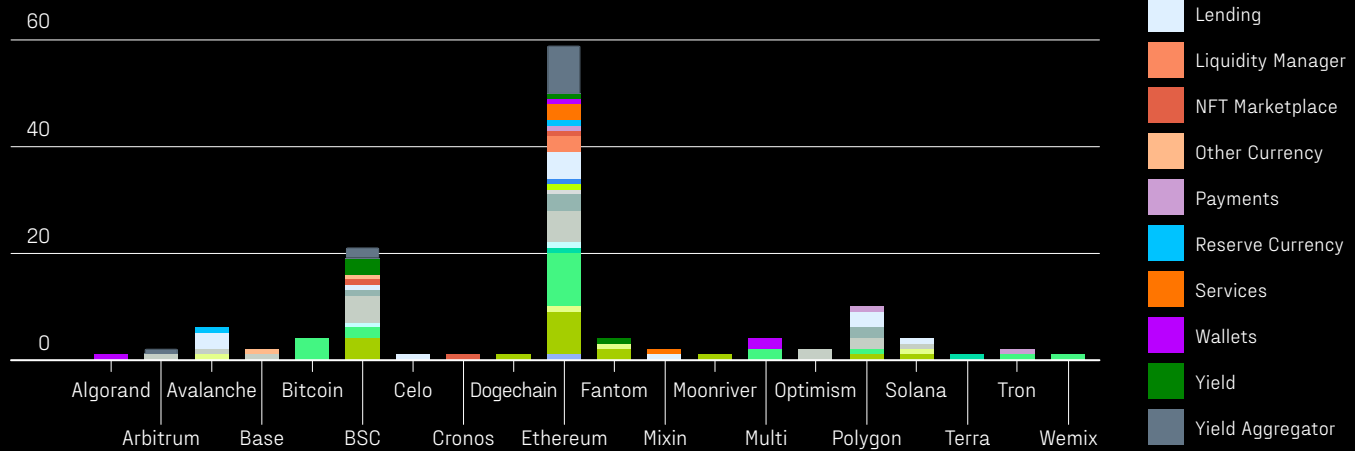


Figure 131: Number of type of protocols per chain [count]

We could also review the distribution of the protocols attacked by loss to see the differences (Figures 132 and 133).

Arbitrum's main cause of losses was attacks on **Yield Aggregator** protocols with 80% of the total (\$80,000,000 USD) versus a 50% of the total attacks. Avalanche's most affected type of protocol has been **Lending** ones (67.7%, \$55,800,000 USD), close to the 50% of the rate of occurrence. On Bitcoin, it has been **CEXs**. Base's main cause of loss was **Other currency** with 92% of the total losses (\$23,000,000 USD) versus 50% of the rate of occurrence. On BSC, **Bridges** have been the ones to deal more damage (67.6%, \$919,749,033 USD) representing only 19% of the total attacks. On Algorand, Celo, Cronos and Dogechain, the protocols attacked also represent the totality of losses (See Figures 130 and 131). Ethereum lost the most due to **Bridge** exploits (34.7%, \$1,382,674,465 USD), which only represent 13.6% of the attacks. For Fantom, attacks mostly targeted **Bridges** (73.5%, \$120,112,640 USD), which is higher than their rate of occurrence: 50%. Mixin's targets were **Services**, accounting for 90.1% of losses (around \$200,000,000 USD), which in contrast only represents 50% of the attacks for the chain. Moonriver and Optimism's losses are due to the same protocols as in Figures 130 and 131. For Polygon, **Lending** protocols produced the most losses, 58.3% (\$206,500,000 USD), while they only accounted for 30% of the attacks. Solana's main cause was also **Bridges** (65.5%, \$326,000,000.00 USD) versus 25% of the attacks for this kind of protocol. On Terra it was **Derivatives**. On Tron, they were **CEXs** and **Payments**, almost equally (49.6% and 50.4%), and very similar to their distribution by occurrence. Finally in Wemix, it was a **CEX**. In general, we can observe that **Lending** protocols and **Bridges** seem to lead to higher losses than their rate of occurrence. This case is especially significant in **Bridges**, for which, in those chains with more than one type of the protocol hacked, they are not the single main one by number of attacks but they are by loss produced.

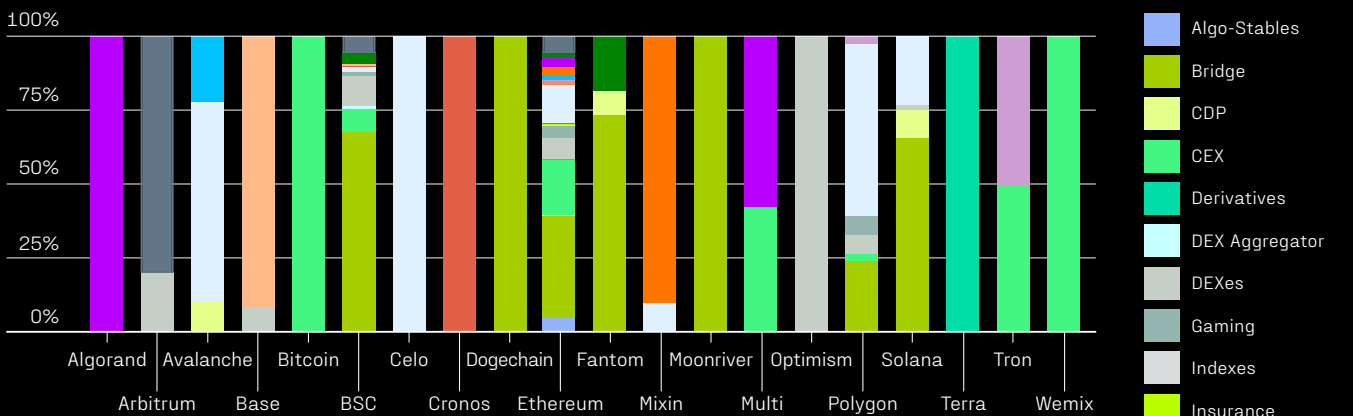


Figure 132: Loss per type of protocols per chain [percentage]

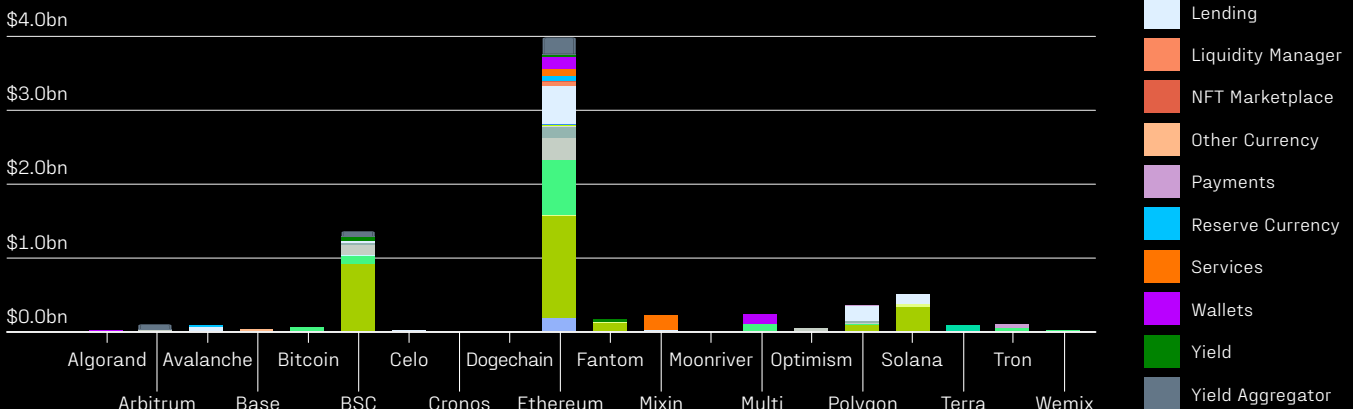


Figure 133: Loss per type of protocols per chain [USD]

If we check the types of protocols attacked by chain and year (Figures 134 and 135), there is not much of a trend in general.

In Ethereum, we can observe that, in later years, there has been an increase in attacks on **Bridges, CEXs** and **DEXes** protocols, while Polygon seems to favor **Gaming, Lending** and **DEXes** in 2021 and 2023.

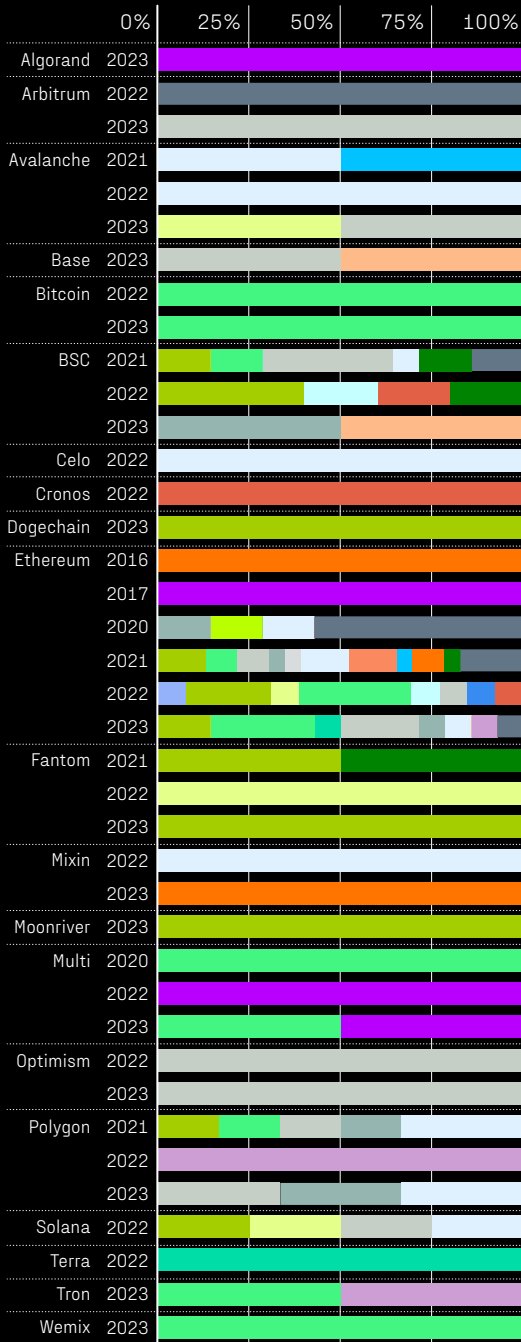


Figure 134: Number of type of protocols per chain and year [percentage]

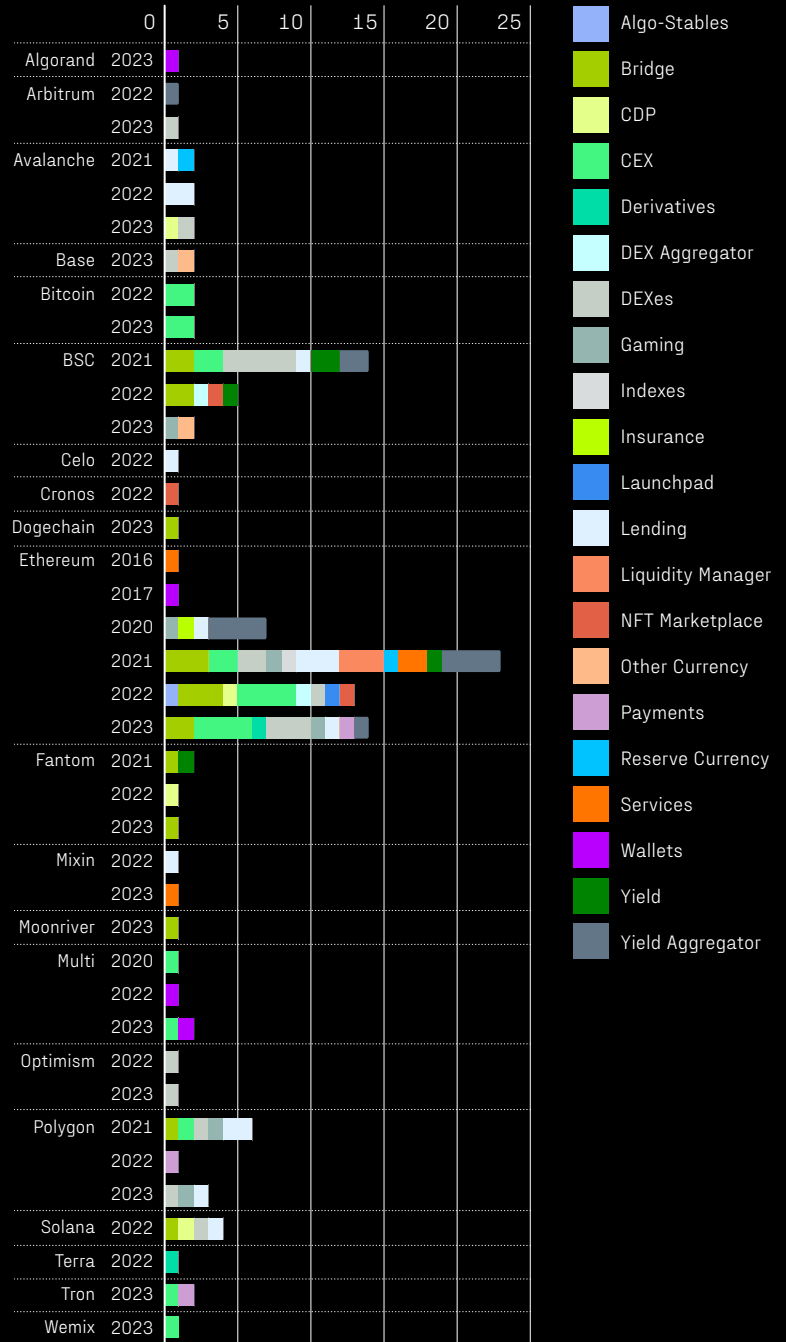


Figure 135: Number of type of protocols per chain and year [count]

Loss distribution by year and chain doesn't seem to show any pattern either.

We can observe in Figures 136 and 137 how the loss on BSC due to **Bridges** is higher than its rate of occurrence in both 2021 and 2022, accounting for 39.7% (\$253,749,033 USD) and 95.9% (\$666,000,000 USD) of the total loss respectively. Polygon in 2021 also seems to suffer a huge loss because of this type of protocol relative to its rate of occurrence: 39.5% (\$85,000,000.00 USD). In general, the rest of the chains and protocols seem to follow what was observed previously.

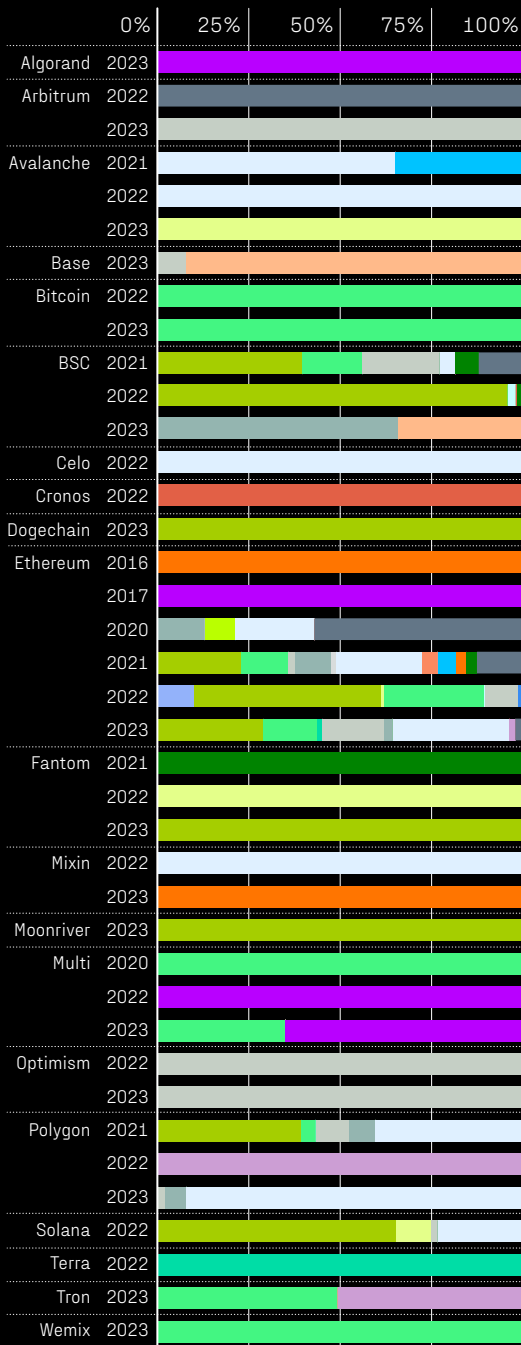


Figure 136: Loss per type of protocols per chain and year [percentage]

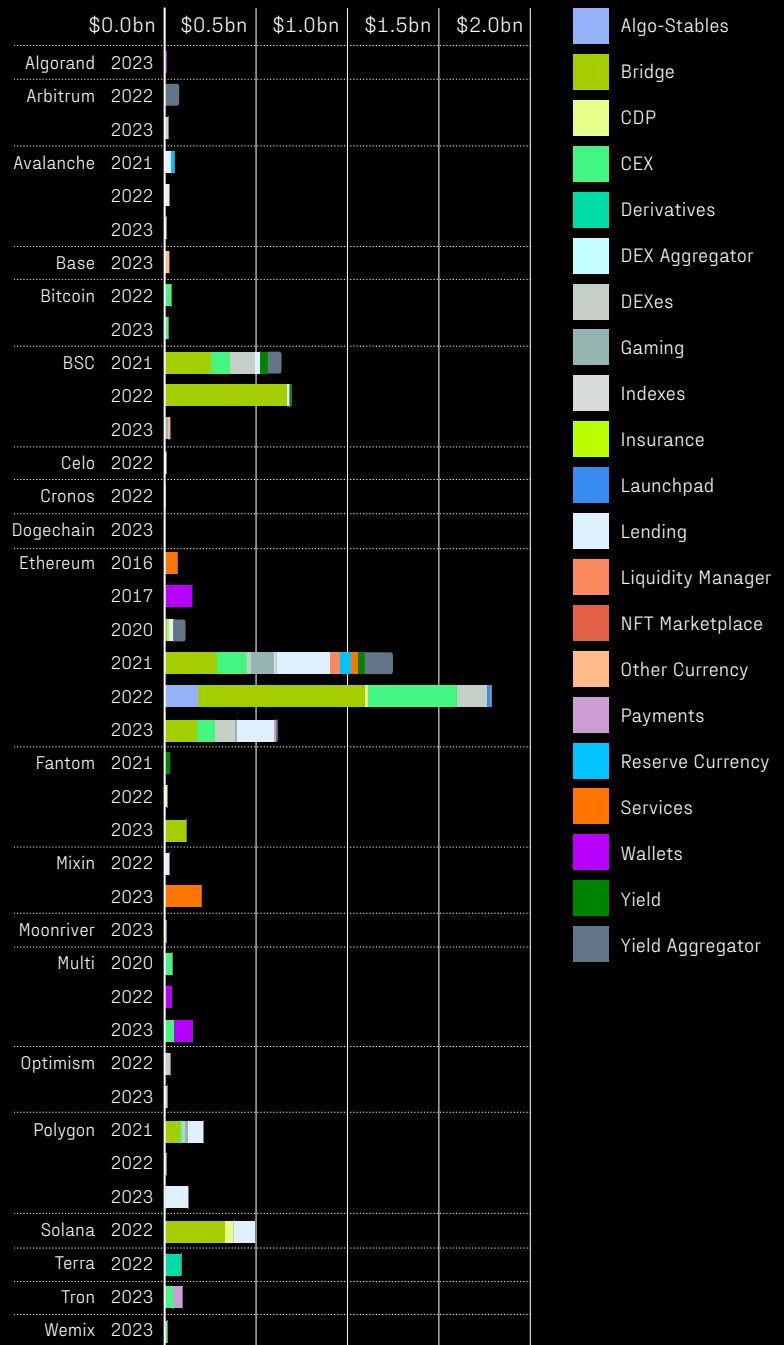


Figure 137: Loss per type of protocols per chain and year [USD]

Type of protocols per type of attacks

Figures 138 and 139 show the types of attacks versus protocols.

Direct contract exploitation is the primary cause of hacks in **Bridges, Dex Aggregators, Insurance, Launchpad, Liquidity Manager** and **Services** and shares the same percentage as **price manipulation** attacks for **Derivatives, DEXes, and Yield**. This latter type of attack is the most common in **Indexes, Lending, and Yield Aggregator**. A **compromised private key** seems to be the main cause of hacks in **CEXs, Gaming, NFT Marketplace** and **Wallet**, while sharing the top spot in **Payments** alongside **direct contract exploitation**. **Rug pulls/scams** seem to be more common in **Other currency** and **Reserve Currencies**.



Figure 138: Number of type of attacks per type of protocols [percentage]



Figure 139: Number of type of attacks per type of protocols [count]

If we check the distribution by loss (Figure 140 and 141), we observe that in **Bridges, CDPs, Derivatives, Lending protocols and Wallets, direct contract exploitation** caused more loss relative to its rate of occurrence.

For **Gaming** and **Services**, the same happens with **compromised private keys**. Rug pulls or scams caused disproportionately large losses in **Other Currency** and **CEXs**.

