BREAKING DOWN THE TOP 50 DEFI HACKS

2016-2022 COMPREHENSIVE REPORT

HALBORN

TABLE OF CONTENTS

INTRODUCTION

DeFi hacks are more common each day, causing losses of millions of dollars. It is estimated that protocols lost $3.9 Billion in 2022 [1].

In this report, we intend to provide a comprehensive review of the top 50 hacks in DeFi history until 2022. 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-htm

KEY FINDINGS

After this study we present a series of key findings to summarize all the data extracted:

- Hacks lead to major losses, and they increase every year. The total amount lost by the top 50 largest hacks accumulates to a total of $5,564,100,000 USD. Furthermore, in recent years the amount seems to be higher than in the previous one, as losses on 2022 are about $1B higher than in 2021 and these are around $2B higher than in 2020.
- Solana, more vulnerable than most. Even though Ethereum is the most attacked chain, it is also the largest of all of them. Thus, it has more protocols to attack. However, Solana accumulates more hacks than it should according to its TVL. Furthermore, it is also one of those chains that accumulates more hacks in their smart contracts despite being audited, and the main cause of attacks is the exploitation of those smart contracts. This could also be because of the complexity of the language in which they are written (Rust) in comparison with Solidity. Fantom presents a similar case; however, the sample studied (number of hacks) is less significant, and the difference in the expected number of hacks by TVL is lower.
- Audits on the code and the whole ecosystem are necessary. The majority of protocols attacked had unaudited smart contracts. However, some attacks, like price manipulations, are hard to find in audits if the whole ecosystem and how the protocol interacts with it are not considered. Furthermore, private key leakage or theft is the second most common cause of attacks, and is not a threat that can be detected by a smart contract audit.
- Bad logic and incorrect or missing input validation are the main causes of hacks in contracts.
- A failure to use multi-signature, MPC, and cold wallets is another common issue. All of the vulnerable keys were stored in hot wallets and only a small percentage used multi-sig wallets.
- Flash loans can be a means of attack. Consider them a possibility if your protocol allows swapping and exchange of assets or uses token quorum power in governance processes.
- Bad oracles can be dangerous. More than a quarter of price manipulation attacks were possible thanks to the use of a bad oracle by the protocol.
- Lending protocols, Bridges and CEXs are the most insecure type of protocols. Those protocols accumulate the highest number of attacks in proportion with the total of protocols available.
- Functions used to ‘withdraw‘, ‘mint‘, ‘swap‘, ‘deposit‘ or ‘calculatePrice‘ of assets; ‘transfer Ownership'; 'initialize' or upgrade contracts; and verify proofs of actions are especially vulnerable.

TIME DISTRIBUTION & AMOUNT LOST

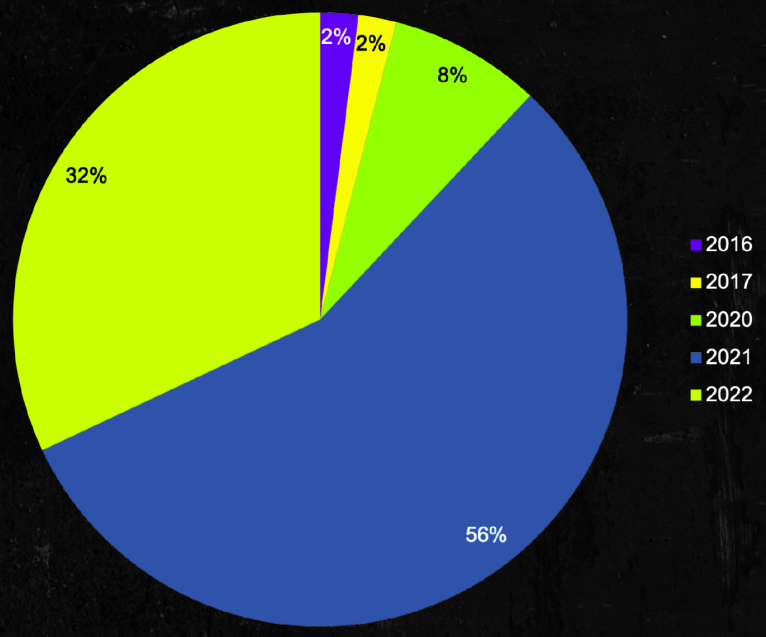

Figure 1: Distribution per year

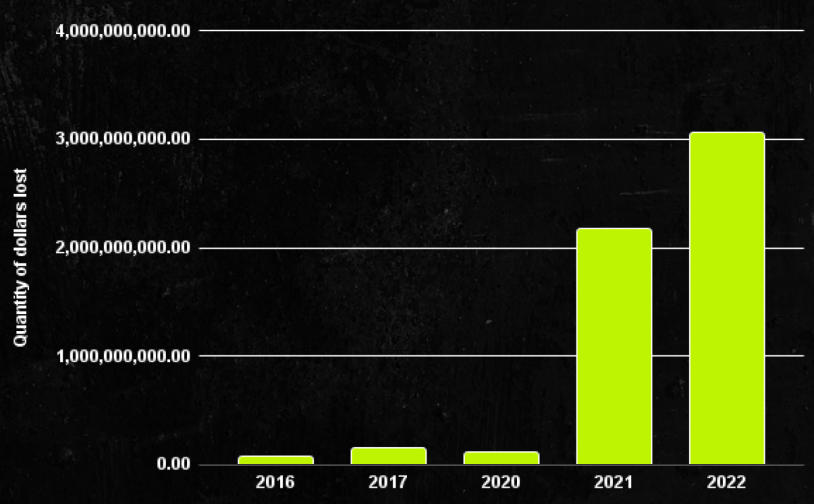

Figure 2: Amount per year (in dollars)

DeFi protocol use and development have increased through the years. However, the biggest losses are not distributed in a constantly increasing way. Figure 1 shows the number of hacks per year. The earliest one corresponds to The DAO hack in June 2016. It can be seen that the number of hacks or other kind of attacks is bigger in 2021 than 2022; however, that does not indicate that the total number of hacks have decreased, but that those were less severe.

Regarding the amount lost by these 50 top hacks, the total amount lost is around 5,564,100,000 USD. By year, we can see that, even if the number of hacks in 2021 was higher (Figure 1), the amount of assets lost in 2022 is about 1B higher than in 2021 (Figure 2). This indicates that the amount lost per hack seems to be getting bigger and bigger as time passes. It should be noted that 2018 and 2019 do not appear on the dataset. This is because the attacks on those years were not big enough in amount of losses to be in the top 50 ranking and not because there were no attacks whatsoever.

ETHEREUM, THE MAIN TARGET BUT NOT ALONE

Figure 3 shows the distribution of the different hacks per chain. If an attack has been carried out in more than one chain is counted in each one. It is noticeable that almost 48% are on the Ethereum network. Indeed, this chain has been identified as the biggest by TVL. Another interesting insight extracted on this study is that Binance Smart Chain (BSC) was used in almost 22% of cases. It makes sense considering that BSC is the second largest blockchain by TVL as of Feb 2023 [2]. However, it should be noted that, while the number of hacks seems to follow, to an extent, the relation of largest chains therefore largest hacks, there are some cases that stand out. Solana, while being the 9th biggest chain by total TVL, behind Avalanche, Arbitrum, Optimism, and Fantom, still surpasses them in number of hacks, accumulating nearly 6% of them. Bitcoin and Terra also appear in the ranking without even being in the top 10 by TVL. On a positive note, Arbitrum, even though is the third chain by TVL, is one of the least attacked ones. Figure 4 shows a comparison between the order of chains by TVL against their order by number of attacks. If the yellow line is below the purple one, it means that the order by number of hacks is higher than would be expected based on TVL and, therefore, may imply more vulnerable chains.

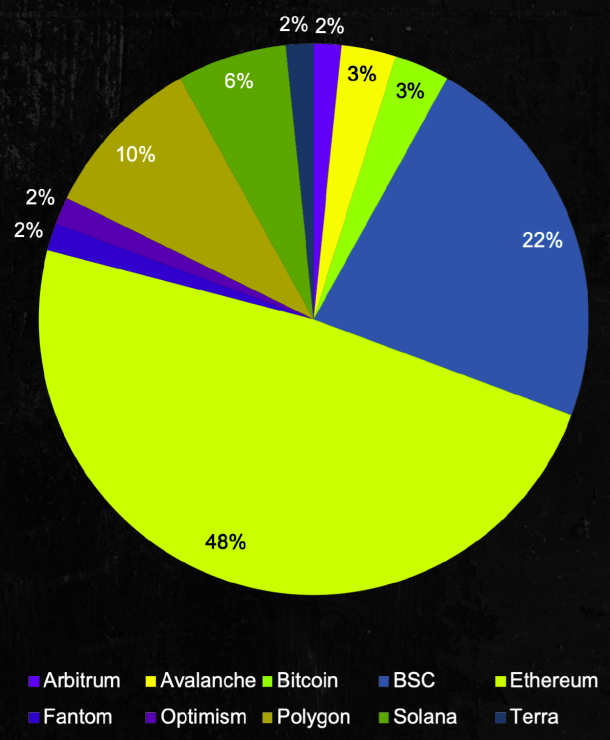

Figure 3: Distribution of attacks per chain

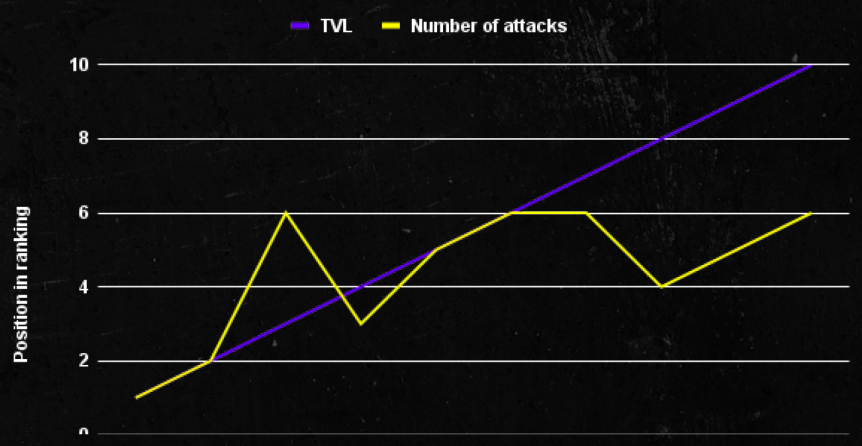

Figure 4: Order by TVL and number of attacks.