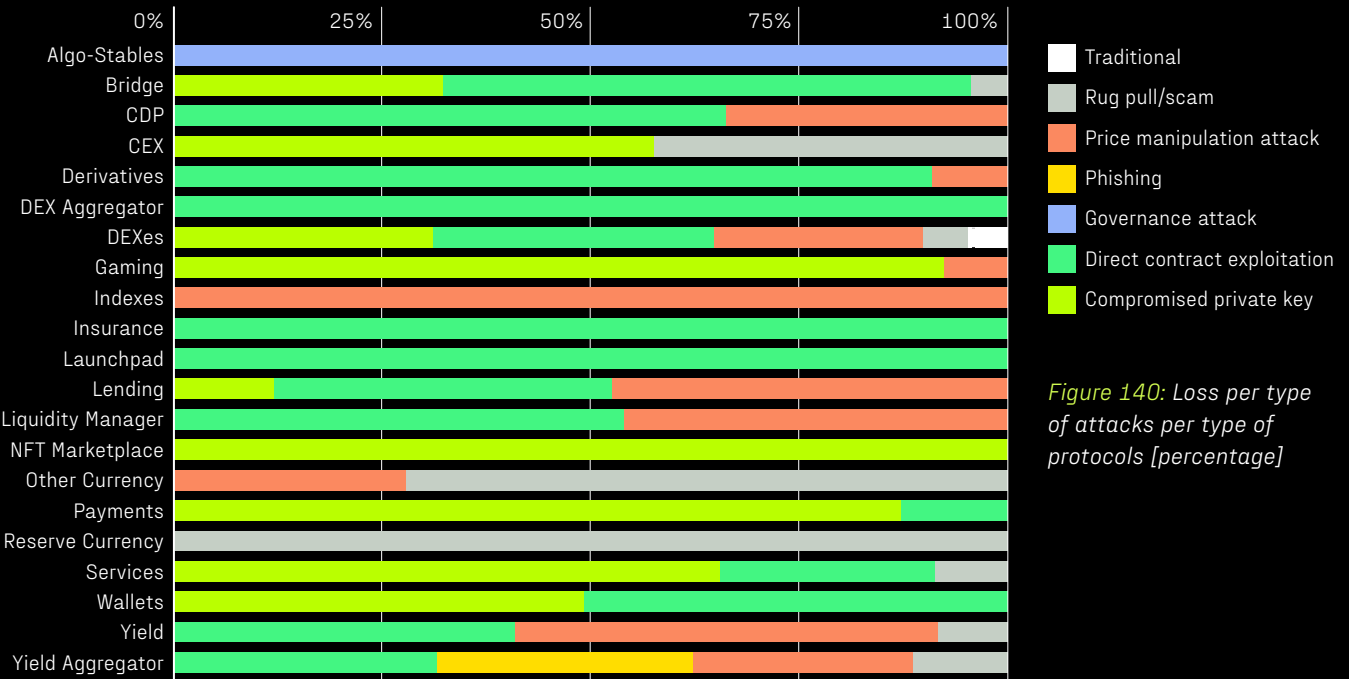


Figure 140: Loss per type of attacks per type of protocols [percentage]

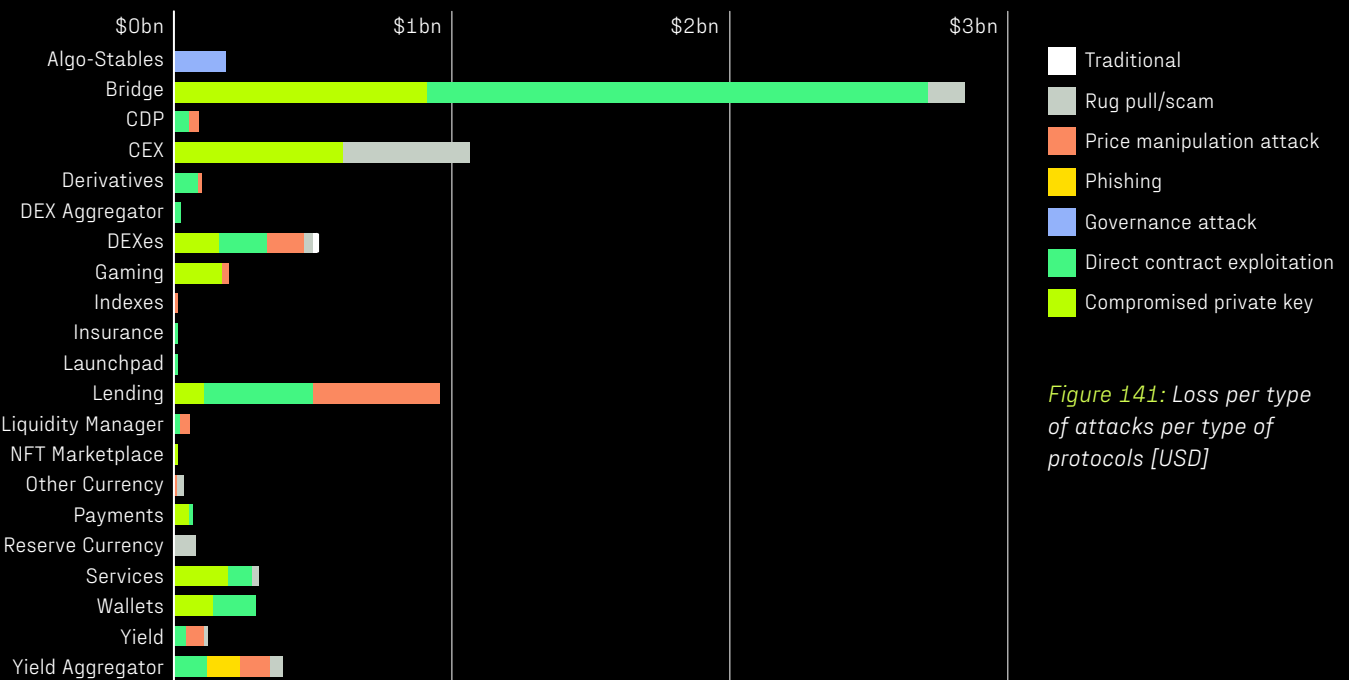


Figure 141: Loss per type of attacks per type of protocols [USD]

If we compare the distribution of types of attack per protocol by year (Figures 142 and 143) and in total (Figures 138 and 139), we can see that they are fairly similar.

Some noticeable exceptions are **Bridges, Wallets** and **Payments**. At the beginning, these were mostly attacked via **direct contract exploitation** while, most recently, the primary cause of attacks has been a **compromised private key**. **Yield Aggregator** and **Derivatives** protocols have evolved from being mostly attacked by **direct contract exploitation** to mostly by **price manipulation** attacks. **Yield** protocols also were **rug pulled** in the last year while in previous years they were the subject of other types of attacks.



Figure 142: Number of type of attacks per type of protocols [percentage]

Figure 143: Number of type of attacks per type of protocols [count]

In Figures 144 and 145, we can observe that, in general, there does not seem to be a big difference in various types of attacks' percentage of losses versus their rates of occurrence.

The most noticeable changes are CEXs in 2022, where rug pulls accounted for 86.5% (\$450,000,000 USD) of the loss versus only 25% of the occurrences and Yield Aggregators in 2021 where phishing attacks led to 51.7% (\$120,000,000 USD) of the value lost versus while making up only 16.7% of the hacks for that year. Regarding trends, it also does not seem to show any difference from the previous charts (Figures 142 and 143).



Figure 144: Loss per type of attacks per type of protocols [percentage]

Figure 145: Loss per type of attacks per type of protocols [USD]

TYPE FUNCTIONS

When interacting with a protocol's smart contracts, different functions can be used to attack it.

In order to better categorize and understand which function types are usually targeted, the following categories of functions have been considered based on their functionality. It should be taken into account that we are only considering those functions callable by the user (public or external) on the protocol's smart contract:

deposit: The main purpose of the function is to deposit assets to the protocol.

withdraw: In this case, the function is used to withdraw assets from the protocol. Borrows are also included in this category.

swap: The function is used to swap assets. It can call an internal function or protocol to know how many assets to swap for another.

mint: The function is used to mint assets. It can call an internal function or protocol to know how many assets to mint.

execute: The function is used to execute certain functionality on the protocol, like proposals.

transferOwnership: The function is used to transfer the ownership or special privileges of a contract or protocol.

initialize: The function is used to initialize the protocol.

upgrade: The function is used to upgrade the protocol. Especially relevant to proxy contracts.

calculateAmount: The main purpose of this function is to calculate some kind of quantity relevant to the protocol like pool liquidity or ratio.

verifyProof: The function objective is to verify certain actions/roles on the blockchain. Especially relevant on bridges.

migrate: The function is used to migrate between protocol versions.

create: The function is used to create another contract.

Protocol specific: The function is used to perform another action very specific to a certain protocol.

If we observe **Figures 146 and 147**, we can see that, by quantity, **withdraw-like functions are the most commonly attacked with 23.3% of the total.**

The second most common are **deposit** functions, with 18.6% of the total. **swap** and **initialize** occupy third place with 11.6% each.

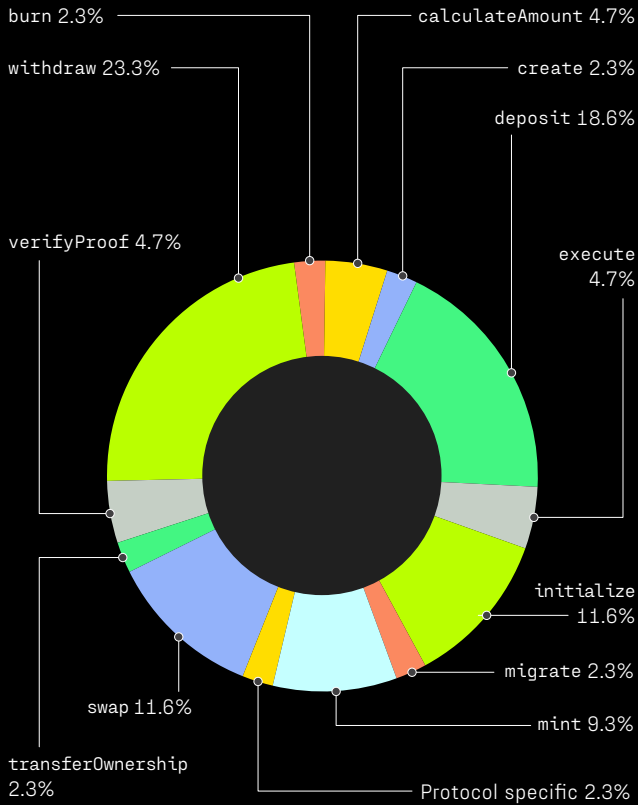


Figure 146: Number of type of function [percentage]

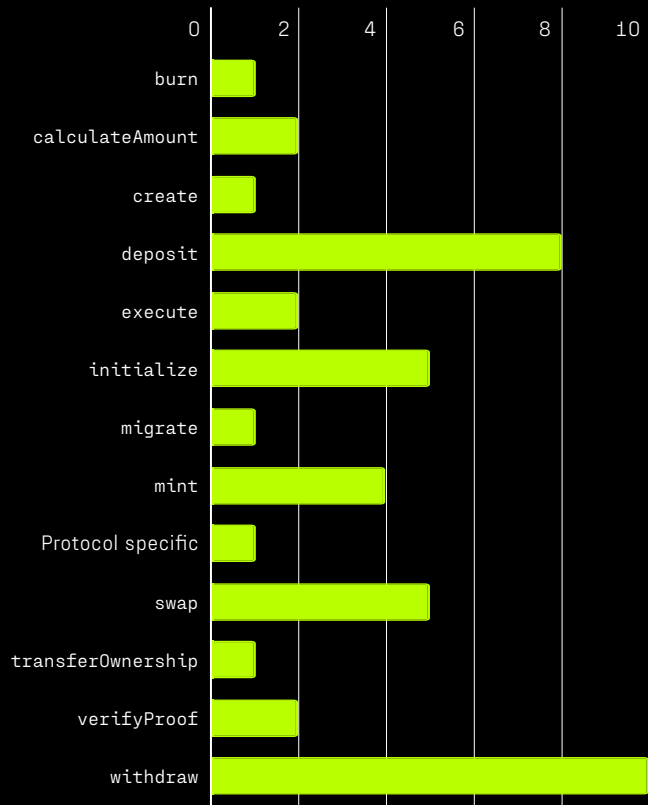


Figure 147: Number of type of function [count]

However, if we check which kind of function results in higher losses (Figures 148 and 149), we can see that **verifyProof** accounts for 27.2% of the total (\$912,000,000.00 USD) and only represents 4.7% of the functions attacked.

transferOwnership is the second most expensive, with 18.2% (\$611,000,000.00 USD), while there is only one attack that used this function. The third place goes to **deposit**-like functions with 11.7% of the total (\$391,600,000.00 USD), slightly less than their rate of occurrence.

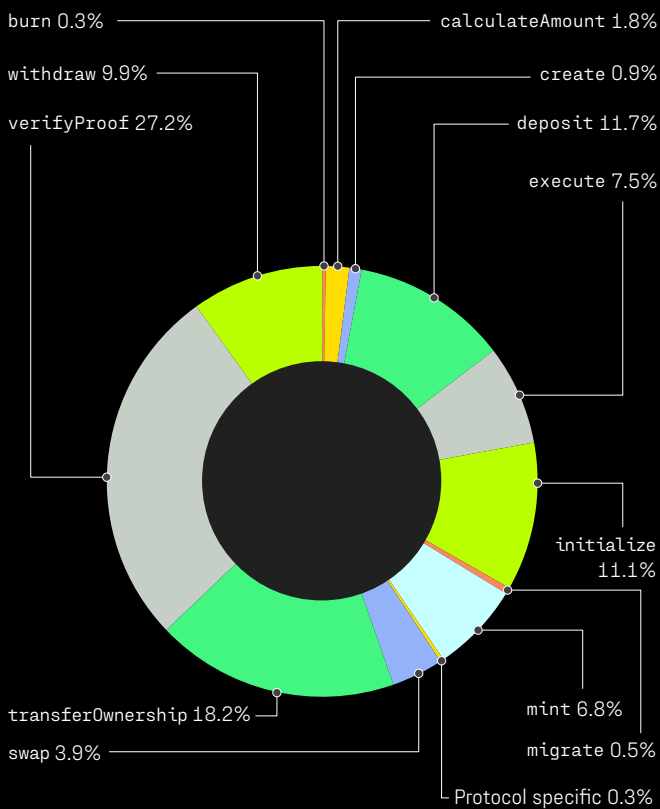


Figure 148: Loss caused by type of function [percentage]

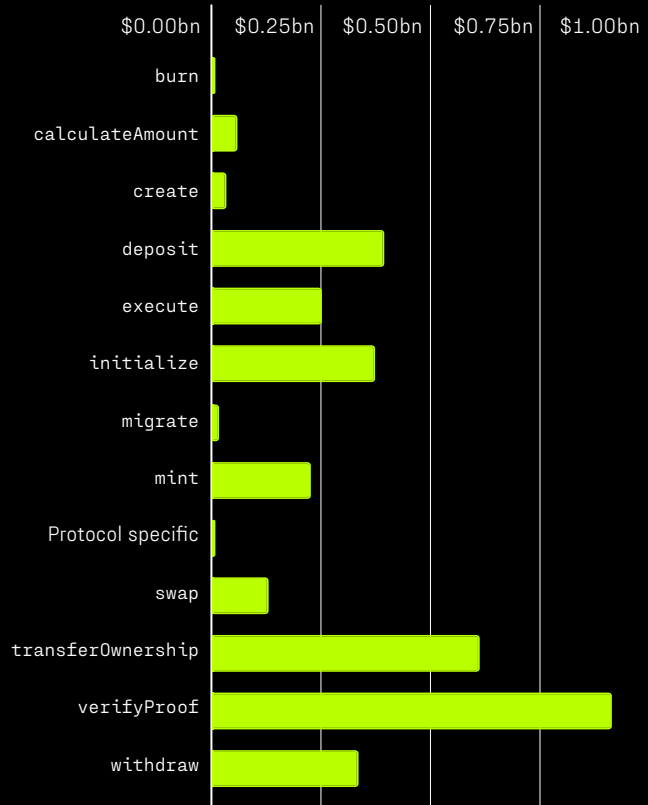


Figure 149: Loss caused by type of function [USD]

Type of Functions vs Chains

Per chain, we can observe that **withdraw** is the most common function attacked for Arbitrum and Terra, while it has the same proportion as **deposit** on Avalanche and as **mint** and **verifyProof** on Solana, also similar to **swap** on BSC (25%).

On Ethereum, they are **deposit** (20.8%), followed by **initialize** and **withdraw** (16.7%). For Fantom, the most attacked one was **deposit**. For Optimism, **create**. And for Polygon, **swap** and some other protocol-specific ones. In general, it seems that **withdraw** is one of the most commonly exploited functions for the majority of the chains.

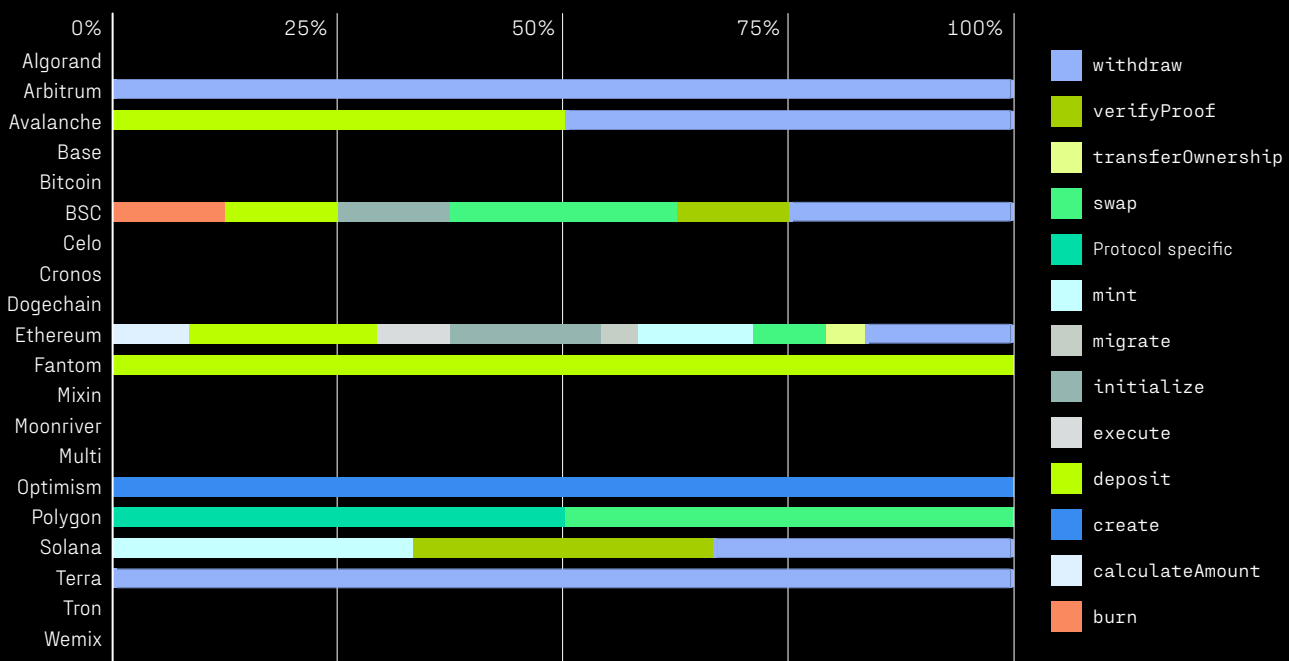


Figure 105: Number of type of attack per chain [percentage]

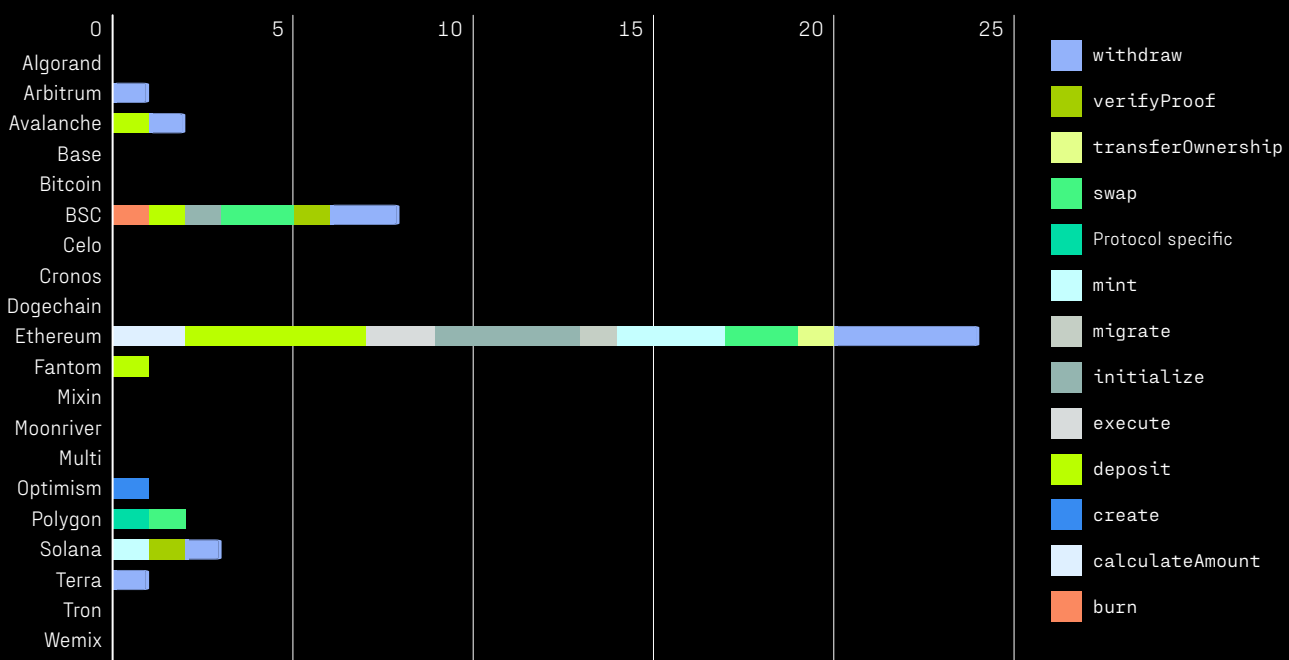


Figure 105: Number of type of attack per chain [percentage]

If we analyze the loss produced by each type of function per chain, we can observe that, for those chains with only one type of function, it remains the same.