2 https://coinmarketcap.com/chain-ranking/

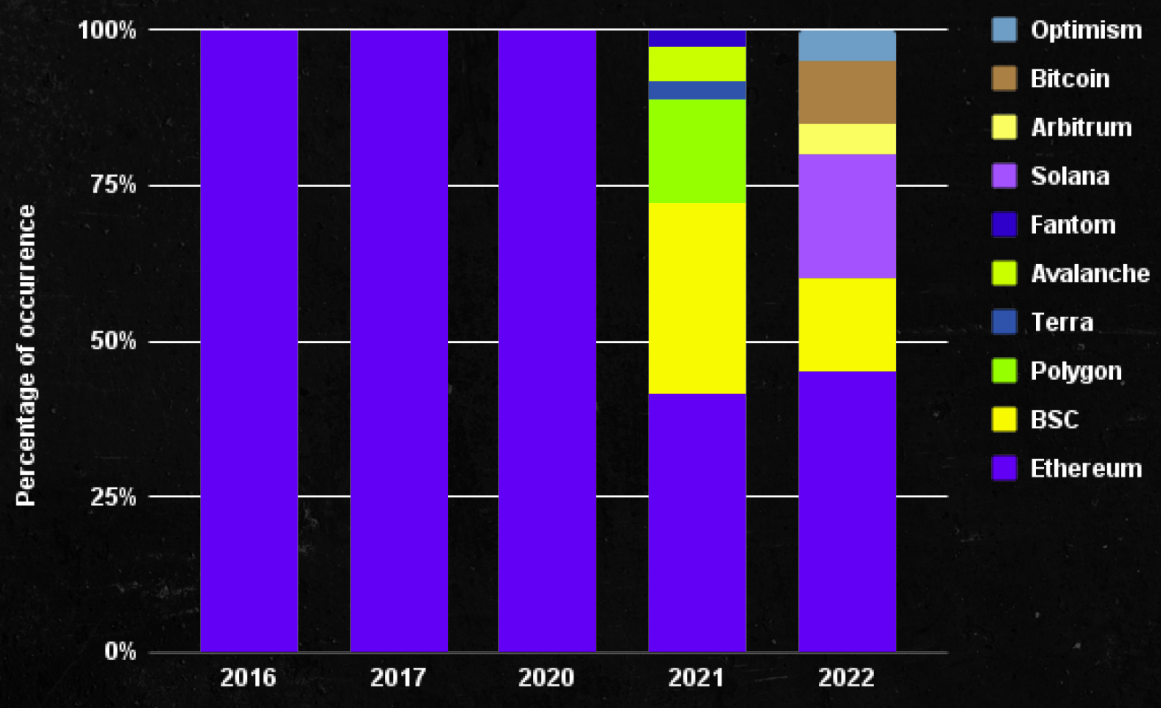

Figure 5: Distribution of hacks by chain per year 16

Figure 5 shows the distribution of hacks by chain per year. It should be noticed that, for example, BSC and Solana were introduced in 2020.

The trend seems to indicate that Ethereum is the most attacked chain, likely because it is also the most used. However, the trend seems to indicate that, when new chains were created and gained market share, attacks became more distributed across Ethereum and these other chains.

TYPE OF ATTACKS

As in the previous section, if an attack fits two different categories, it will be counted in each one of them. For example, a price manipulation attack could be aided by the exploitation of a contract:

- Contract exploitation: An attack will fall into this category when the smart contract code had a vulnerability that was exploited by an attacker. For example, they may have taken advantage of a mathematical error or another common vulnerability like re-entrancy.
- Private key leakage/theft: This attack occurs when a key with sufficient privileges is stolen or leaked, commonly by the use of phishing attacks or by compromising the system in which a wallet is stored.
- Price manipulation: The attacker manipulates the price of assets to take advantage of the protocol. This entails making certain actions (like trades or loans) with the objective of changing the price of an asset to either increase or decrease its value. Changing its value allows the attacker to generate returns in a number of ways, such as selling the assets in greater quantities, creating higher prices that normally wouldn't be possible, and even destabilizing and damaging a protocol (for example, by leaving it with an under-collateralized position). This can be done, for example, by taking advantage of the protocol's use of a bad oracle or faulty logic in the contract code. These concepts will be further explained in Section 5.4.
- Rug pull: Sometimes the creators are the ones stealing from their protocol. A rug pull is defined as a type of scam that happens when the developers steal a protocol's funds.
- Traditional: Other kinds of non-web3 specific attacks. This could include for example script injection.
- Governance: When there is an attack on the governance process of a protocol, for example, passing malicious proposals.

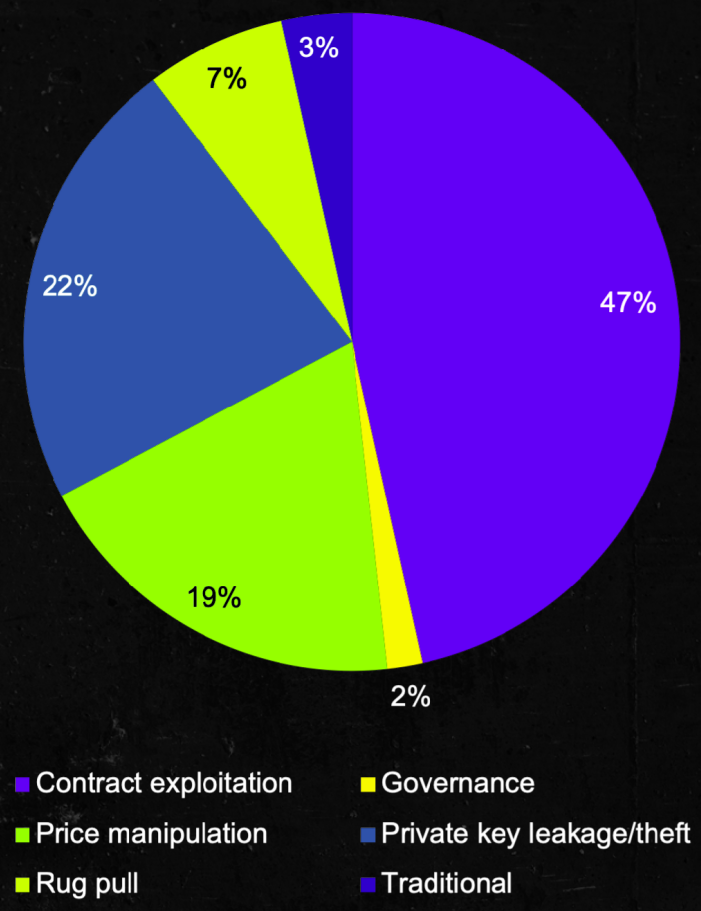

Figure 6: Types of attacks

Figure 6 shows how attacks are distributed with regard to their type. Most attacks are possible because of contract exploitation (nearly 47% of them).

The second most common cause of hacks are related to the theft or leakage of a project's private keys (22%). The third most common is by price manipulation (19%), followed by rug pulls done by project developers. Traditional and governance attacks are present in lesser numbers.

It can be observed (Figure 7) that, while smart contract exploitation is still the most common, other causes like private key theft and price manipulation are gaining popularity, especially in recent years.

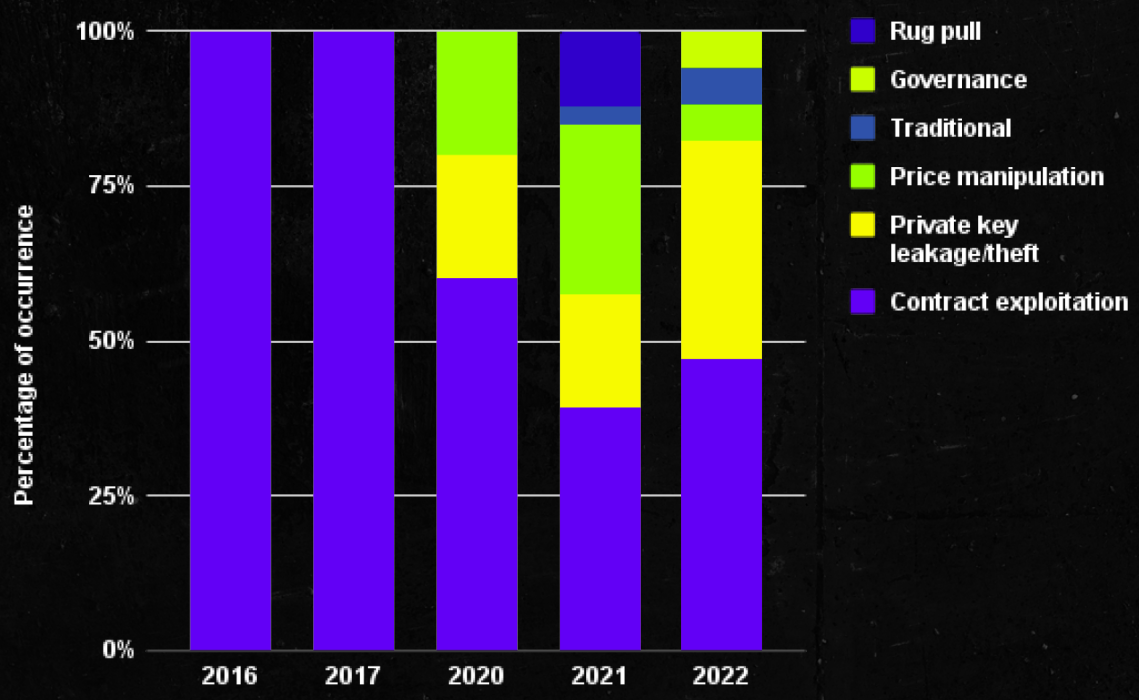

Figure 7: Type of attacks per year

5.1 CONTRACT EXPLOITATION

Contract exploitation is the most common cause of hacks in the current DeFi environment. However, a contract can be exploited in different ways.

- **Math bug:** Math bugs are when an error in a mathematical formula or in the calculation process occurs, such as rounding mistakes.
- **Lack/faulty input verification/validation:** A contract is exploited in this category when there is a missing 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.
- **Re-entrancy:** 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.
- **Incorrect call permissions check:** The caller's ability to execute the function is not properly set. For example, a function that should be executed only by certain roles is left open for anyone to call.
- **Faulty proof verification:** Especially relevant in bridges and other cross-chain protocols, it occurs when there is a faulty proof verification on one chain which allows the attacker to falsify actions on the other paired chain. For example, the signature verification algorithm may be implemented incorrectly.
- **Faulty initialization:** This occurs when a contract is left uninitialized or it is initialized with the wrong arguments. Checking for faulty initialization is important for proxy contracts.
- **EVM based:** This is the exploitation of a contract by taking advantage of how the EVM works. For example, by repeatedly making the contract deploy other contracts until a certain contract address is generated.
- **Bad logic:** Any other kind of programming error that results in the contract's exploitation.

Figure 8 shows the distribution of types of contract exploitation. The two most common are missing or faulty input verification or validation and bad logic, accumulating 26% and 23% of the cases each.

Re-entrancy is the cause in 15% of the cases while math bugs account for 12% of them. The other types appear in lesser percentages. It should be noted that, unlike in other studied parameters, there is no significant leading cause of contract exploits.