However, we can see how **deposit** functions cause the majority of the losses on Avalanche (80%, \$34,000,000.00 USD) versus 50% by number of attacks. On BSC, **verifyProof**-like functions add up to 73.1% (\$586,000,000.00 USD) of the losses for all functions while having a rate of occurrence of 12.5%. On Ethereum, **transferOwnership** has caused the majority of the loss of funds (33%, \$611,000,000.00 USD), while only being used once. On Polygon, the ones responsible for the biggest losses are **swap** functions with 78.3% (\$31,400,000.00 USD) versus a 50% of the total number of attacks. On Solana, **verifyProof** is primarily responsible for the losses, with 85.2% (\$326,000,000.00 USD), while only accounting for 33.3% of the attacks. In this case, while **withdraw** is the most attacked function for the majority of the chains, it seems like **verifyProof** is the one that causes more loss, especially for those chains where more than one type of function is attacked.

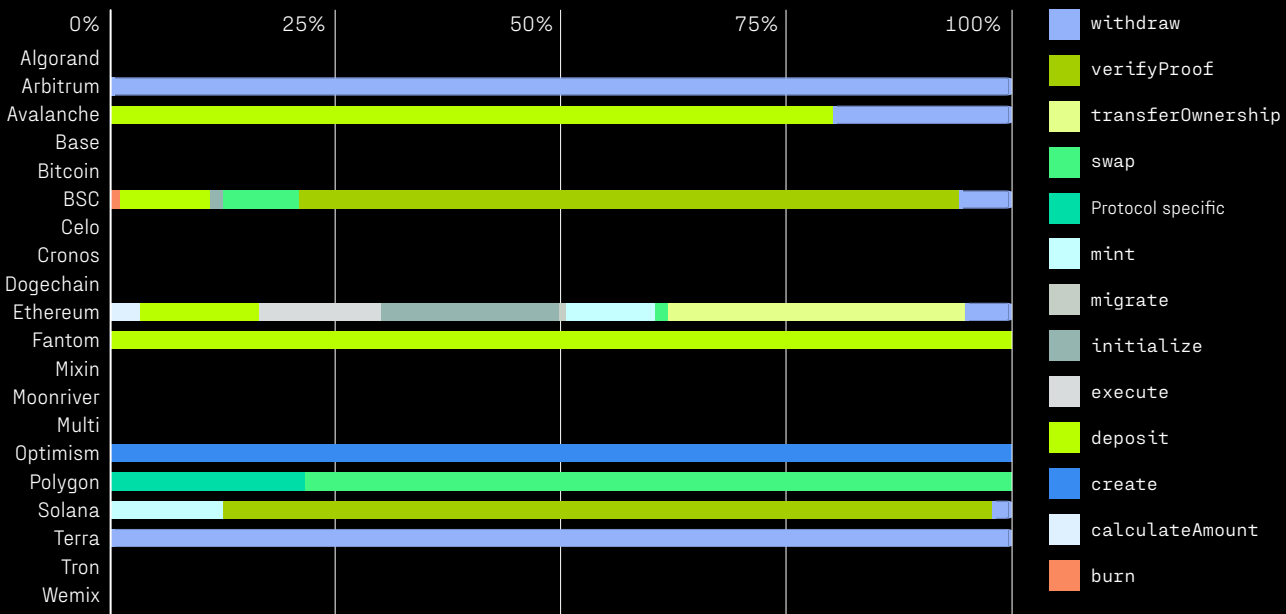


Figure 152: Loss caused per type of function per chain [percentage]

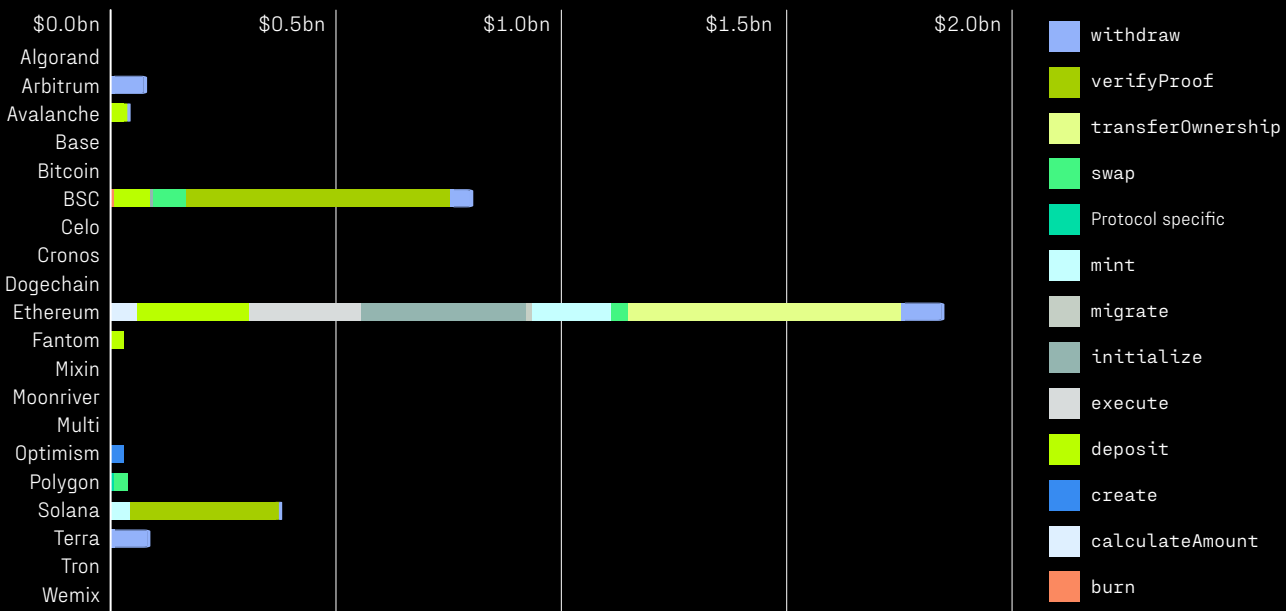


Figure 153: Loss caused per type of function per chain [USD]

By year, we can observe that, for example, BSC goes from attacks being mostly due to **swap** and **withdraw** functions, to **burn** in the last year.

On Ethereum, we can see how it evolves from **execute** in 2016, to **calculateAmount** in 2023. In general, there does not seem to be any distinguishable trend, however.



Figure 154: Number of type of function per chain and year [percentage]

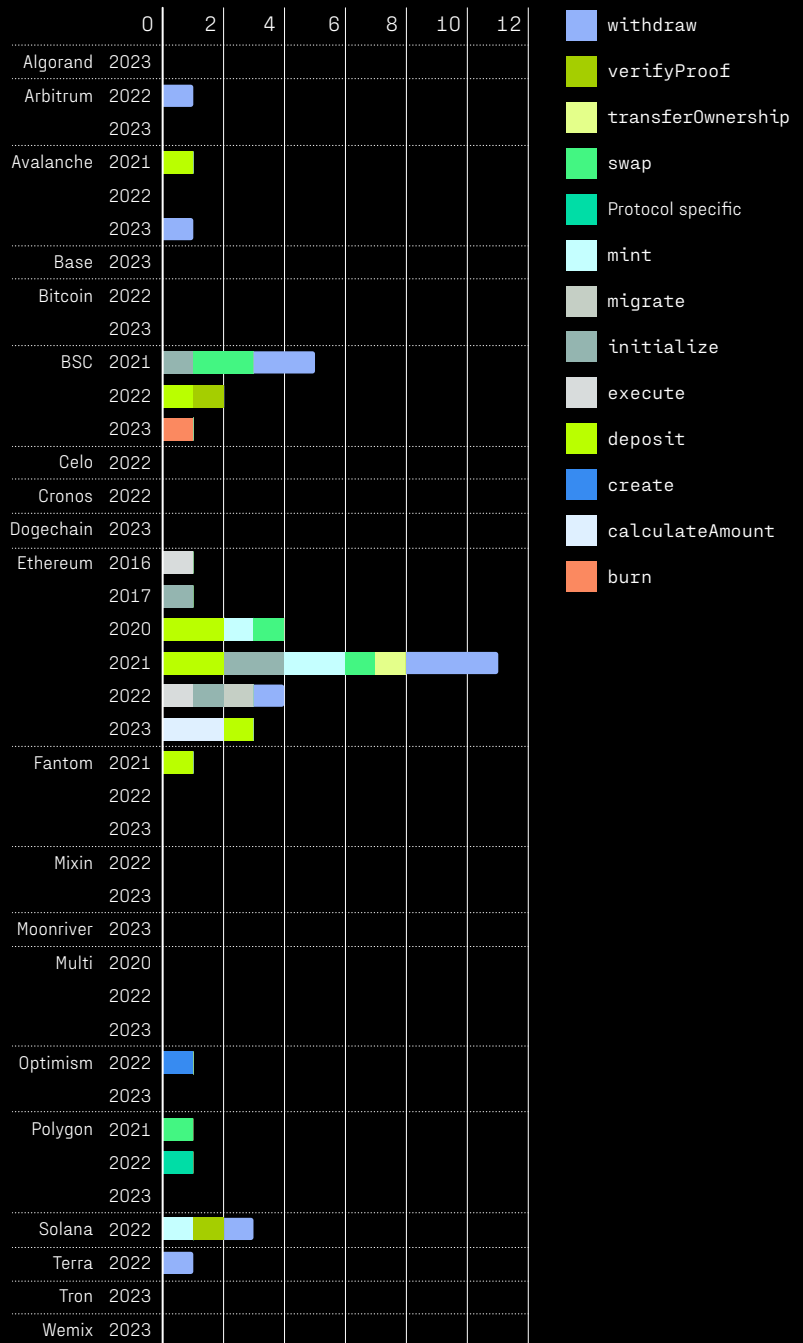


Figure 155: Number of type of function per chain and year [count]

By loss (Figure 156 and Figure 157) we can observe that, for example, in the case of Ethereum, the majority of the losses in the last year were caused because of deposit functions.

This is in contrast to calculateAmount, which is the most commonly attacked. There does not seem to be a noticeable trend either.



Figure 156: Loss caused by type of function per chain and year [percentage]

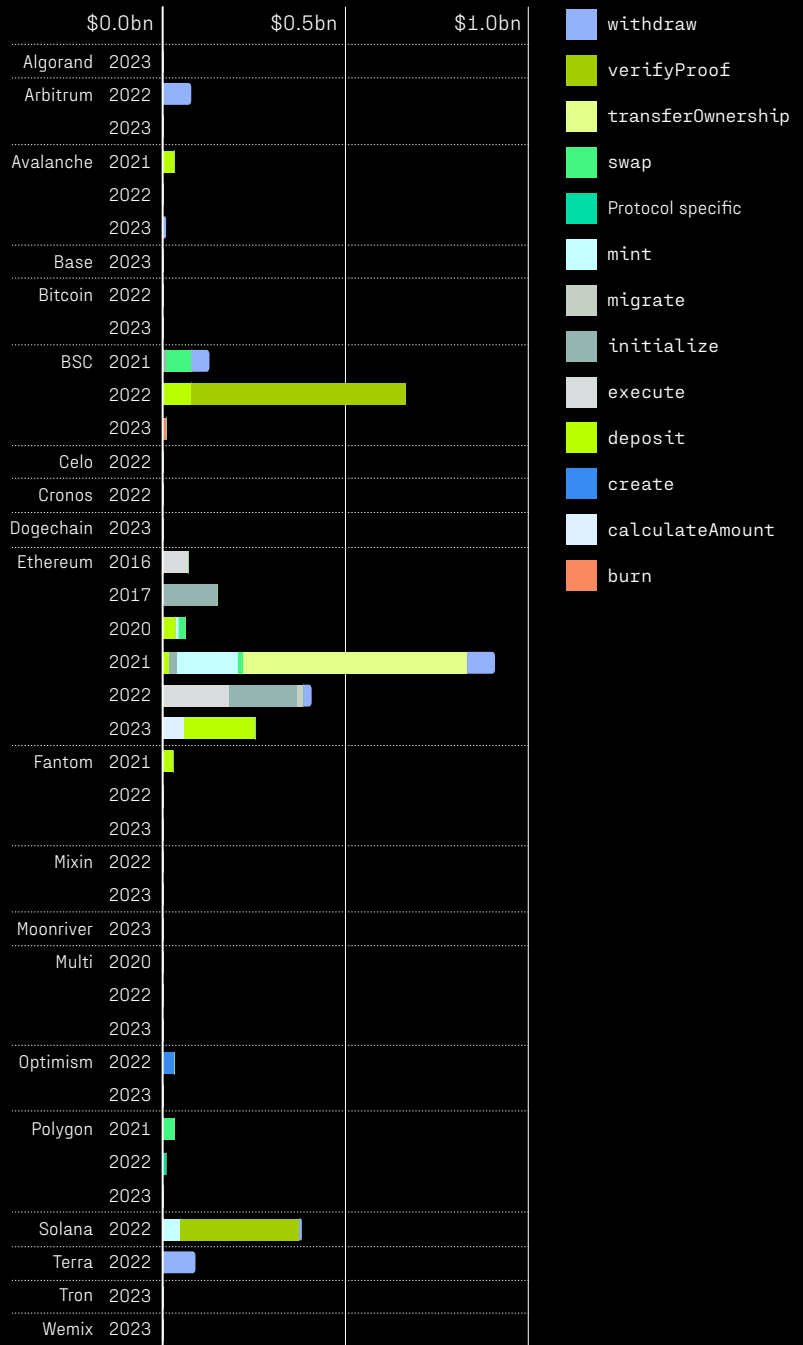


Figure 157: Loss caused by type of function per chain and year [USD]

- withdraw
- verifyProof
- transferOwnership
- swap
- Protocol specific
- mint
- migrate
- initialize
- execute
- deposit
- create
- calculateAmount
- burn

Type of Functions vs Types of Attacks

In order to better understand which types of functions are used in different types of attacks, Figures 158 and 159 are presented. We can observe that only those types of attacks that interact with functions have data. It can be observed that, for **direct contract exploitation**, the most common type of function used is **withdraw** (25%), followed by **deposit** (21.9%) and **initialize** (15.6%). For **governance** attacks, the only function used is **execute**, because the attack was possible via a malicious proposal execution. In the case of **price manipulation** attacks, **swap** is the most common function used to attack the protocols (30%), followed by **withdraw** and **calculateAmount** (20%). This latest is typically used to calculate some parameter of the pool. It also makes sense because **price manipulation** attacks usually involve some kind of swap, price or amount calculation mechanism to exploit and withdraw assets.

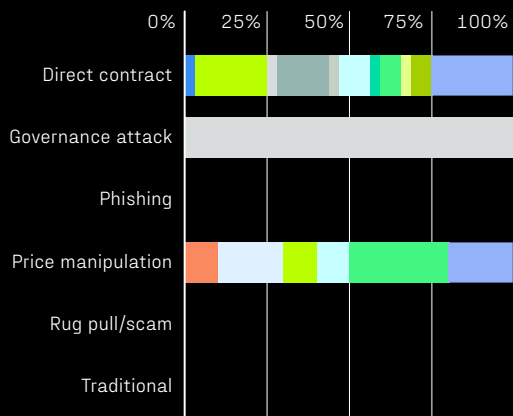


Figure 158: Number of type of function per chain [percentage]

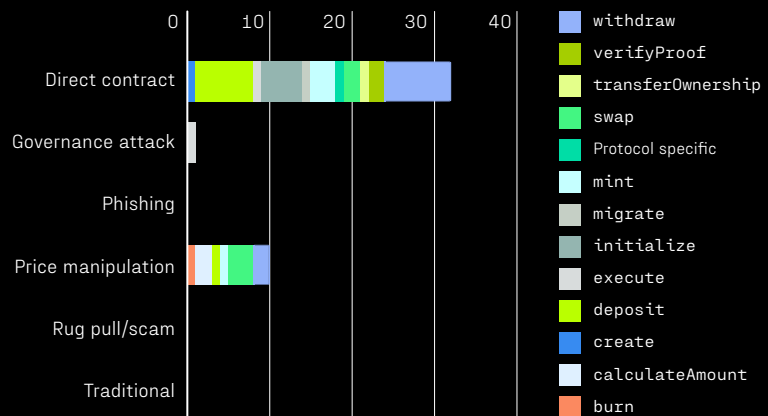


Figure 159: Number of type of function per chain [count]

By loss, we can observe that, in the case of **direct contract exploitation**, the majority of it is due to **verifyProof** functions (31.2%, \$912,000,000.00 USD), followed by **transferOwnership** (20.9%, \$611,000,000.00 USD) and **initialize** (12.8%, \$372,950,000.00 USD). This seems to suggest that **Bridges** are the main cause of losses because of **direct contract exploitation**. For **price manipulation** attacks, we can observe that the majority of the funds are lost because of **withdraw**-like functions (27.5%, \$68,000,000.00 USD), followed by **calculateAmount** (24%, \$59,400,000.00 USD) and **swap** (21.6%, \$53,400,000.00 USD).

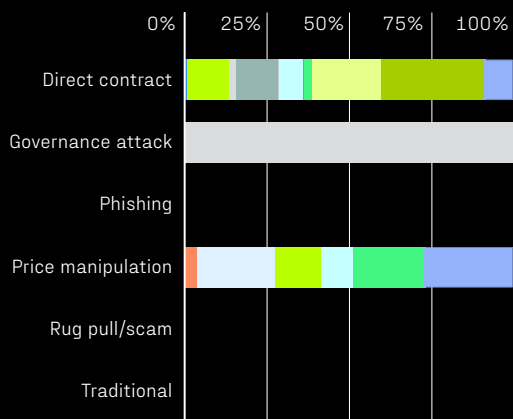


Figure 160: Loss caused by type of function per chain [percentage]

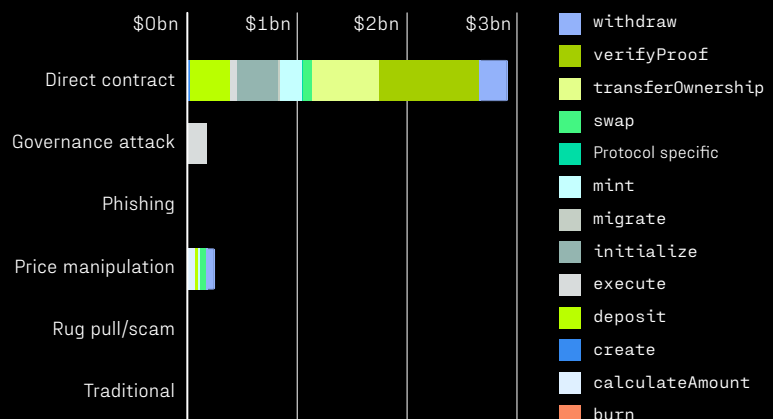


Figure 161: Loss caused by type of function per chain [USD]

By year, we can observe that, for **direct contract exploitation**, using a **withdraw** function for attacking seems to be a trend in recent years (2021-2023).

Regarding **price manipulation** attacks, it has shifted from mostly attacks using **swap** functions, to taking advantage of **burn** functions to destabilize the price and **calculateAmount** functions.