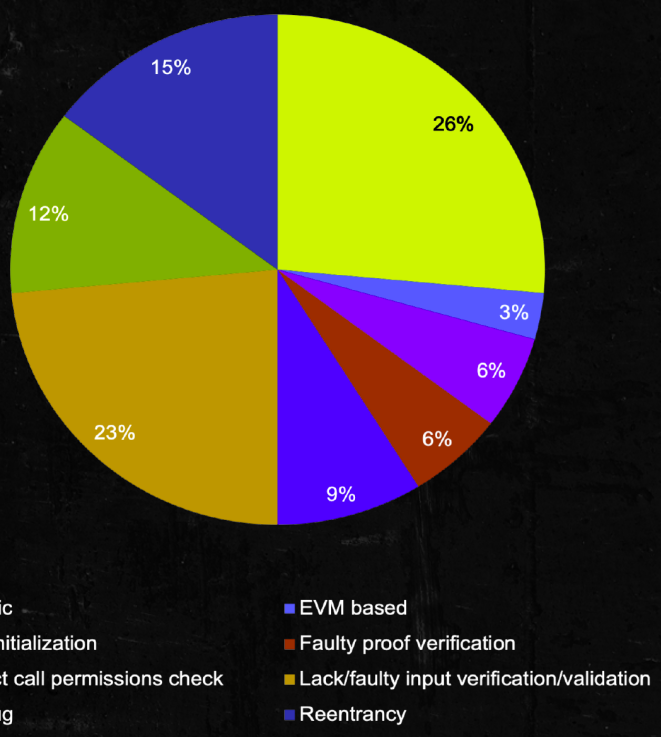

Figure 8: Types of contract exploitation

According to Figure 9, there does not seem to be a clear correlation between the year and type of contract exploitation, except for the case of faulty proof verification, which is only present in 2022, probably because of the popularization of bridges.

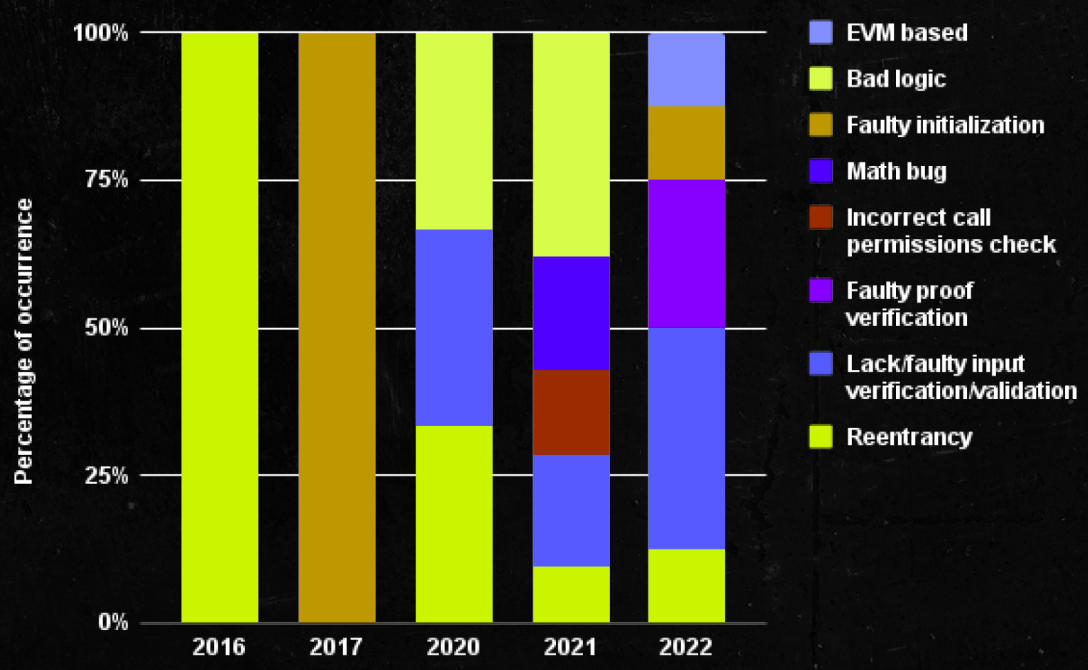

Figure 9: Contract exploitation per year

5.2 WALLET PROTECTION

In Section 5, it has been presented that the second most common cause of attack is theft or leakage of a project's private keys, which allowed attackers to exploit the protocol.

In order to provide a better understanding on how these keys could have been exploited, two things will be analyzed. First, if the vulnerable key was part of a multi-sig or multi-signature. This entails that, in order to execute a transaction, it needs to have two or more signatures. Multi-sig provides more security than single-signature transactions because more keys need to be compromised in order to damage the protocol. According to our research, only 17% of the attacked addresses were part of a multi-sig (Figure 10).

Another security measure is to use cold wallets instead of hot wallets. Hot wallets are connected to the internet, while cold wallets utilize private keys kept offline. A benefit to hot wallets is ease-of-use; however, they are less secure than cold wallets. In order to steal from a cold wallet, the attacker would usually require physical access to the cold wallet and know any associated password to unlock access to the funds. In all attacks analyzed in which private keys have been compromised, those private keys belonged to hot wallets (Figure 11).