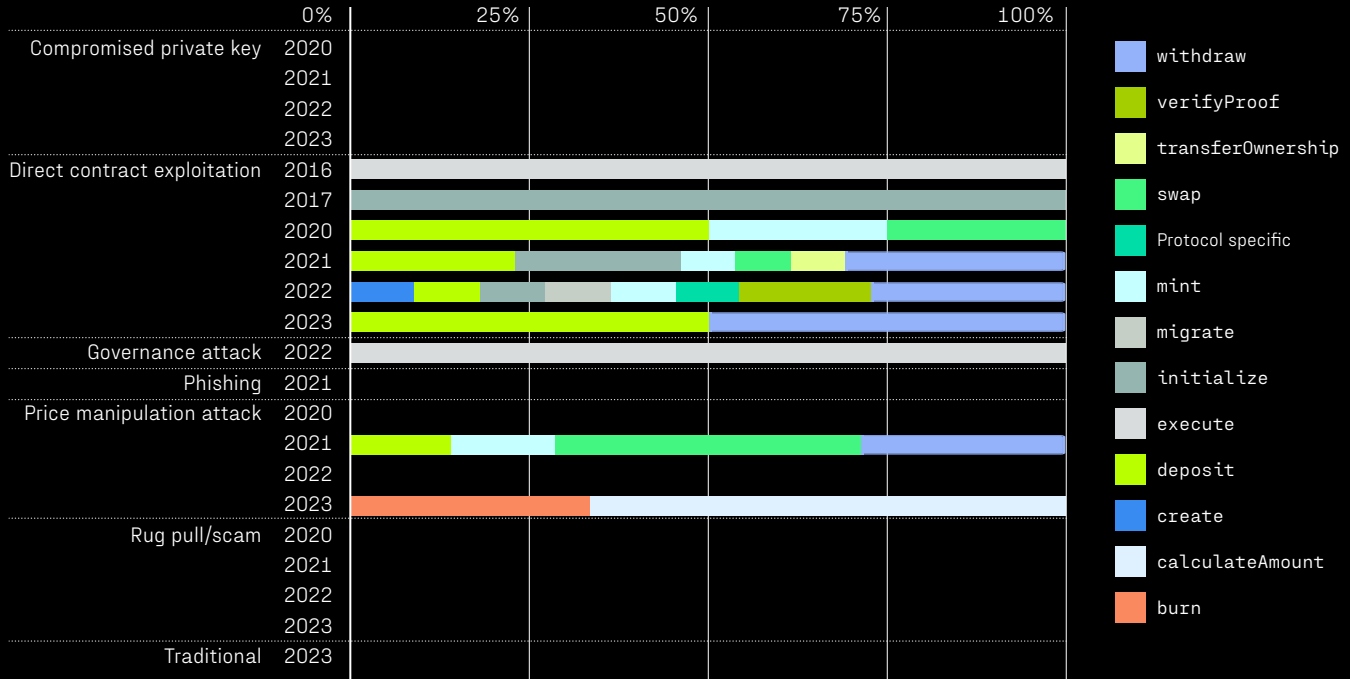


Figure 162: Number of type of function per chain and year [percentage]

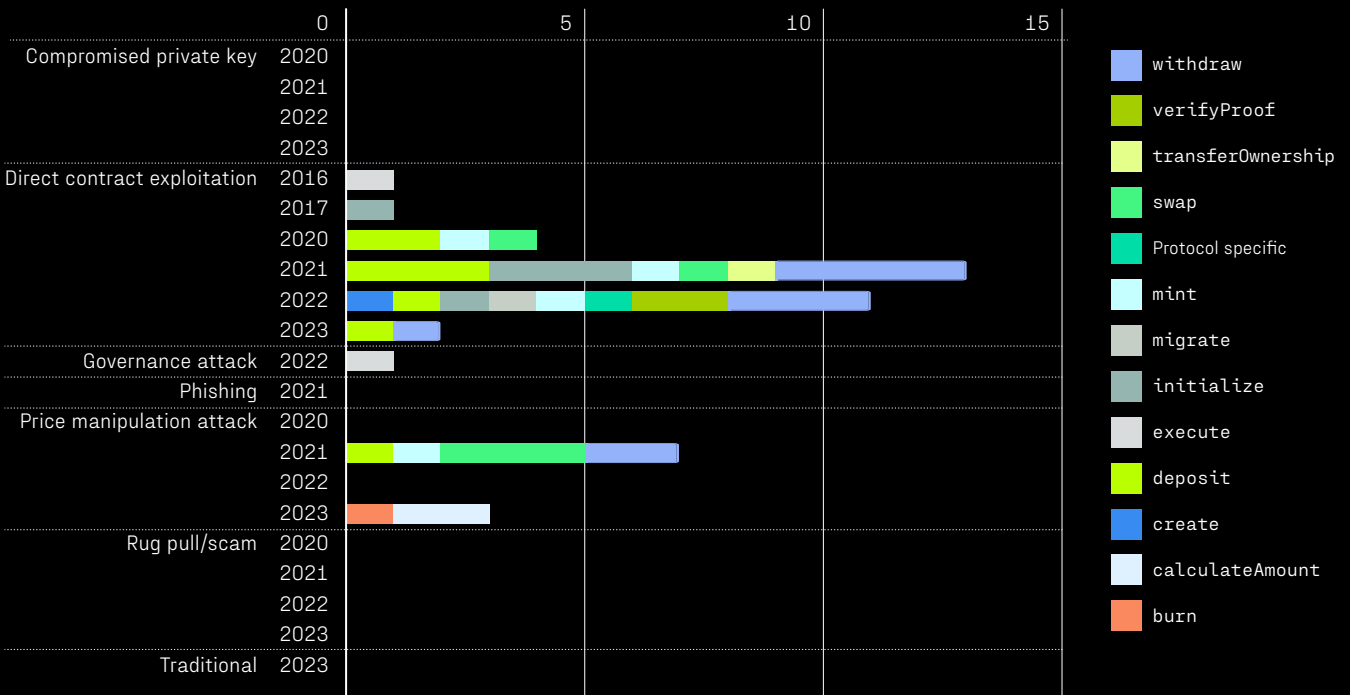


Figure 163: Number of type of function per chain and year [count]

In terms of losses, we can see how, in direct contract exploitation, deposit seems to be the primary cause in 2020 and 2023.

In 2022, the main cause was `verifyProof`, probably because of the increased use of `Bridges` (see Figures 164 and 165). Regarding `price manipulation` attacks, it is very similar to the previous charts (Figures 162 and 163).

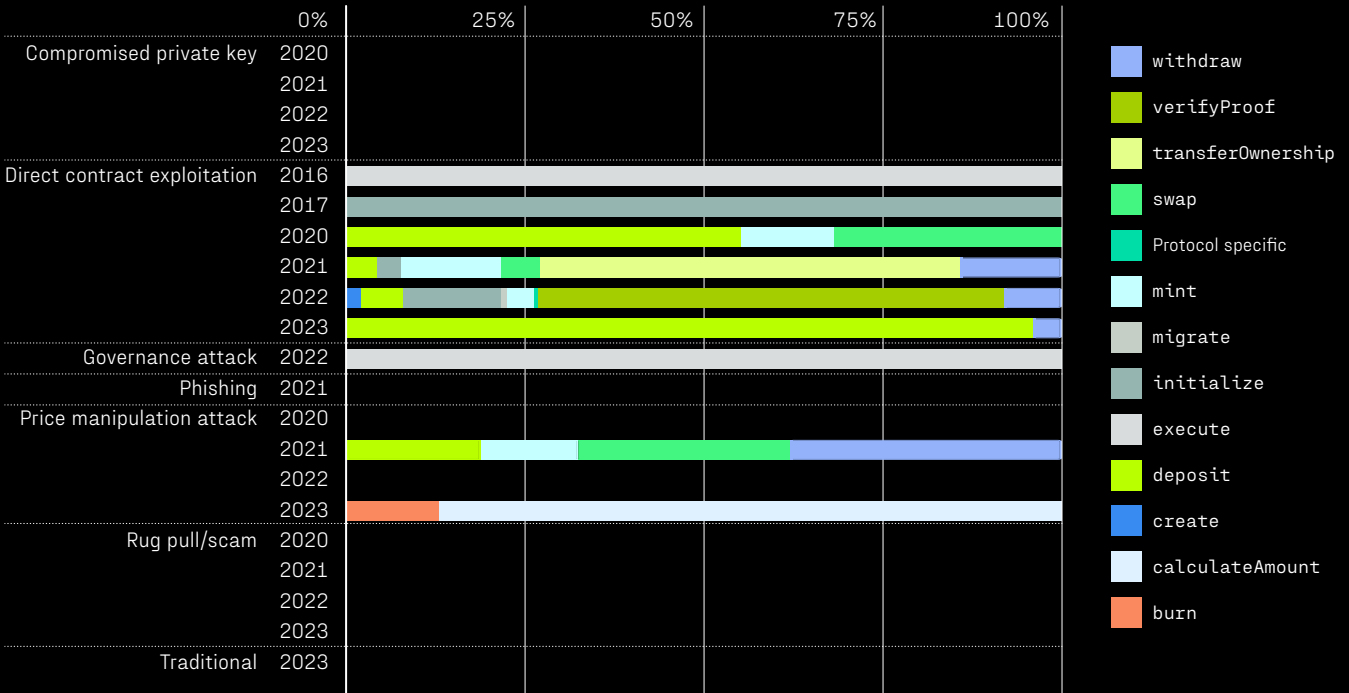


Figure 164: Loss caused by type of function per chain and year [percentage]

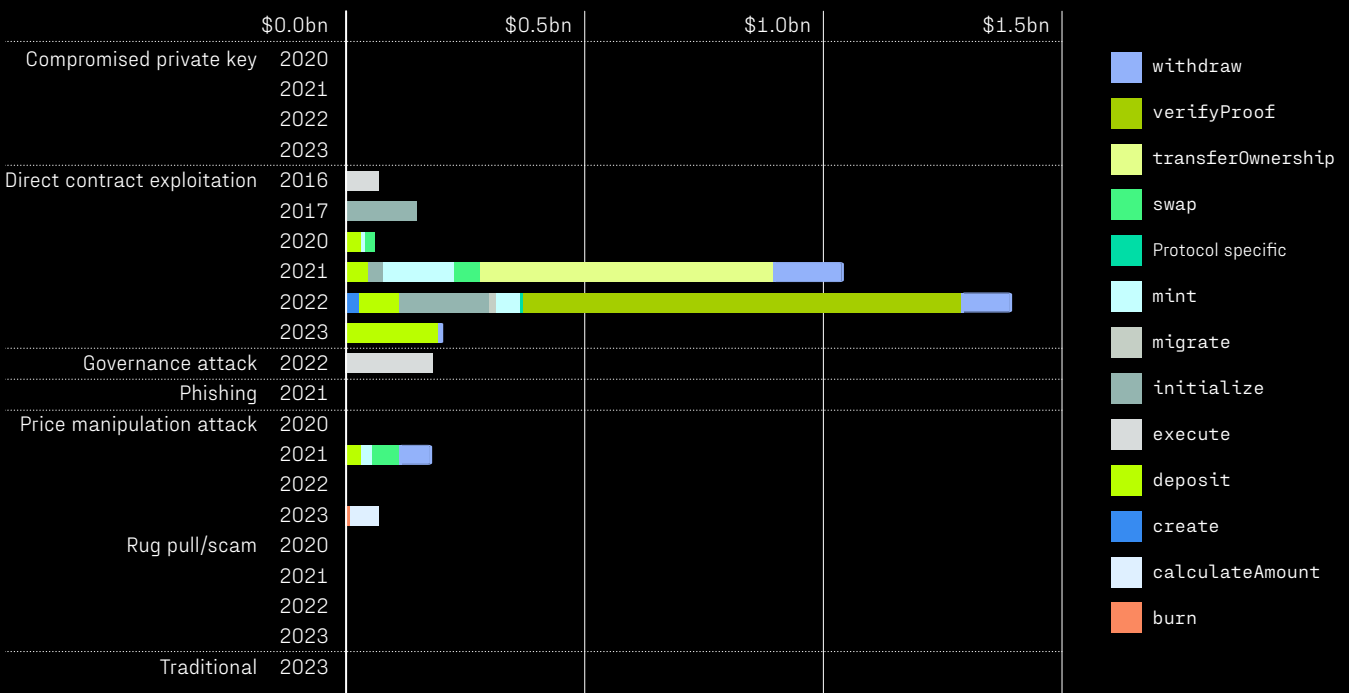


Figure 165: Loss caused by type of function per chain and year [USD]

Type of Functions vs Protocols

In **Figure 166** and **Figure 167**, we can observe that **execute** is the main function used to hack **Algo-Stables** and **Services** (together with **initialize**).

deposit is the most used function to attack **Insurance** and **Lending**, as well as being one of the two most common (along with **verifyProof**) in **Bridges** and (with **mint** and **withdraw**) in **Liquidity managers**. **withdraw** is the most commonly used function in **Derivatives**, **DEX Aggregators**, and **Yield** protocols and the most used alongside **mint** for **CDP**. **swap** is the most common function used to attack **DEXes**, while **migrate** is the one for **Launchpads**. For **Other Currency**, it has been the **burn** function. In **Payments**, protocol specific functions have been the ones used to attack the protocol. For **Wallets**, it has been the **initialize** function. Finally, for **Yield Aggregators**, it has been a combination of **withdraw**, **swap**, **mint**, **initialize** and **calculateAmount** in equal parts.

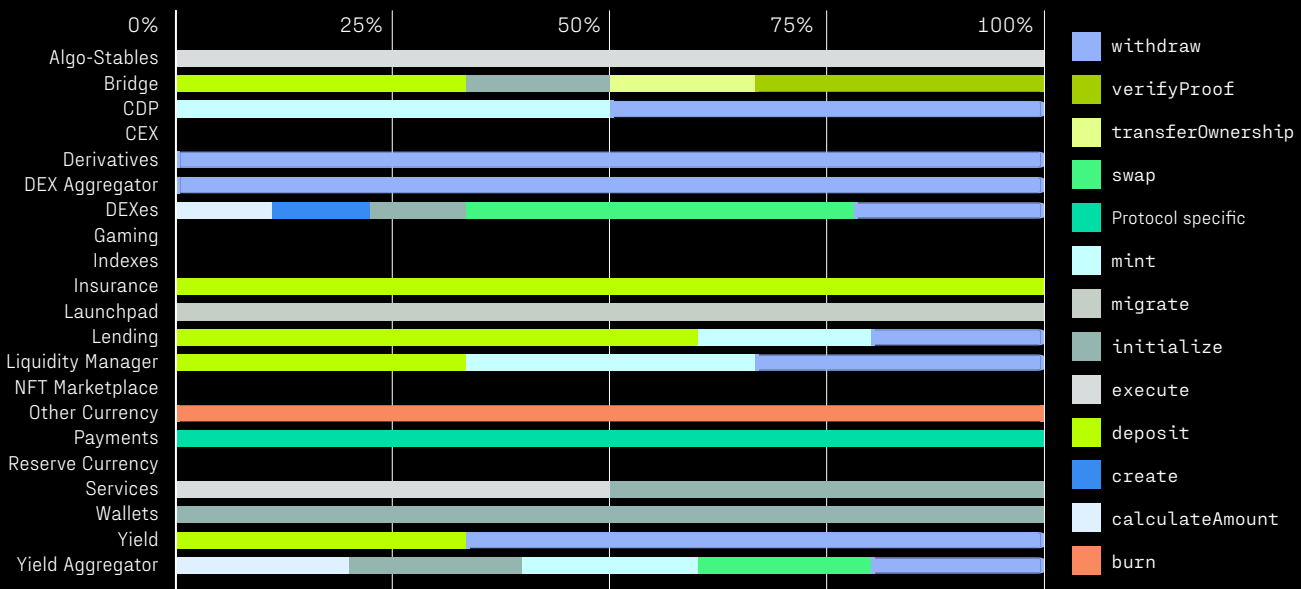


Figure 166: Number of type of function per type of protocol [percentage]

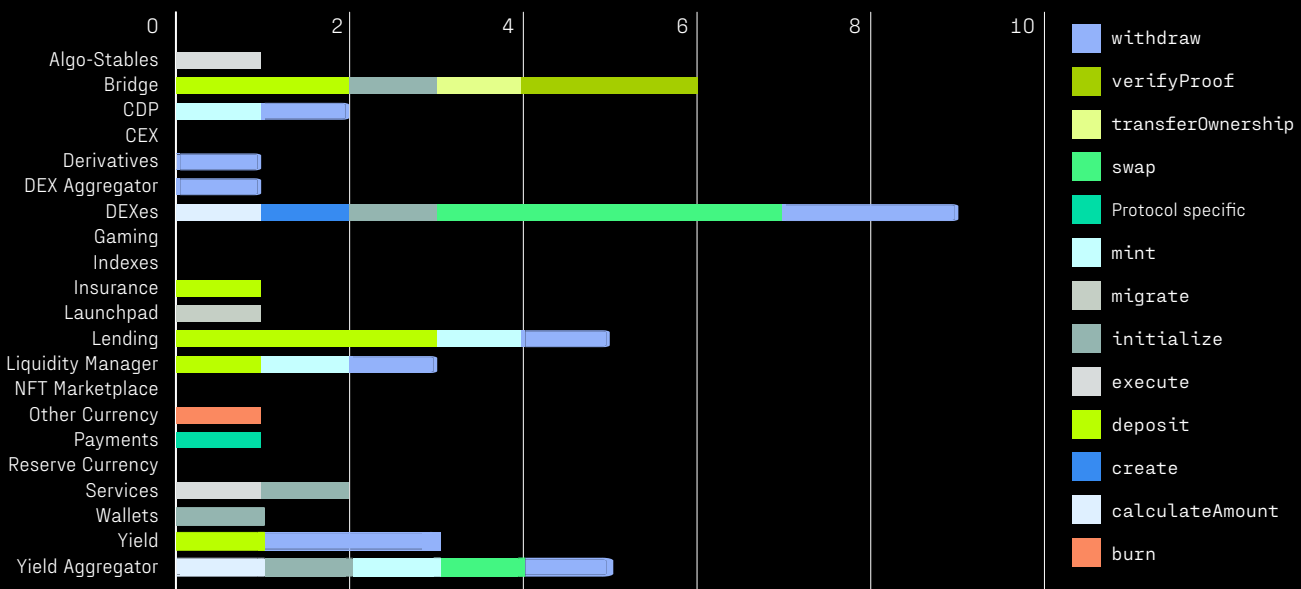


Figure 167: Number of type of function per type of protocol [count]

We can see how **verifyProof** has been the main cause of lost funds for **Bridges (50.6%, seen on Figures 168 and 169)**, although it is tied by rate of occurrence with **deposit**.

swap functions have been the main cause of losses for **DEXes**, besides being the most common type of attack. **withdraw** is the main cause in the same protocols as it was by number but also for **Yield Aggregators**. In **CDPs**, the majority of the losses have been due to **mint** functions, although they sum up only half of the attacks. The rest of the loss distribution is very similar to the distribution by occurrence.

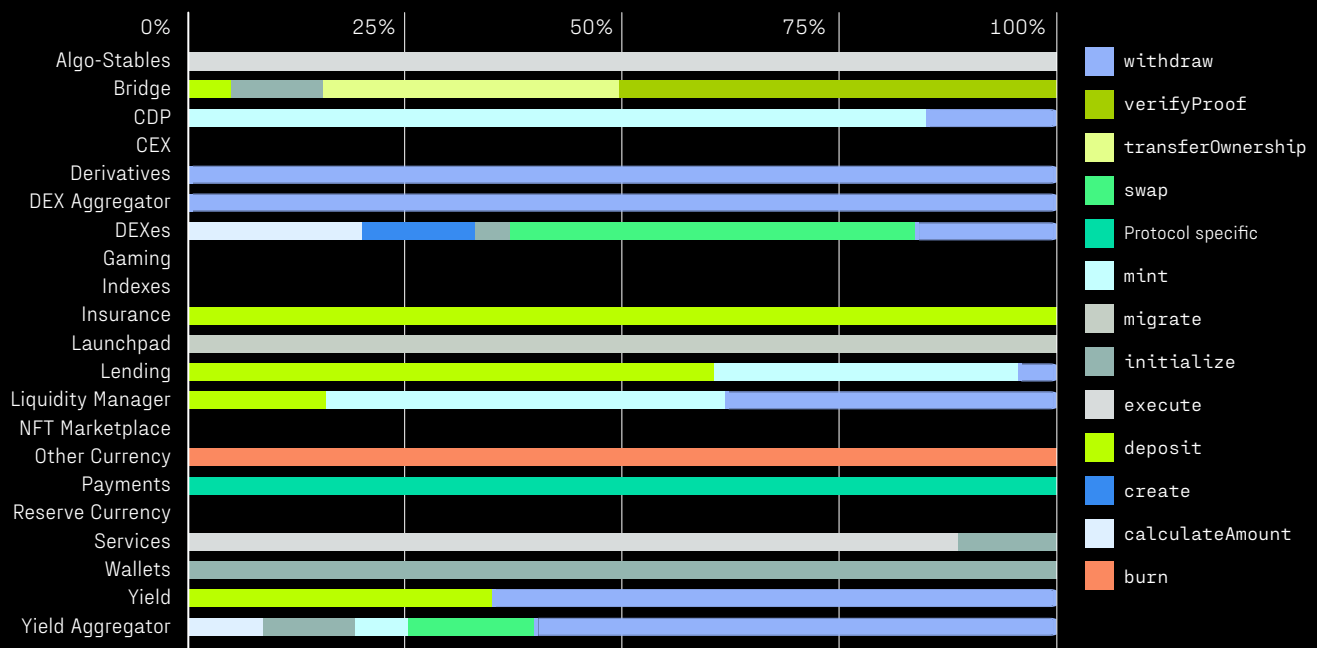


Figure 168: Loss caused by type of function per type of protocol [percentage]

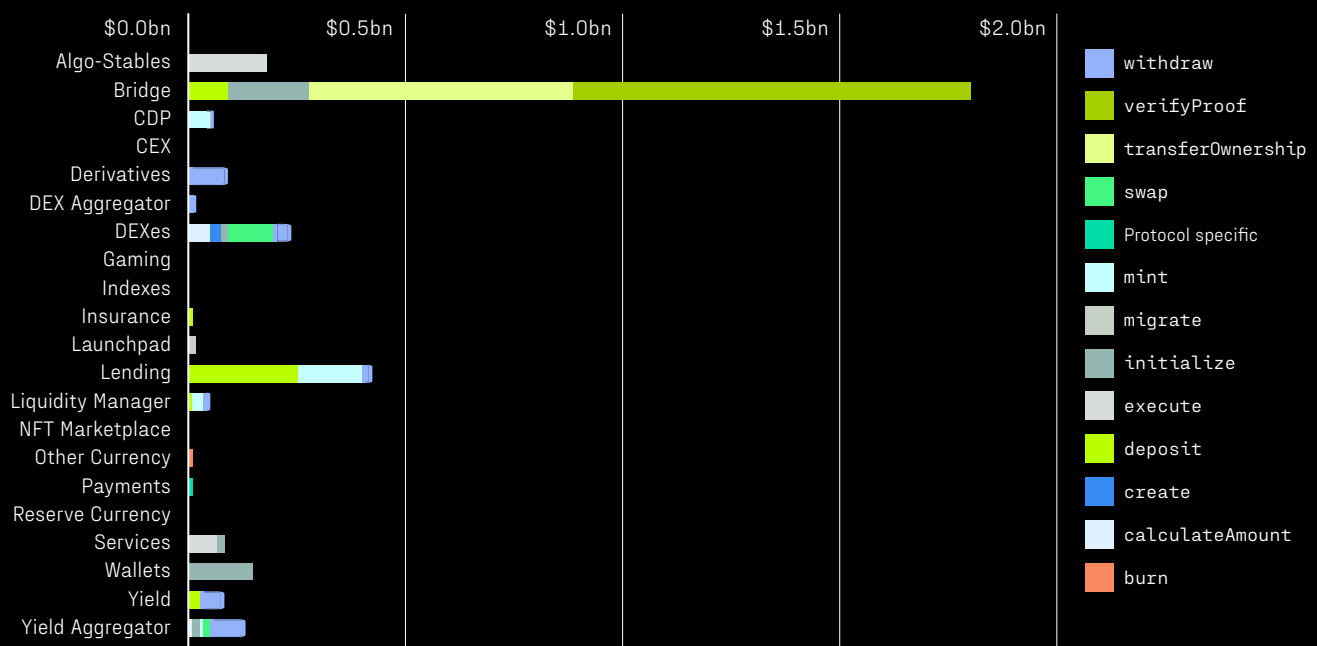


Figure 169: Loss caused by type of function per type of protocol [USD]

If we examine the distribution by year (Figures 170 and 171), it doesn't seem to show any noticeable trends.