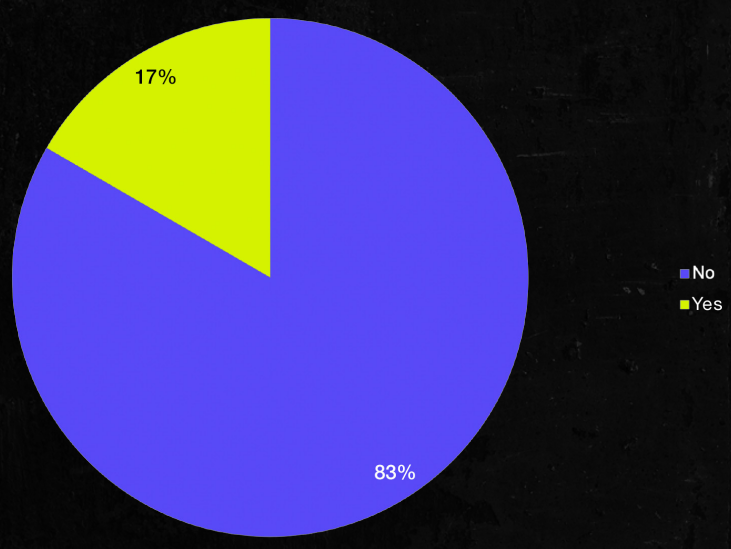

Figure 10: Multi-sig usage

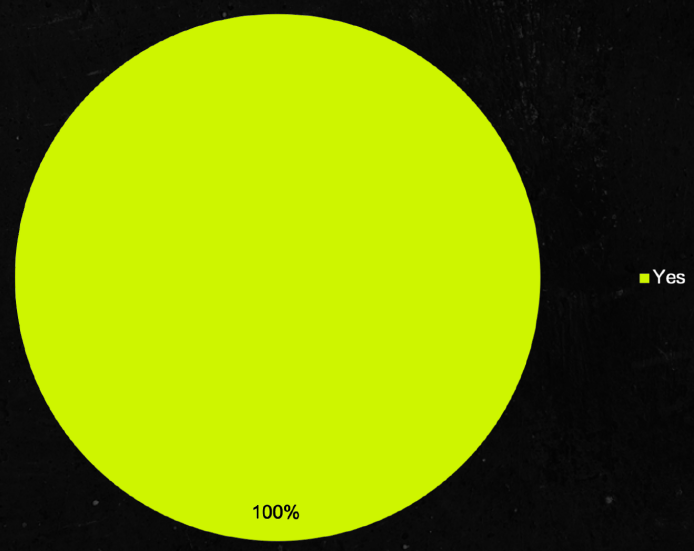

Figure 11: Hot wallet usage

5.3 FLASH LOANS

A flash loan is a 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 in what measure?

Among all the types of attacks in which flash loans can be leveraged, only in 39% of them was a flash loan used to execute the attack (Figure 12). Regarding contract exploitation, flash loans are not used in the majority of the cases, only in approximately 29% of them (Figure 13). However, when price manipulation attacks are being carried out, flash loans are used in 73% of them (Figure 14). When there is an attack against the governance protocol, according to the studied data, flash loans are used in all cases (Figure 15). This data shows that flash loans should be taken into account as a possible attack vector in those protocols that could be vulnerable to price manipulation and governance attacks.

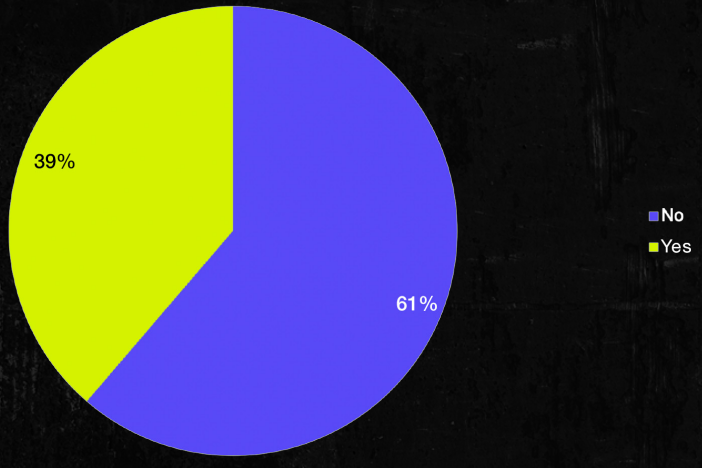

Figure 12: Flash loan usage

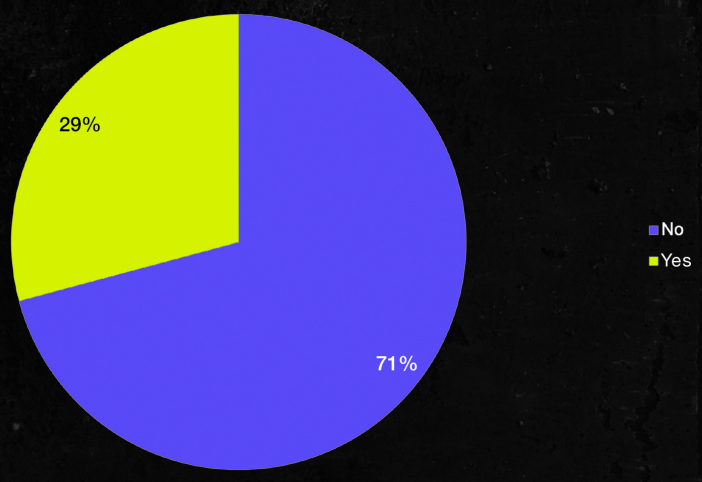

Figure 13: Flash loan usage in contract exploitation attacks

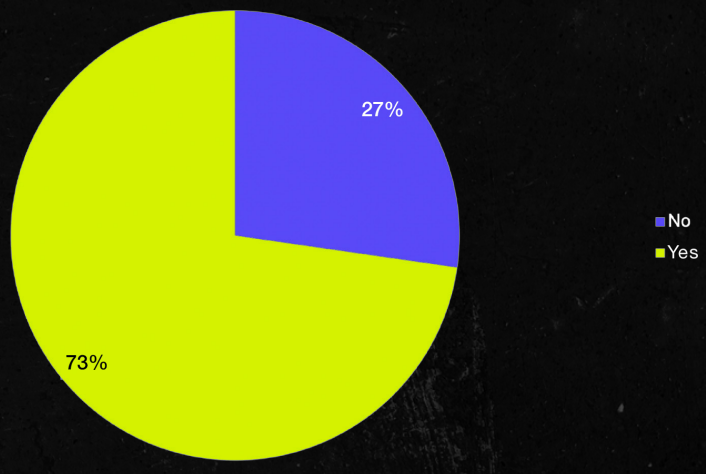

Figure 14: Flash loans usage in price manipulation attacks

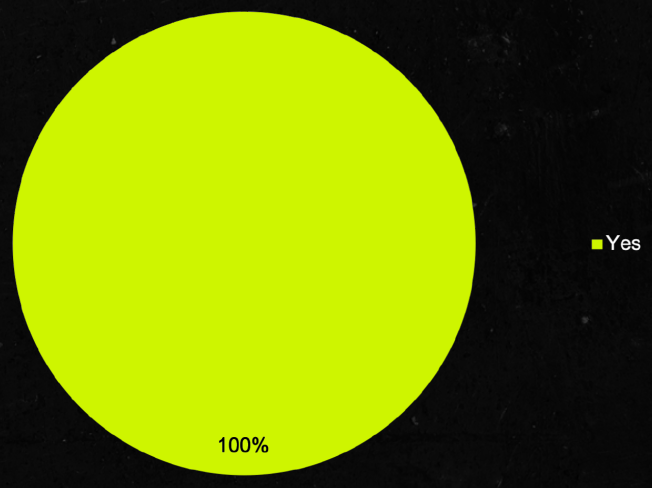

Figure 15: Flash loans usage in governance attacks

Flash loan-based attacks seem to spike in 2021, when more than half of the attacks used this mechanism, before falling in 2022 (Figure 16)

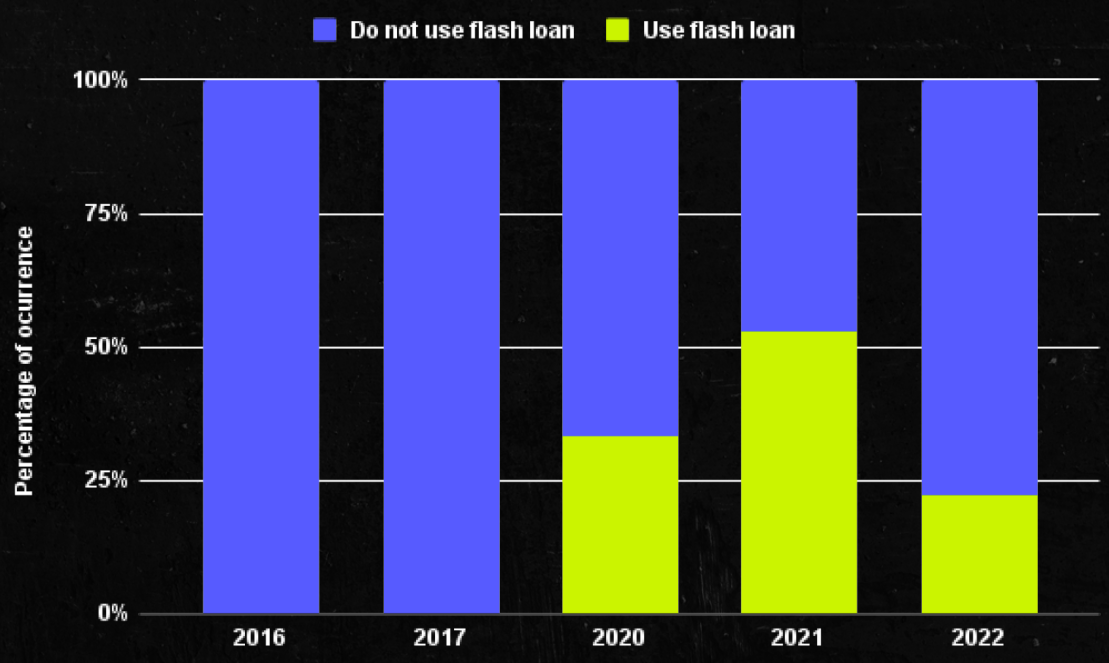

Figure 16: Flash loans usage through the years

5.4 PRICE MANIPULATION

Price manipulation attacks can take advantage of different elements to execute the attack. For this study, we are going to consider three categories:

- **Contract exploitation:** The attacker can take advantage of some kind of bug or failure in the contract code to manipulate the price of assets. One example is the way in which decimals are processed.
- **Bad oracle:** Oracles provide a way to access external data sources. A bad oracle is 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 the user with incentives to report in a faithful way.
- **Other causes:** For example, a low supply of tokens

Figure 17 shows the distribution per type of price manipulation attacks. The majority (64%) is possible because of contract exploitation, while the second most common cause is the use of a bad oracle (29%). Thus, while the most important thing to consider when trying to avoid this kind of attack is securing the contract code, using a reputable oracle could prevent more than a quarter of them.