Maybe something to point out could be that **Bridges** seem to have been exploited more by **VerifyProof** functions in recent years, compared to **transferOwnership** and **deposit** in earlier ones. On **DEXes**, causes vary depending on the year, although **withdraw** is present both in 2021 and 2022. **Lending** protocols seem to come back to being mostly exploited by **deposit** functions, after some variation of causes in 2021. Other protocols seem to have been exploited by different functions each year.

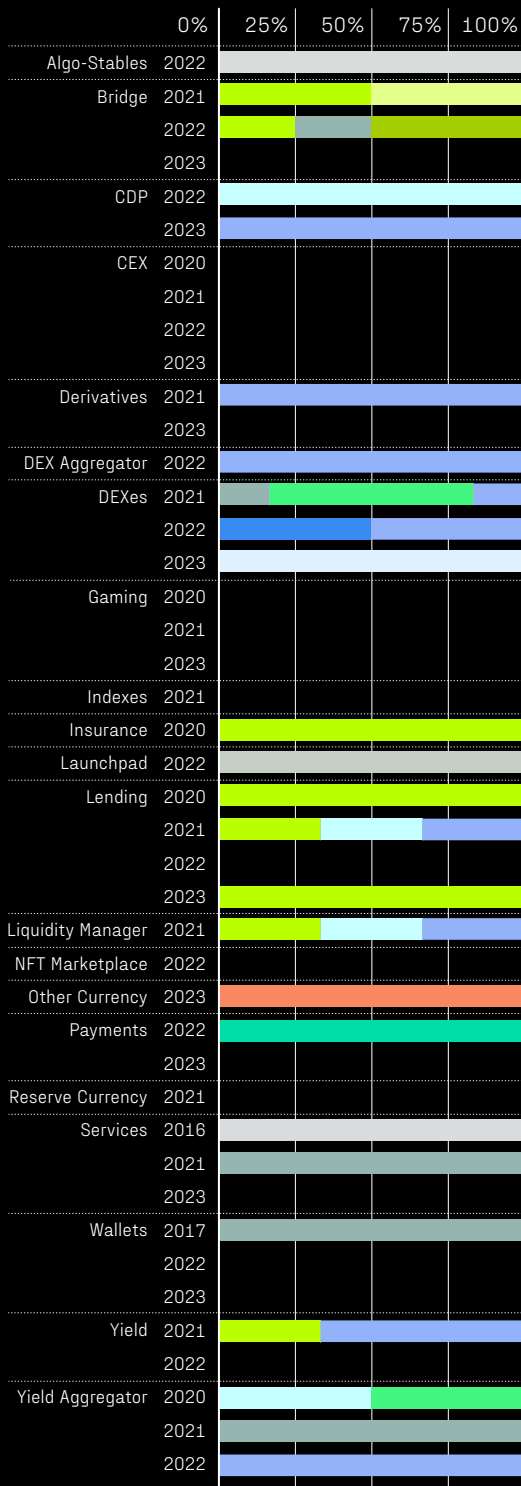


Figure 170: Number of type of function per type of protocol and year [percentage]

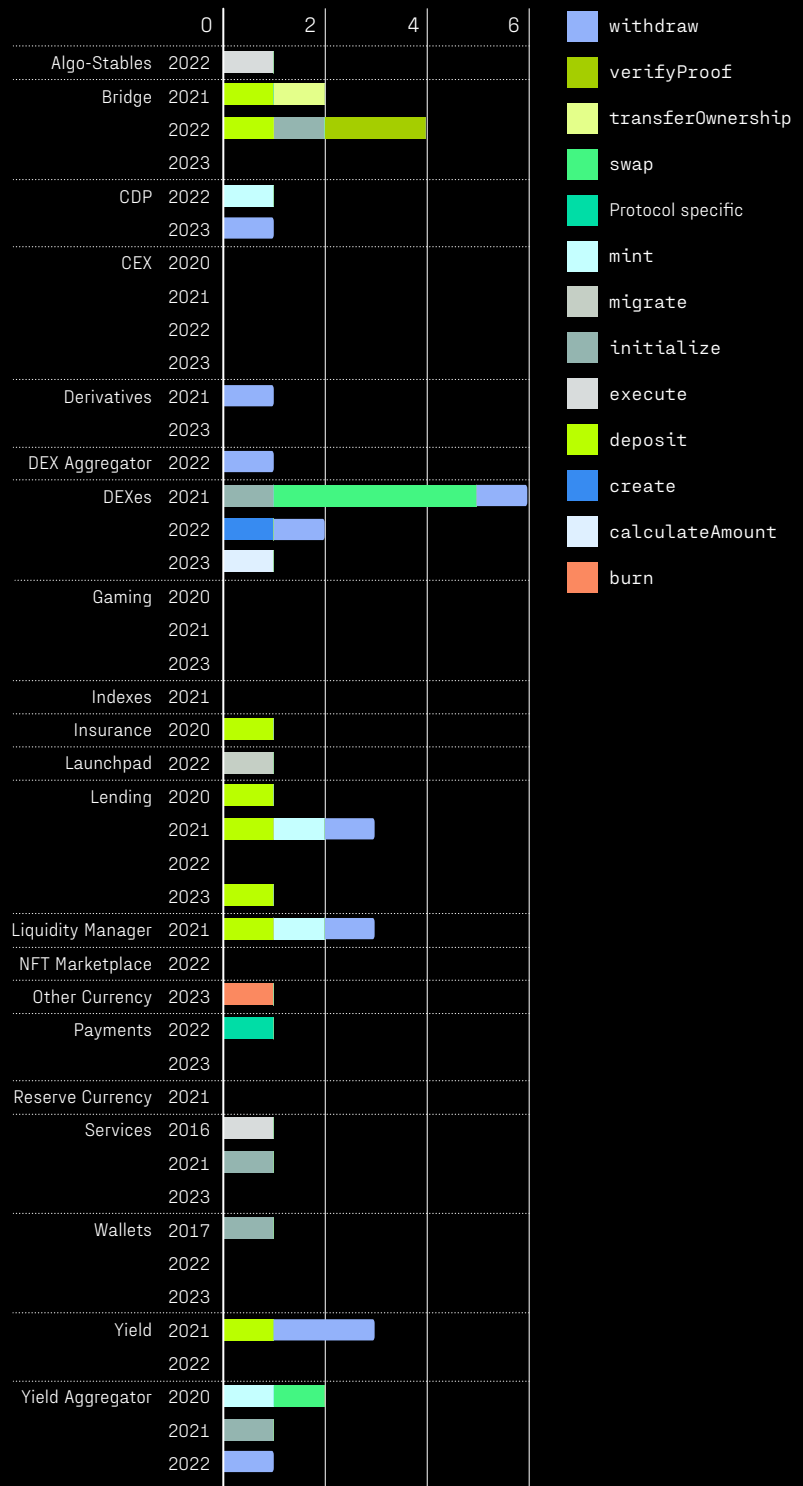


Figure 171: Number of type of function per type of protocol and year [count]

If we study the distribution by loss value (Figures 172 and 173), it also does not seem to follow any trend, although there are some differences compared with their occurrence.

In Bridges in 2023, `verifyProof` causes 77.2% (\$912,000,000 USD) of the stolen amount versus 50% of the occurrences. Something similar happens with `mint` functions in Lending protocols in 2021, accounting for 73.6% (\$147,000,000 USD) of the loss versus 33.3% of the attacks. The same is true for Liquidity managers, with 46% (\$24,000,000 USD) and 33.3% respectively. `swap` functions also caused more losses in Yield Aggregators in 2020, 71.1% (\$19,700,000 USD) versus 50%.

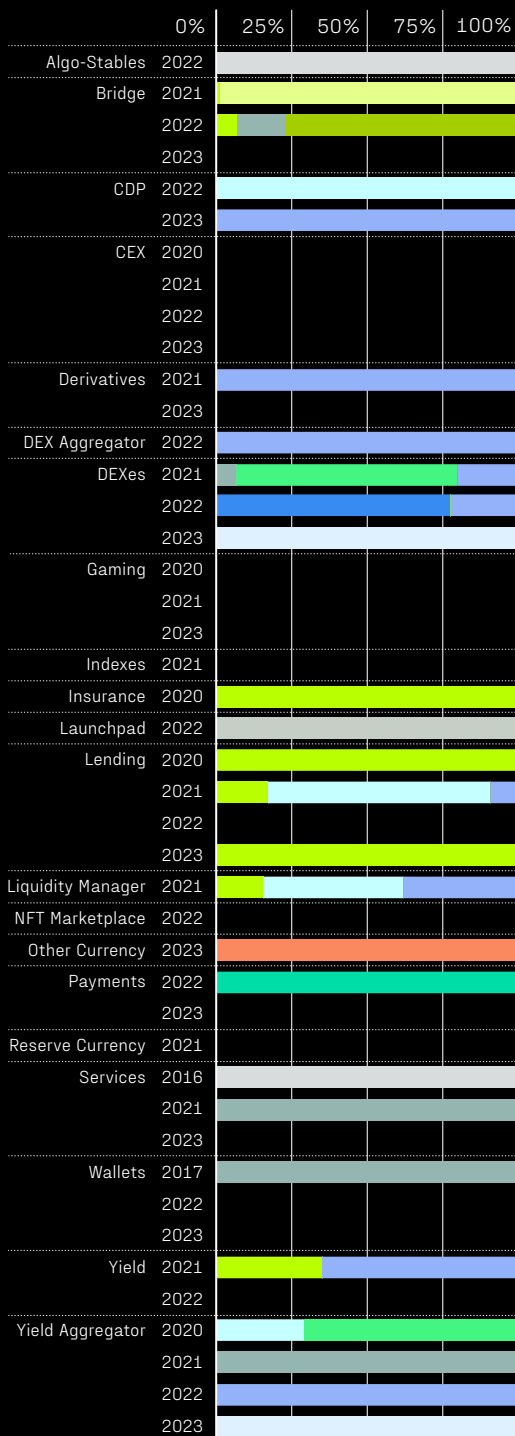


Figure 172: Loss caused by type of function per type of protocol and year [percentage]

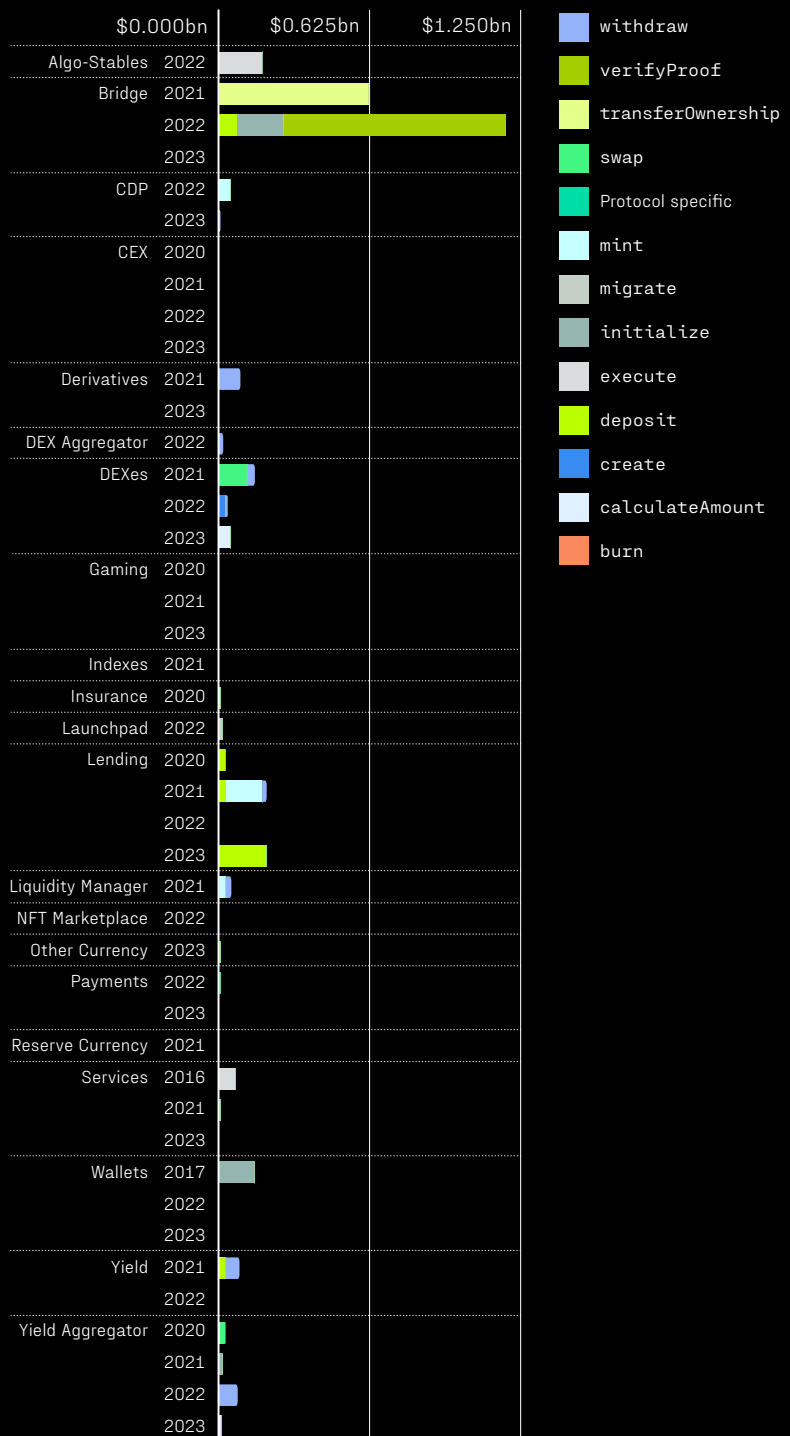


Figure 173: Loss caused by type of function per type of protocol and year [USD]

WERE THEY AUDITED?

Security audits of protocols' smart contracts can help to prevent attacks.

We can observe in Figures 174 and 175 that only 20% of the protocols hacked were audited, while 39% were not. 41% of them are not applicable (rug pulls and off-chain attacks).

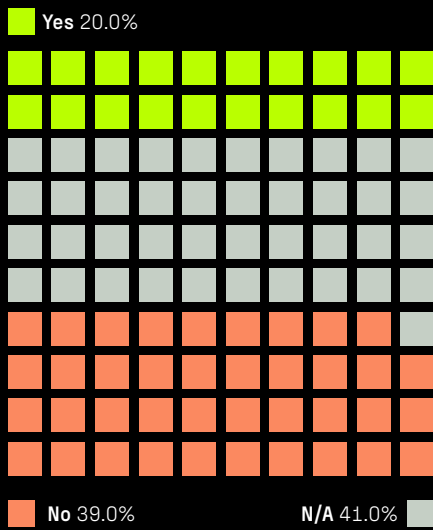


Figure 174: State of audition [percentage]

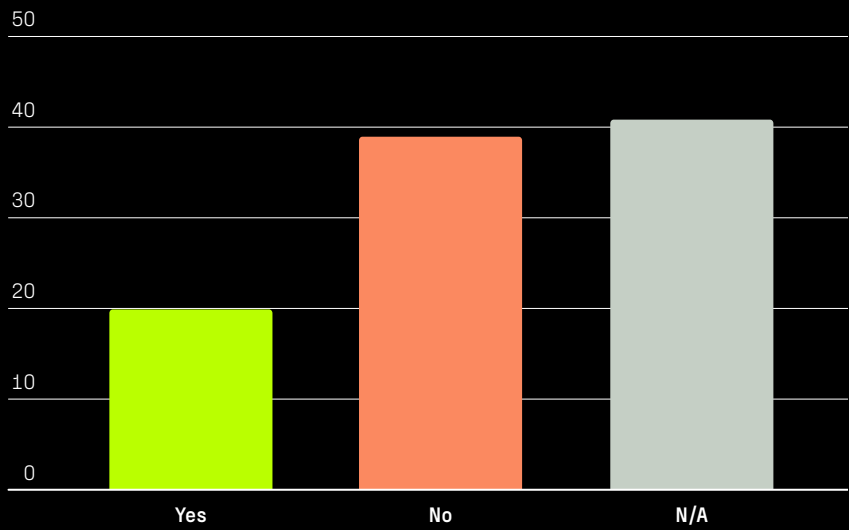


Figure 175: State of audition [count]

By loss, we can see in Figures 176 and 177 that the amount lost in those not audited is greater than in those audited, accounting for 39% of total losses (\$2,870,450,000.00 USD). In proportion, audited protocols seem to have slightly lower losses compared to their occurrence (14.3% of the loss vs 20% of the total attacked), adding up to \$1,049,223,000.00 USD stolen.

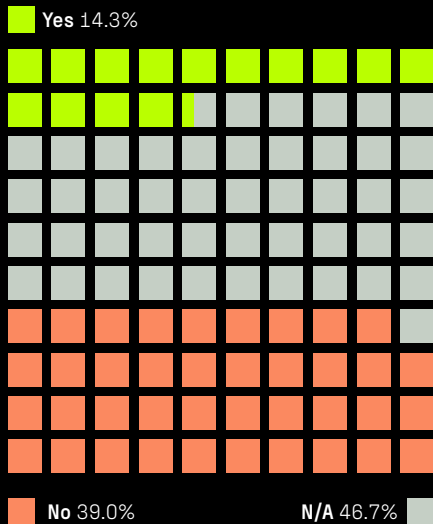


Figure 176: Loss caused per state of audition [percentage]

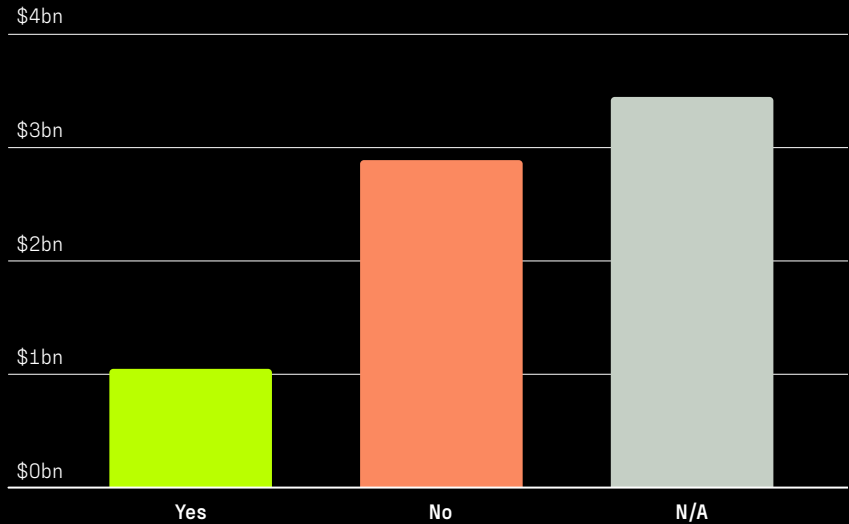


Figure 177: Loss caused per state of audition [count]

If we examine all this data over time (Figure 178 and Figure 179), we can observe that the percentage of protocols audited that have been hacked has decreased over time, from 25% of the total in 2020 to 8.7% in 2023.

For those not audited, the loss has also decreased, from 50% to 21.7% in favor of attacks that are not related to smart contracts audits (25% to 69.6%).

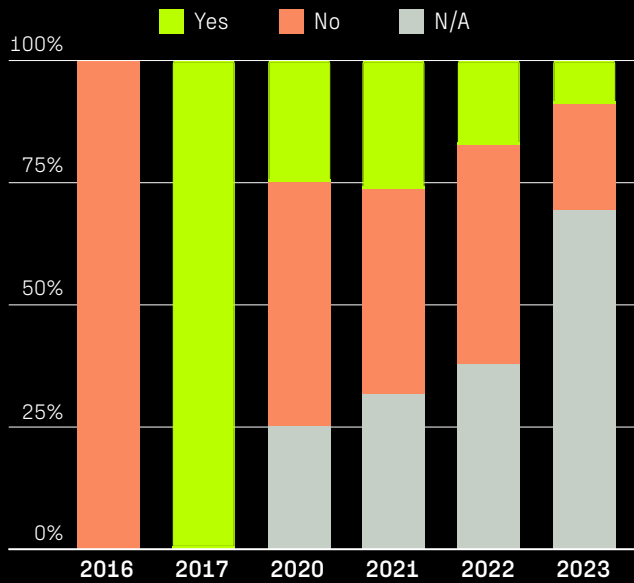


Figure 178: State of audition per year [percentage]

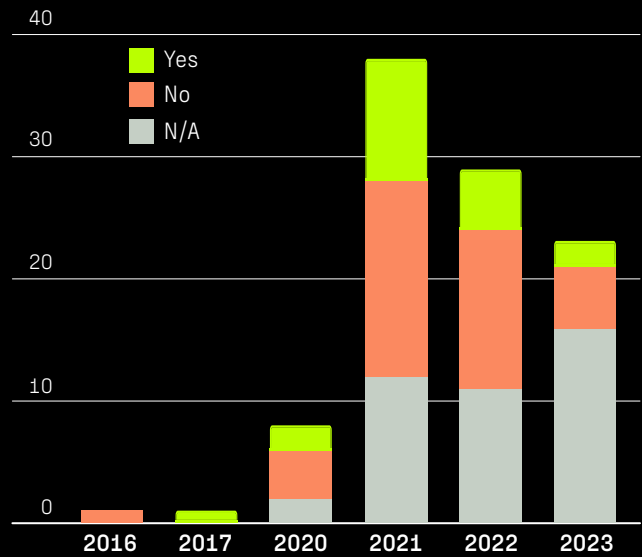


Figure 179: State of audition per year [count]

By loss, we can observe that these decreases have been slightly less, while the loss for non-smart contract related vectors increased from 35.8% (\$57,000,000 USD) to 72.5% (\$1,062,214,951 USD, see Figures 180 and 181).

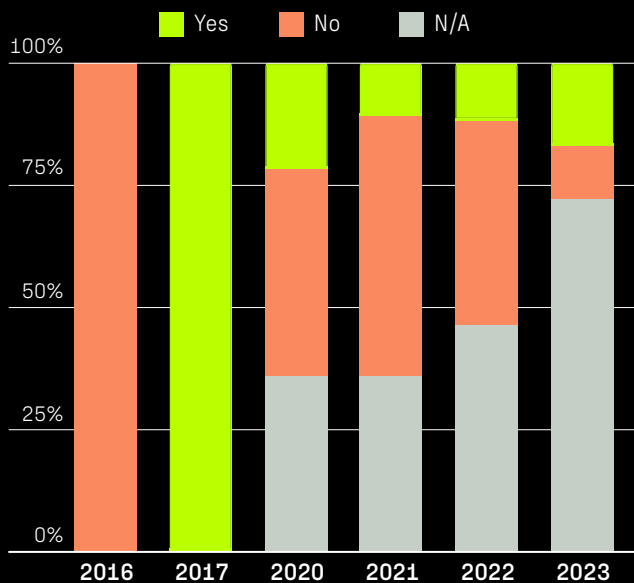


Figure 180: Loss caused per state of audition per year [percentage]

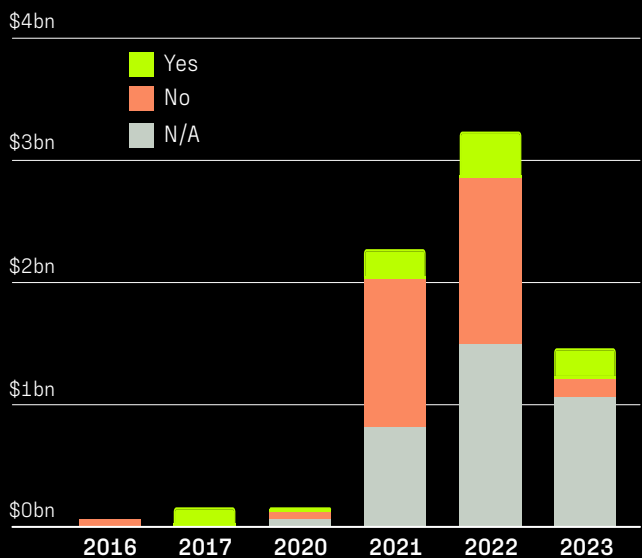


Figure 181: Loss caused per state of audition per year [USD]

Audited Protocols by Chain

If we study whether protocols have been audited by chain (Figures 182 and 183), we can observe that the majority of attacks have been non-smart contract related.

In the case of Avalanche, BSC, Celo and Terra, the majority of the hacked protocols have not been audited. For Ethereum, the number of attacked protocols not audited is bigger than those that were (33.9% vs 20.3%). Arbitrum and Solana have the same percentage of audited and unaudited hacked protocols, while Mixin’s protocols have been either not audited or N/A. In the case of Fantom, the majority of the attacked protocols were audited. Base and Optimism share the same percentage between audited and N/A. The rest of the chains have a majority of N/A protocols. It is noticeable how some big chains by TVL like Arbitrum, Avalanche, BSC or Solana seem to have a high number of protocols attacked that were not audited.

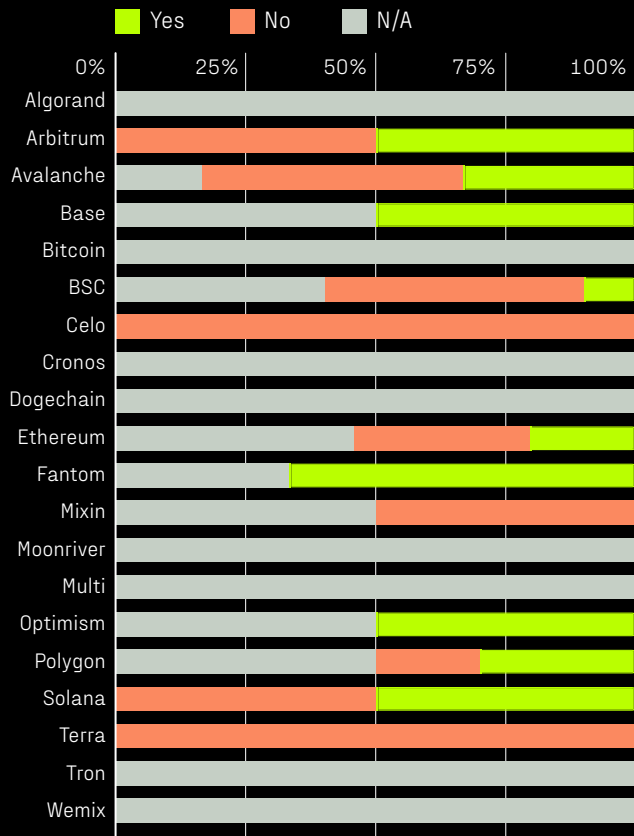


Figure 182: State of audition per chain [percentage]

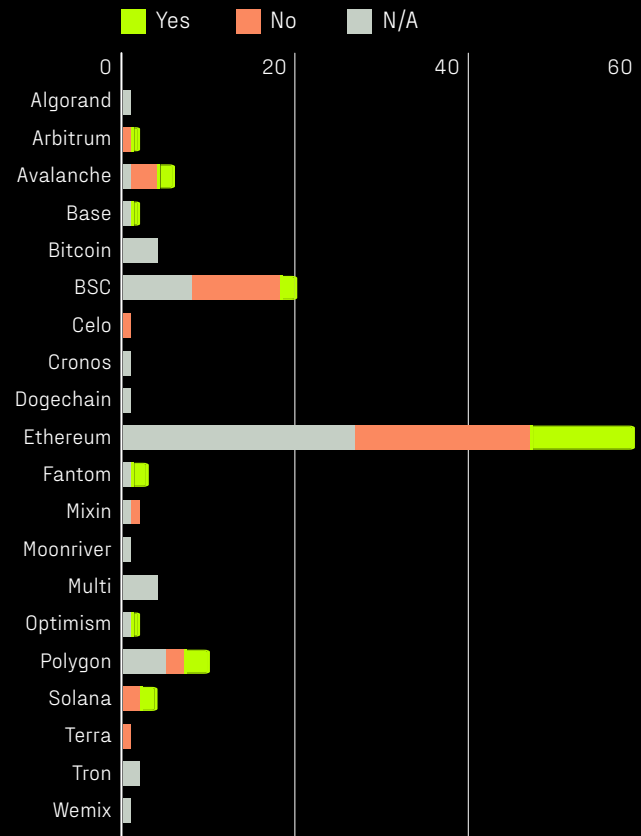


Figure 183: State of audition per chain [count]

Looking at Figures 184 and 185, we can see that most of the losses on Arbitrum, BSC, Celo, Polygon, and Terra were in unaudited protocols.

In the case of Arbitrum, BSC and Polygon, their percentage in losses is higher than their rate of occurrence, especially in Polygon’s case. In the case of Avalanche, Solana, Base, and Optimism they were caused by audited protocols, having their rate of occurrence lower than their percentage by amount lost . In the other chains, the majority of the loss of funds have been caused by attacks not preventable with smart contract audits.

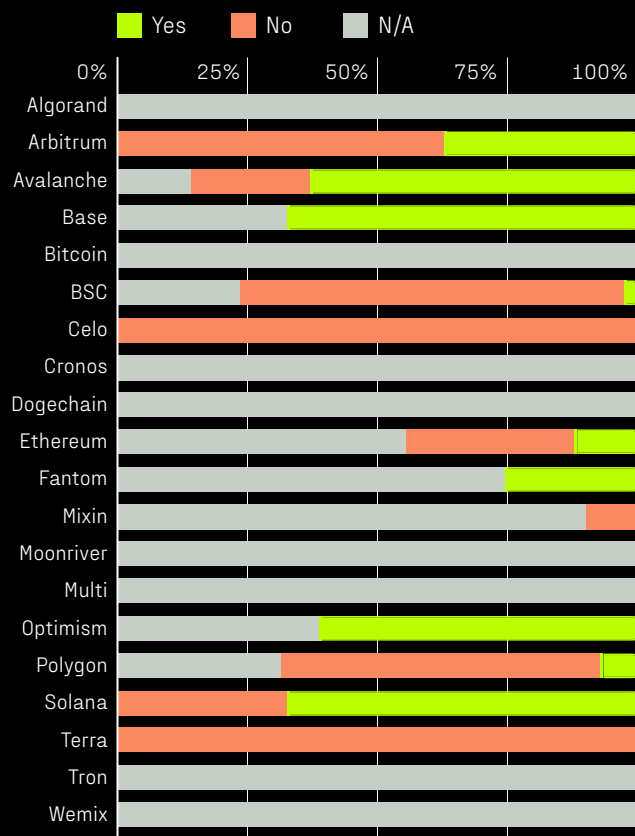


Figure 184: Loss caused per state of audition per chain [percentage]

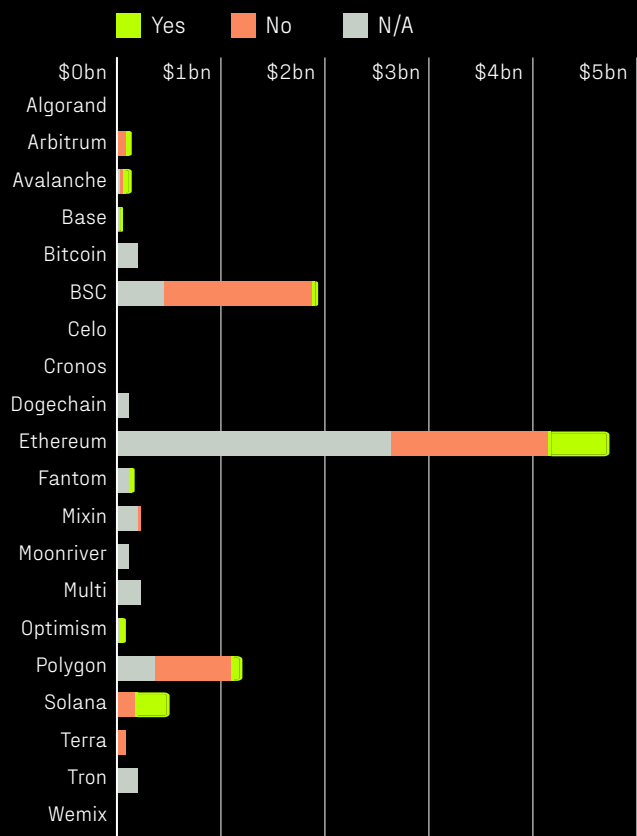


Figure 185: Loss caused per state of audition per chain [USD]

Figure 186 and Figure 187 show the evolution of occurrences of attack in audited protocols over the years.

We can observe that, in the case of Arbitrum, it seems to change between unaudited and audited. BSC seems to evolve towards N/A and unaudited protocols. Ethereum also seemed to follow this trend until 2023, when the percentage of audited protocols increased slightly. Optimism seems to evolve from N/A to audited, but the sample is small. Polygon seems to evolve to be more or less in equilibrium among the three categories.

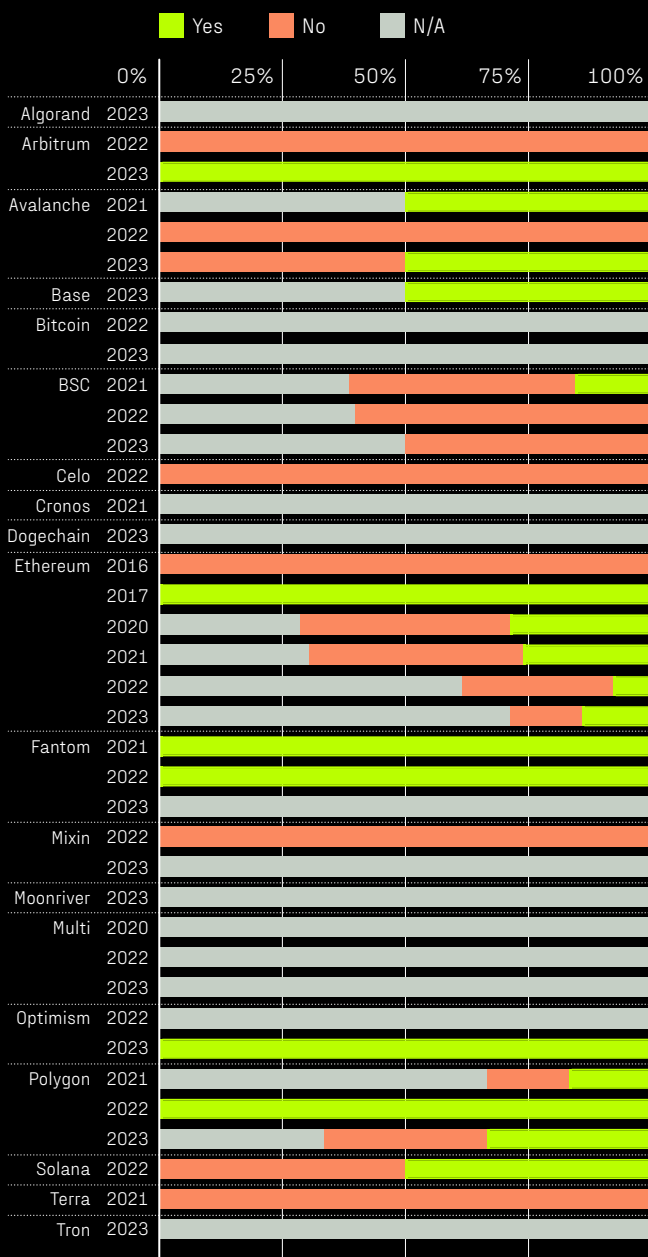


Figure 186: State of audition per chain and year [percentage]

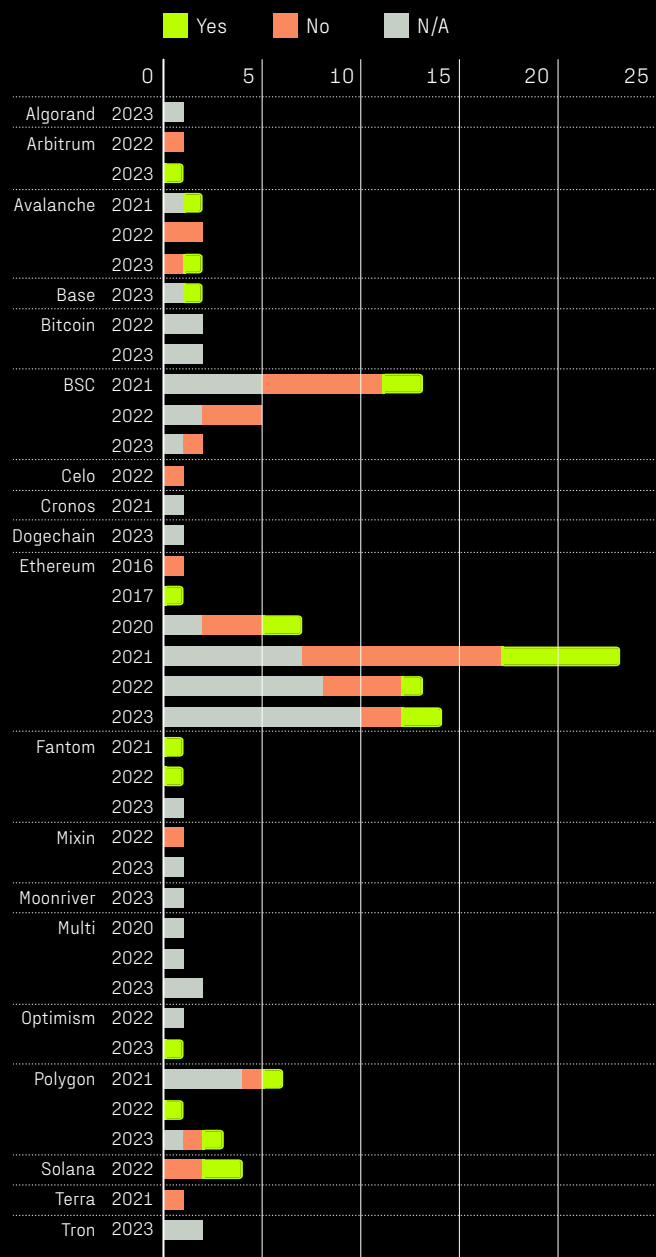


Figure 187: State of audition per chain and year [count]

If we check by loss (Figures 188 and 189), the distribution is pretty similar to the previous figures.

Some major differences are, for example, that the vast majority of losses on BSC in 2023 are because of N/A attacks (65.9% of loss versus 50% of occurrence), which seems to be very profitable for the hackers. On Ethereum, however, in 2023, attacks against audited protocols seem to be more devastating than those for unaudited ones and in comparison with their occurrence (33.1% of the loss versus 14.3% of occurrences). On Polygon, however, the opposite occurs, unaudited protocols seem to cause more damage to those that were audited (91.8% versus 33.3% by rate of occurrence).

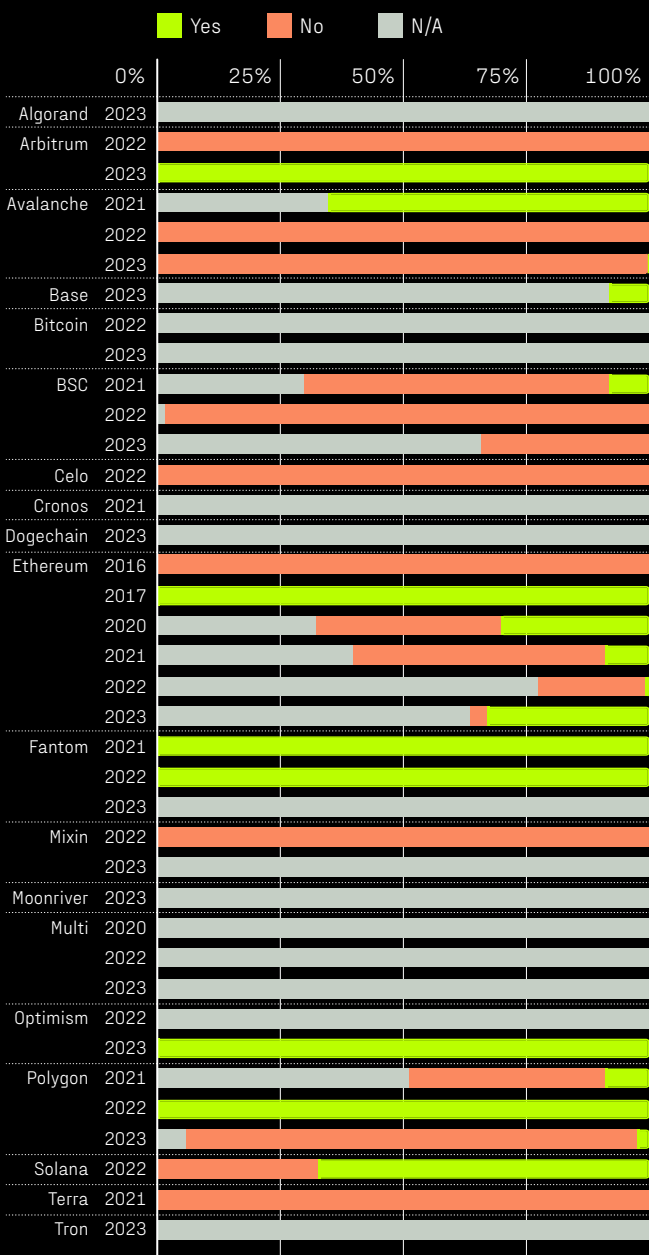


Figure 188: Loss caused per state of audition per chain and year [percentage]

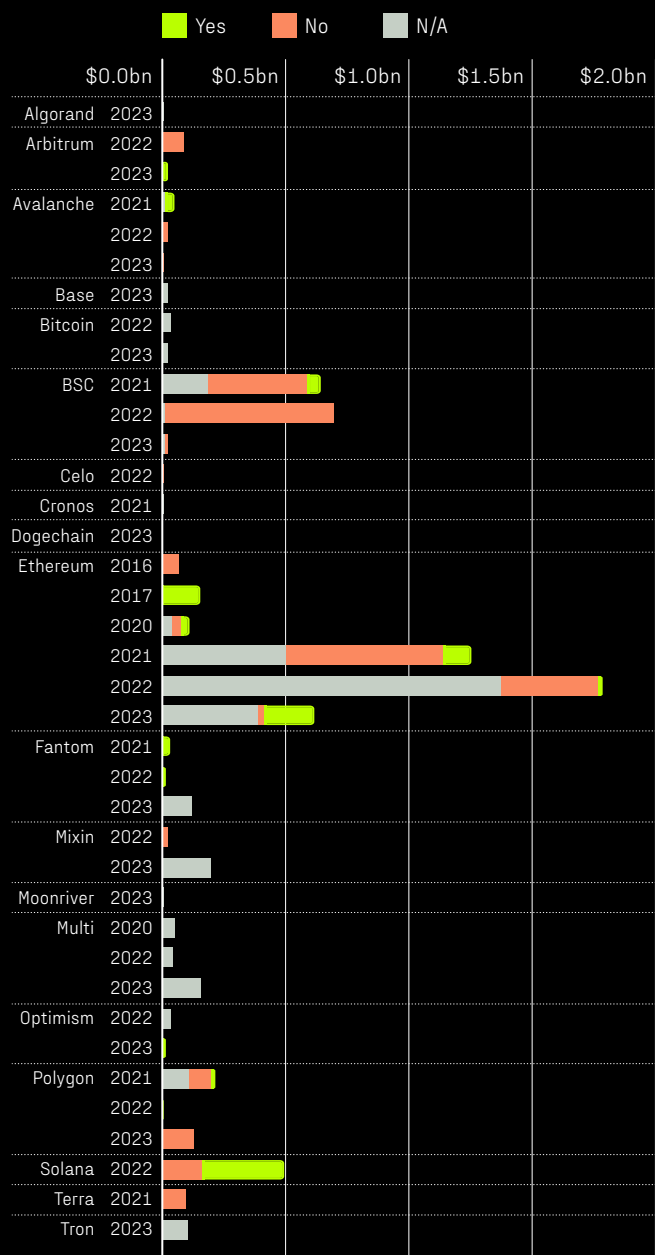


Figure 189: Loss caused per state of audition per chain and year [USD]

Audited Protocols by Type of Attack

Figures 190 and 191 show whether the attacked protocols were audited according to the type of attack perpetrated.

We can observe that only those types related to smart contracts have been considered as audited or not audited, while the rest are N/A. There are two cases of **direct contract exploitation** that we have considered as N/A: the Wintermute multi-sig deployment hack, in which the hacker took advantage of how Gnosis Safe proxies are deployed, and the Curve Vyper vulnerability. The first one used something external to the protocol and the hack was possible because of human error, so it would be out of scope of a normal audit. The second one was really a compiler bug and, therefore, would also probably be out of scope for a smart contract audit. **Rug pulls** are also considered N/A.

In the case of **direct contract exploitation**, the vast majority of the hacks (63.6%) are of unaudited protocols. For **price manipulation** attacks, they are still the majority but to a lesser degree, 63%. The **governance** attack present in the sample was also of an unaudited protocol.



Figure 190: State of audition per type of attack [percentage]



Figure 191: State of audition per type of attack [count]

If we compare which type of attack produced the biggest losses, we can observe in **Figures 192 and 193** that, in the case of **direct contract exploitation**, unaudited protocols are responsible for **70.8% of total losses (\$2,118,350,000.00 USD)**.

Compared to their rate of occurrence, it seems that they cause, in general, more damage than those attacks against audited protocols. The same happens with **price manipulation attacks**, making up 67.5% (\$571,100,000.00 USD) of losses. Still, the difference between the loss and rate of occurrence percentage values is relatively small.



Figure 192: Loss caused per state of audition per type of attack [percentage]

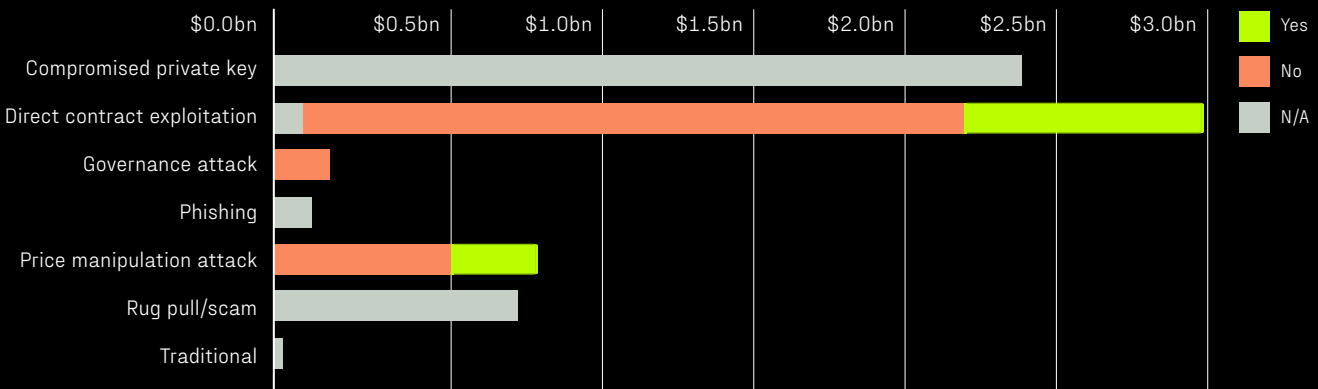


Figure 193: Loss caused per state of audition per type of attack [USD]

By year, we can observe in **Figures 194 and 195** that **attacks on audited protocols via direct contract exploitation** from 2020 to 2023 are in a similar range (around 25% to 35%); however, there has been a slight increase in recent years.

Regarding **price manipulation attacks**, there is a huge reduction in attacked protocols that have been audited from 2020 to 2023, from 50% to 20%. This could be because of the popularization of this type of attack and the improved knowledge that professionals have on how to prevent them.

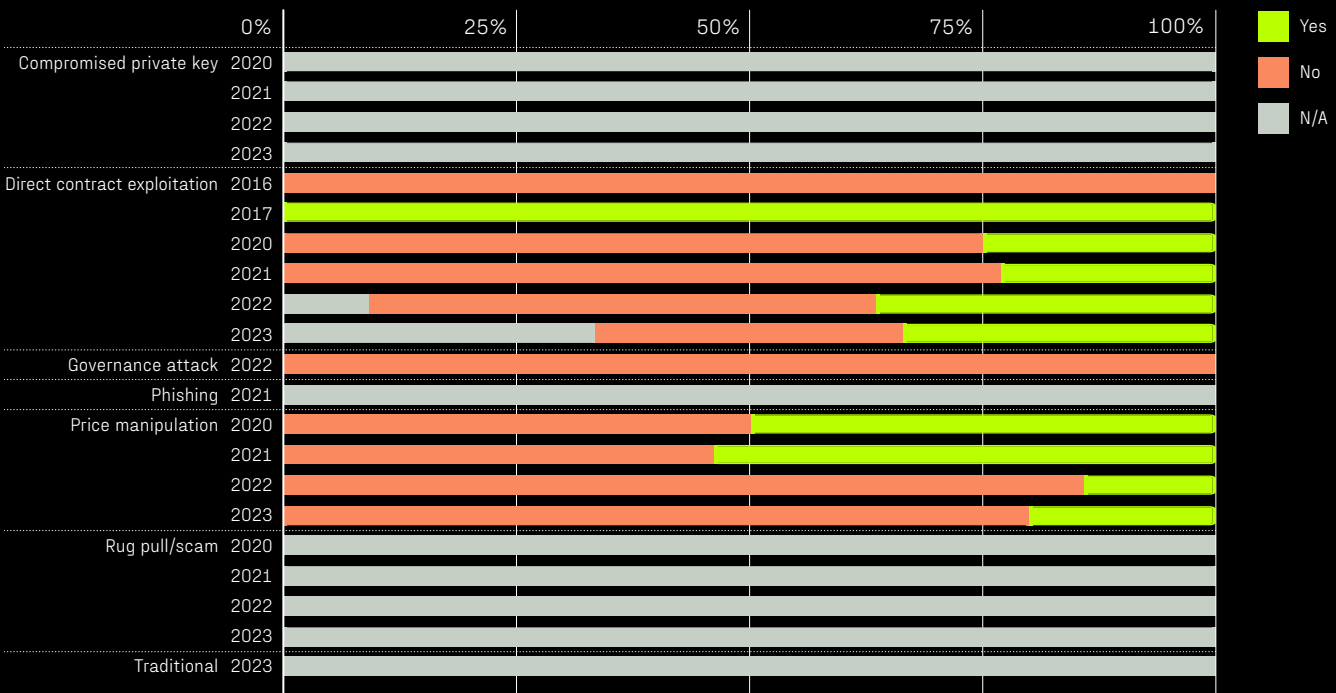


Figure 194: State of audit per type of attack and year [percentage]



Figure 195: State of audit per type of attack and year [count]

If we check the amount lost per year and types of attack in **Figures 196 and 197**, we can see that, in 2023 **direct contract exploitation**, attacks against audited protocols have caused a lot of stolen funds when compared with their rate of occurrence (**71.7%, \$197,000,000.00 USD**).

This underscores the importance of a good audit of the smart contract. In the case of **price manipulation attacks**, the funds lost by audited protocols are a bit higher, (24.1%, \$47,523,000.00 USD) but, in general, seem to be more or less proportional.

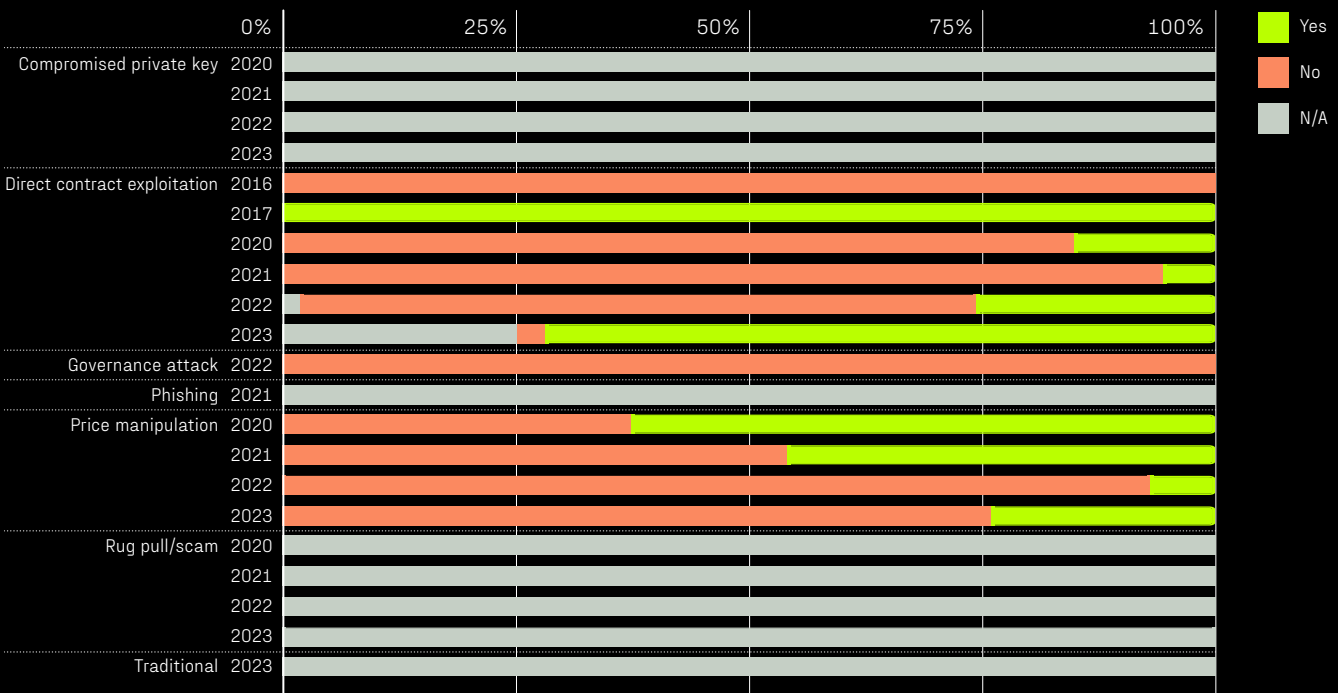


Figure 196: Loss caused per state of audition per type of attack and year [percentage]

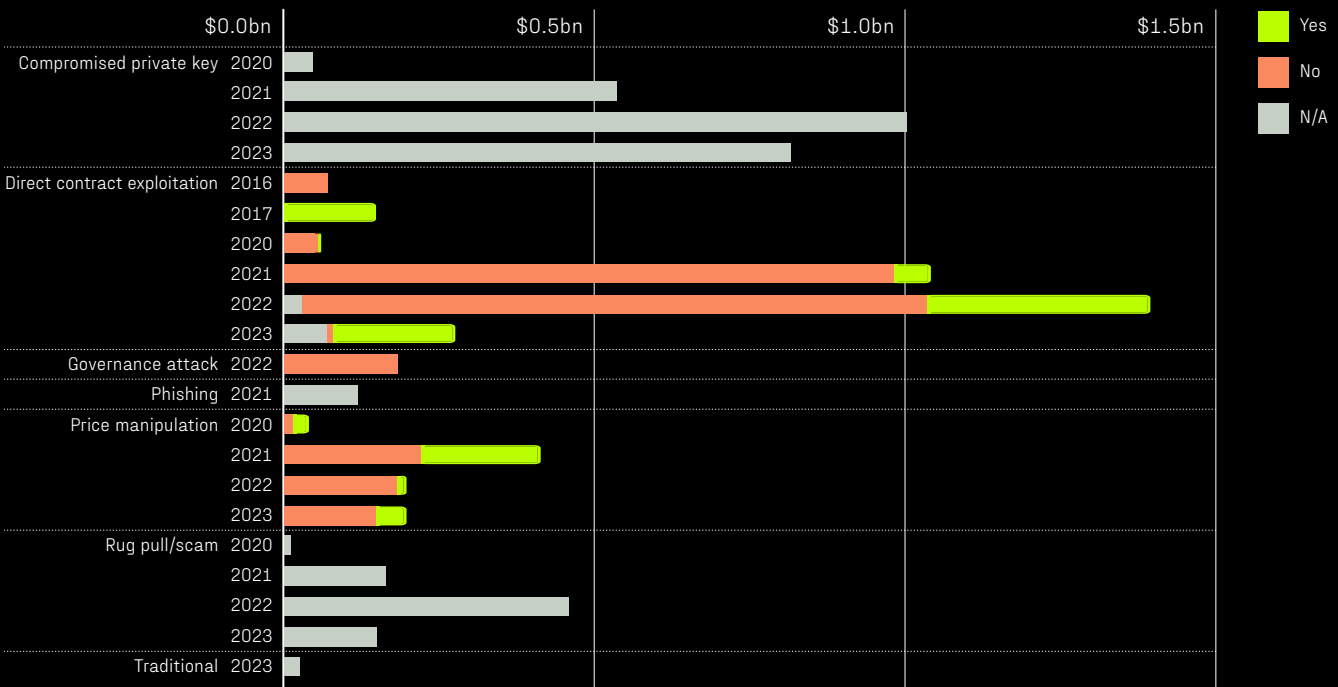


Figure 197: Loss caused per state of audition per type of attack and year [USD]

Audited Protocols by Type of Protocol

There are some types of protocols that have been attacked despite being audited, for example **Insurance**, **Launchpads** and the majority of the **Liquidity Managers** and **Yield** protocols.

For **Bridges**, **CEXs**, **DEXes**, **Gaming**, **NFT Marketplace**, **Reserve Currencies** and **Wallets**, a traditional audit would probably not have prevented their attack. However, it is noticeable that the majority of **Algo-Stables**, **CDPs**, **Derivatives**, **DEX Aggregators**, **Indexes**, **Lending** protocols and **Yield Aggregators** attacked have not been audited (see Figures 198 and 199).

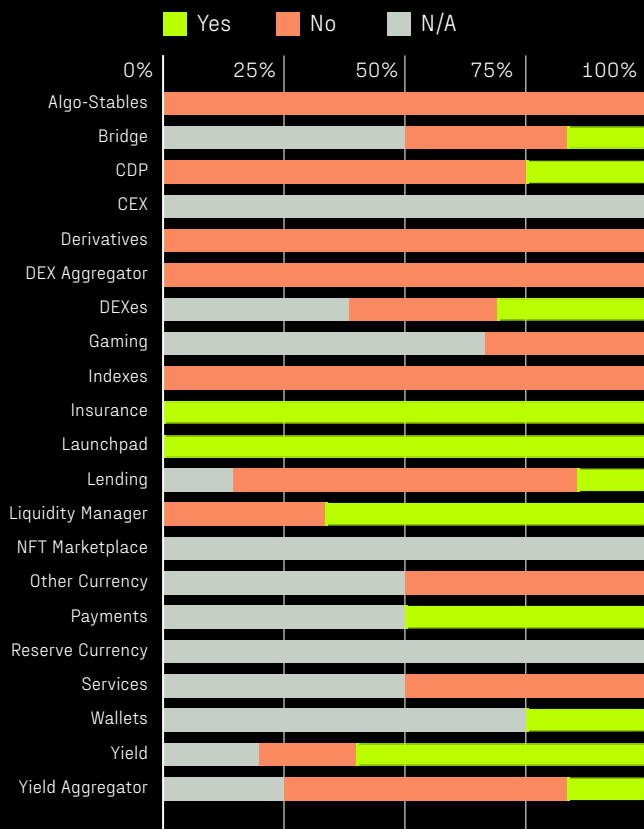


Figure 198: State of audition per type of attack [percentage]

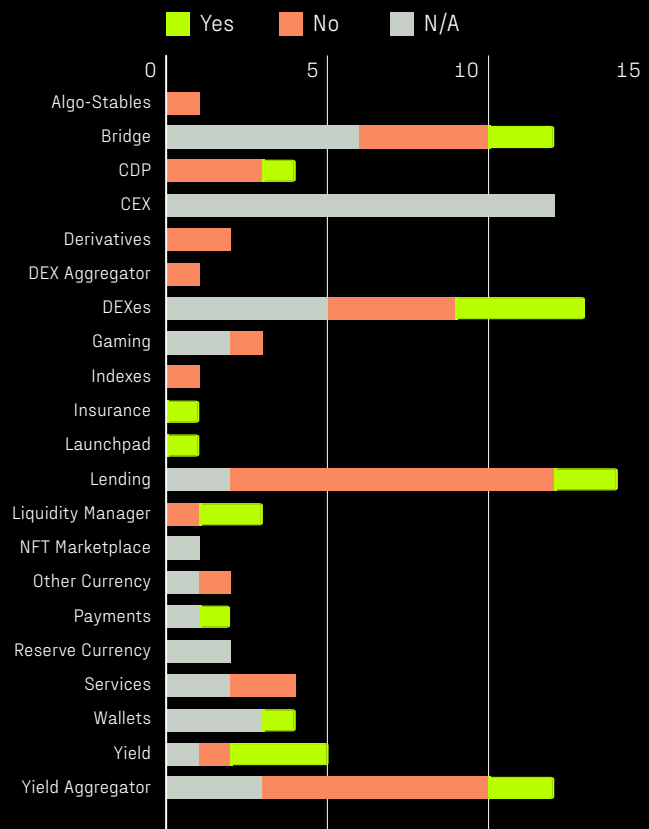


Figure 199: State of audition per type of attack [count]

If we observe the monetary losses in each case, we can observe in **Figure 200** and **Figure 201** that, in general, it is pretty similar to the occurrence, with some exceptions where issues that would not be covered by a smart contract audit accumulate a higher percentage of loss than of occurrence, for example, are the **Other currencies category, Gaming, and DEXes**.

It should also be taken into account that in some cases, like **Bridges, CDPs or DEXes**, the losses incurred by audited protocols seem to be lower on average than for other types of audit status. The opposite happens, however, in **Yield and Lending** protocols and **Liquidity Managers**.

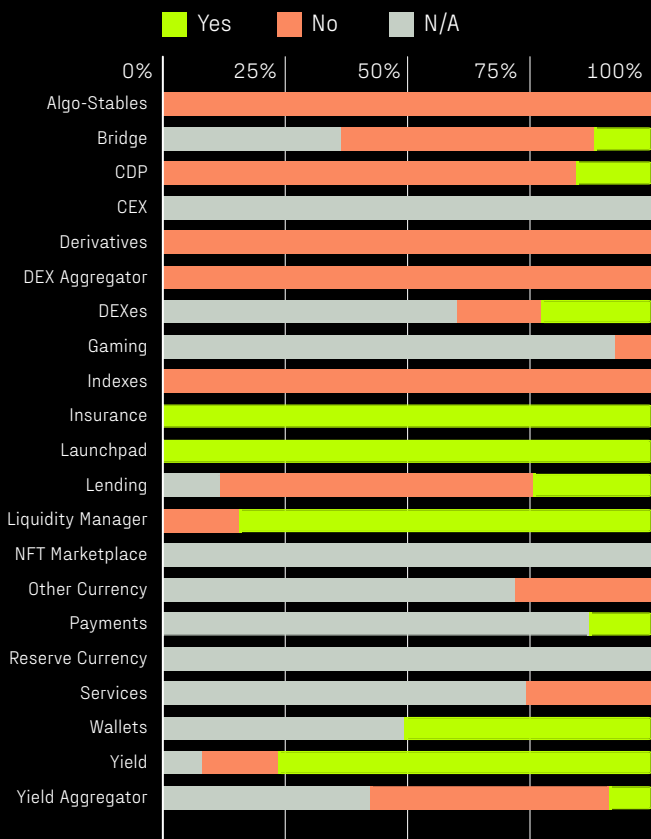


Figure 200: Loss caused per state of audition per type of attack [percentage]

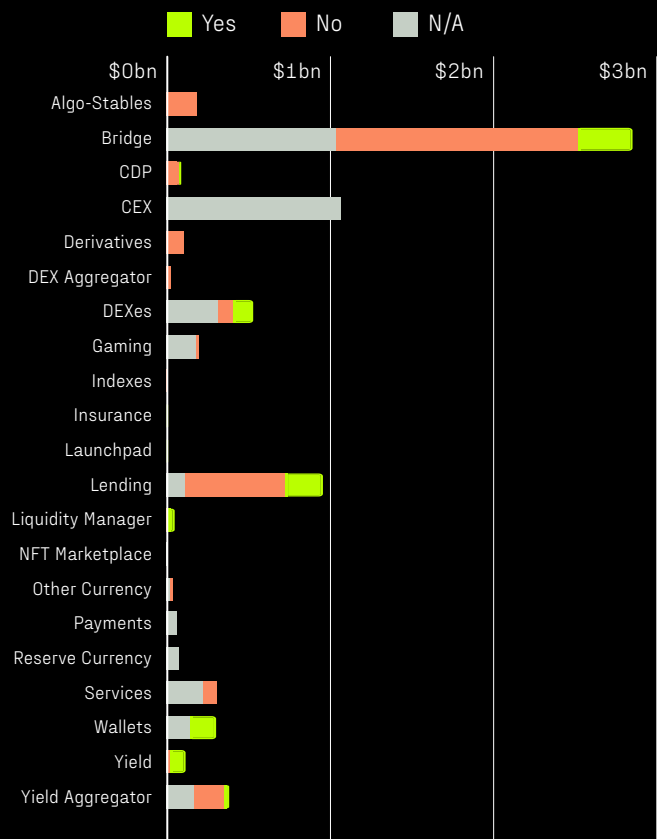


Figure 201: Loss caused per state of audition per type of attack [USD]

By year, we can see in Figures 202 and 203, how the number of attacks in audited protocols for some categories have decreased.

This happens, for example, in **Bridges, CDP, Payments, Wallets, Yield** and **Yield Aggregators**. A noticeable case is **Lending** protocols, in which they have increased up to 50% in 2023, however, it is also true that the sample for this year is small.



Figure 202: State of audition per type of attack and year [percentage]

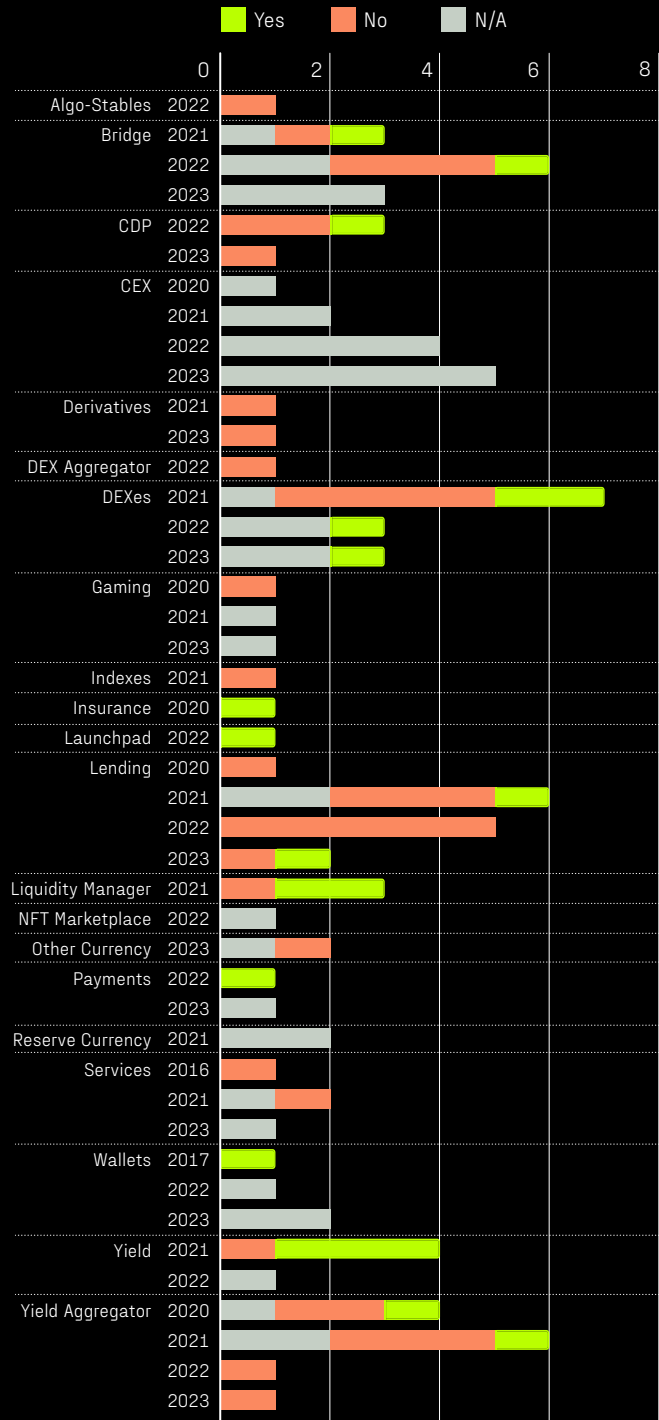


Figure 203: State of audition per type of attack and year [count]

If we compare the loss by type of protocol and whether they were audited or not, we can observe that the loss in **Yield Aggregator** protocols decreased greatly in 2021 for those protocols that were audited.

This is also true for **Bridges** and **Lending** protocols in 2021 and **DEXes** in 2022. In general, the values don't seem to show anything different than in previous charts (Figures 204 and 205).



Figure 204: Loss caused per state of audition per type of attack and year [percentage]



Figure 205: Loss caused per state of audition per type of attack and year [USD]

Audited Protocols by Type of Function

In most cases, the functions attacked have largely not been audited.

Especially noticeable are the cases of `initialize`, `mint` and `swap`, where (with a rather large sample) 75% or more of them were not audited (see Figure 206 and 207). `deposit`, `migrate` and some protocol-specific functions seem to have been hacked despite being audited. The latter two probably entails a deeper knowledge of the protocol being audited in order to both audit and exploit them.

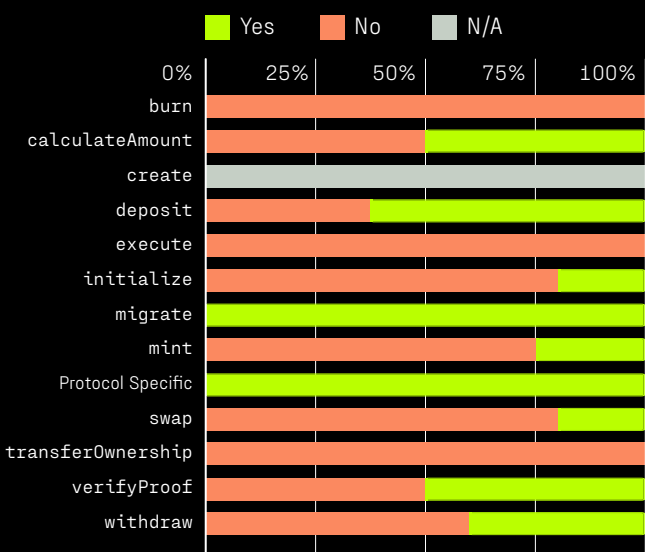


Figure 206: State of audition per type of function [percentage]

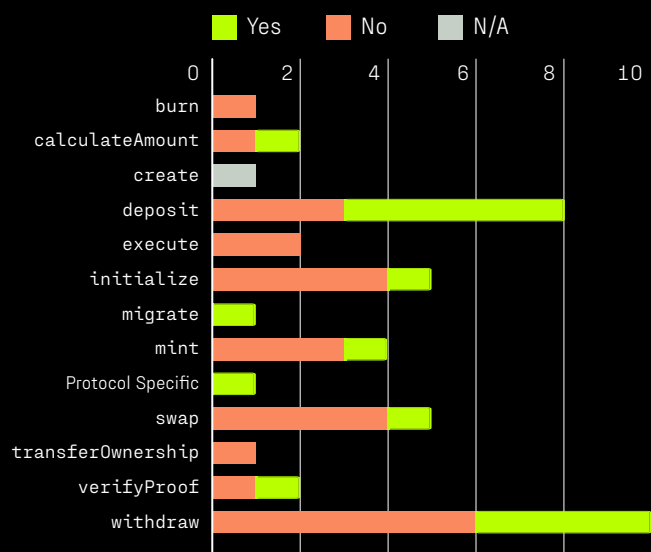


Figure 207: State of audition per type of function [count]

With regard to losses incurred, we can observe in Figures 208 and 209 that `mint`, `withdraw` and `verifyProof` functions that were audited produced less loss than those that weren't. The opposite happens, however, for `calculateAmount`, `initialize`, `deposit` (slightly) and `swap` (also slightly).

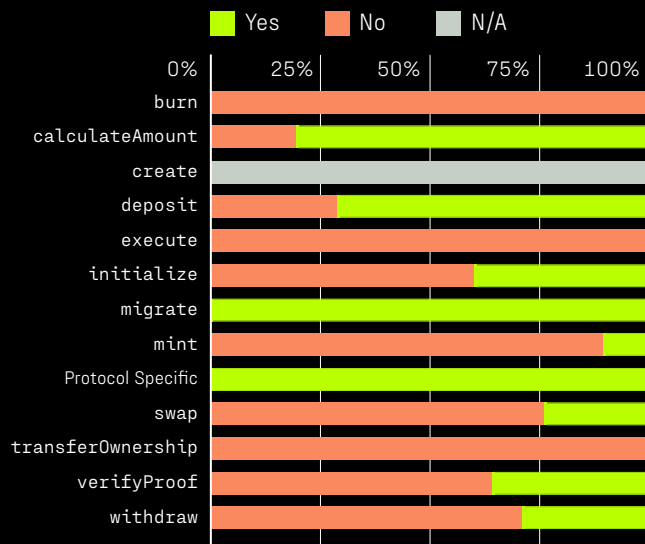


Figure 208: Loss caused per state of audition per type of function [percentage]



Figure 209: Loss caused per state of audition per type of function [USD]

If we compare these results by year, we can observe in **Figures 210 and 211** that, for all functions except for **swap**, **deposit** and **calculateAmount**, in 2023 the attacks on all of them were not of audited protocols.

It should be noted, however, that, in some of them like **deposit** or **mint**, the number of audited protocols hacked was larger in 2021 than in the previous year.

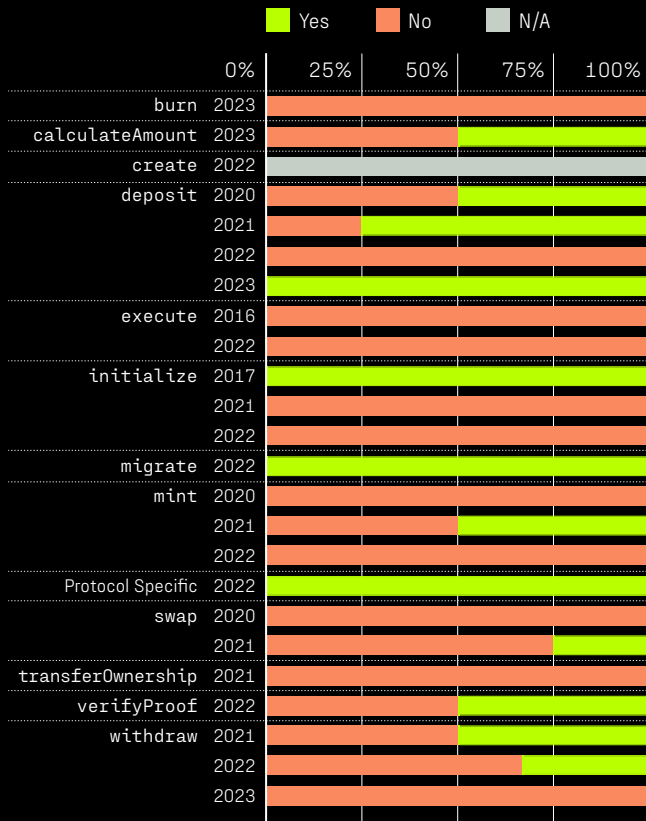


Figure 210: State of audition per type of function and year [percentage]

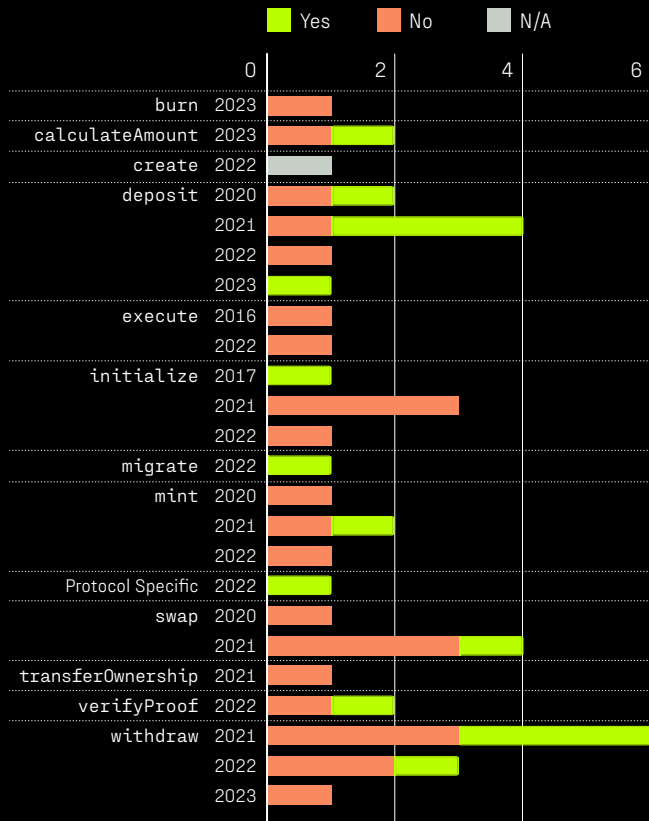


Figure 211: State of audition per type of function and year [count]

By year and loss (Figures 212 and 213), the situation is very similar to the previous charts (Figures 210 and 211).

We are able to observe how those hacks on `deposit` for 2021 actually produced more damage than their not audited counterparts. On the other hand, `mint` in 2021, `verifyProof` in 2022, and `withdraw` in 2022 produced less lost funds.

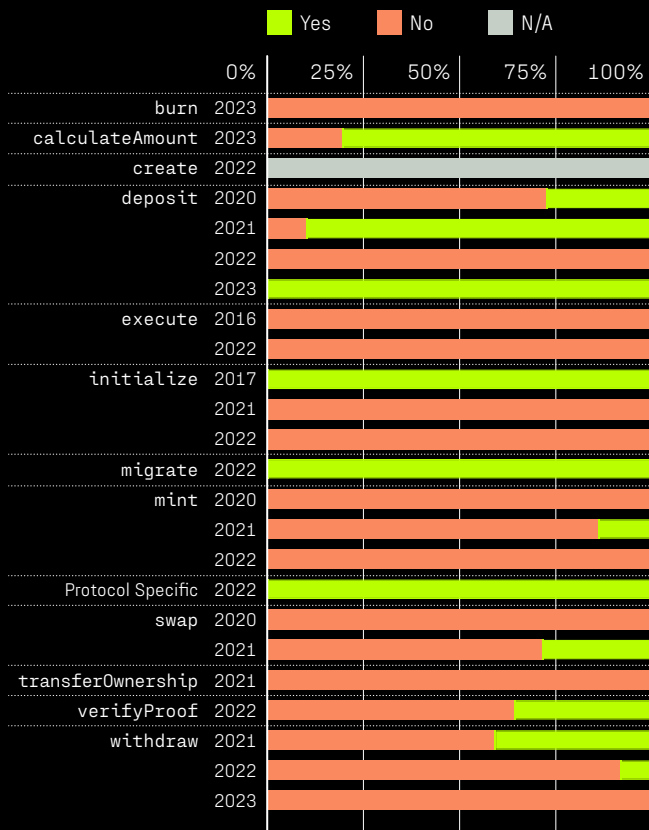


Figure 212: Loss caused per state of audition per type of function and year [percentage]

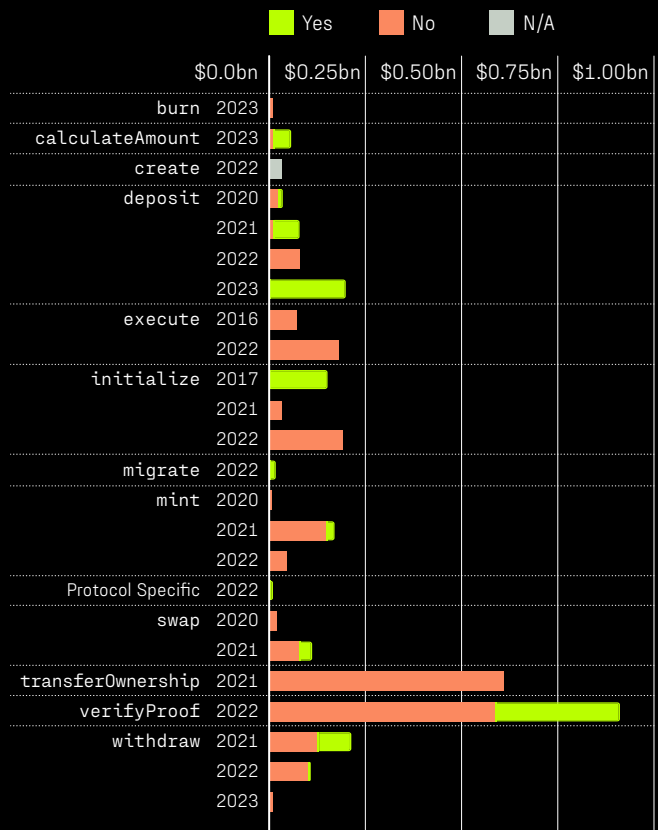


Figure 213: Loss caused per state of audition per type of function and year [USD]

ACTIONABLE TAKEAWAYS

According with the data analyzed and key findings extracted (recall **Section 2**) the following actions are recommended:

- **Audit your code but don't forget to take into account the whole ecosystem and traditional security audits.** From a developer perspective, audit your smart contracts and protect the private keys and the system in which they are contained. From a user perspective, look for protocols that perform complete audits and not only smart contracts ones.
- **Consider using multi-signature or MPC and cold wallets.** Using a cold wallet reduces the chance of them being stolen. Furthermore, using multi-signature or MPC wallets or schemes to perform permissioned and administrative actions on the protocol could minimize the chance of a key being compromised.
- **Beware of flash loans.** Program the protocol taking them into account, for example, by using snapshots to calculate exchange prices and voting power.
- **Avoid flawed oracles.** Use reputable, multi-source, decentralized and incentive-driven oracles like Chainlink and consider having a backup one.
- **Be careful with Lending protocols, Bridges and CEXs.** As a protocol owner, consider potential direct contract exploitation and possible price manipulations when programming a lending protocol. In the case of bridges, also review the code carefully to avoid possible attacks and secure administrative keys. If you are programming a CEX, make sure that all keys relevant to the protocol, especially those controlling funds, are secure. As a user, be careful with these types of protocols; make sure they were properly audited and consider using alternatives (like DEXes) instead.
- **Pay special attention when programming functions whose functionality coincides with those defined in Section 8.** especially withdrawals, deposits, transfer of ownership and proof verification. Also pay attention to the code in order to avoid mistakes that could result in the vulnerabilities explained in Section 5.
- **Decentralize.** Decentralized protocols have been less targeted by attacks and incur lower losses. If you are a user, look for fully decentralized protocols.