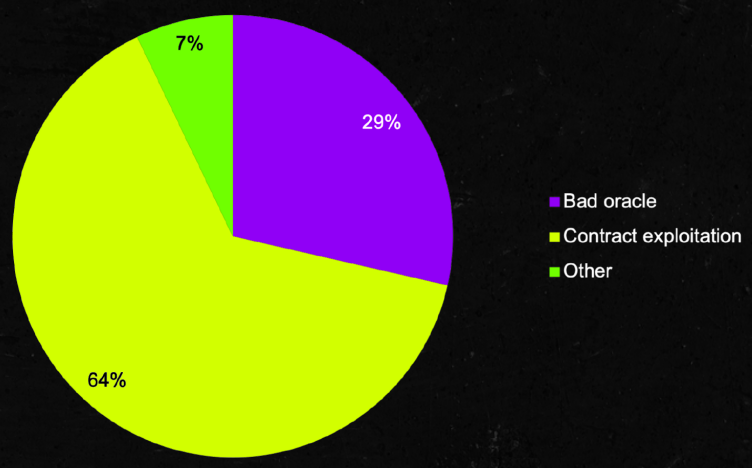

Figure 17: Price manipulation

According to Figure 18 most of the contract exploitation and all attacks related to bad oracles happened in 2021.

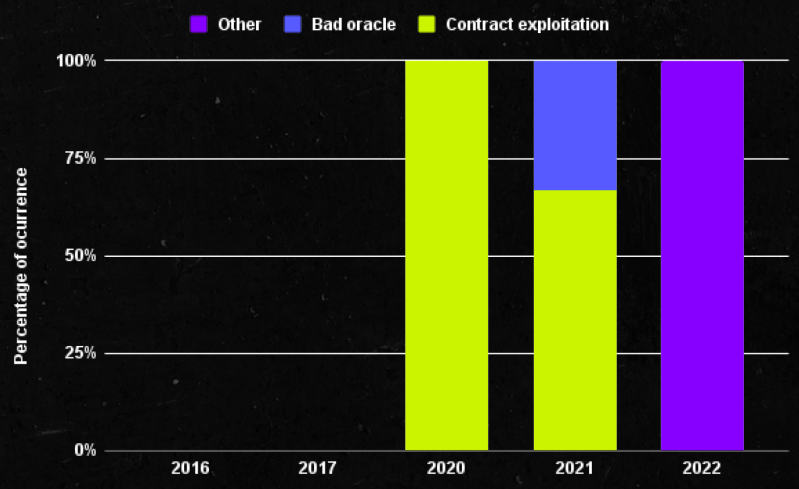

Figure 18: Price manipulation per year

5.5 ATTACKS PER CHAINS

To see how different chains are being attacked, this report studies the relationships between the types of attacks defined previously and the chains studied in Section 4.

Figure 19 shows the relationship between the different types of attacks and the different chains. It should be noted that contract exploitation seems to be the most common cause in all of them except for Polygon, Bitcoin, and Avalanche. In In Polygon and Bitcoin, the most common cause is private key theft or leakage. Avalanche has been attacked equally by contract exploitation, price manipulation, and rug pulls.

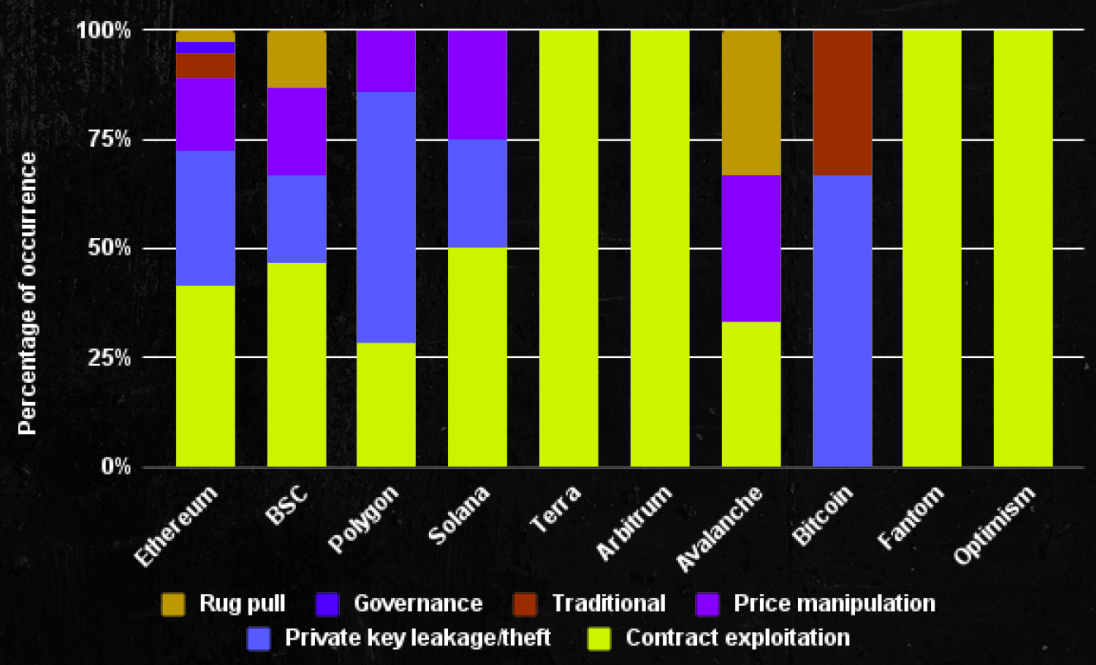

Figure 19: Types of attacks per chain

TYPE OF PROTOCOLS

The DeFi space is a diverse ecosystem with a myriad of protocols offering different services.

To identify which protocols are more vulnerable to attacks, we analyzed the nature of the victims of past attacks. It should be noted that a project can belong to more than one category. For example, it can be a gaming protocol and a marketplace.

Figure 20 shows which protocol types are usually targeted in attacks. Most of the victims are lending and borrowing platforms (approximately 20% of the total). After that, bridges and yield farming are the second most vulnerable protocols, accumulating almost 13% of them. CEXs [3], currencies and Automated Market Makers accumulate around 11%, 9% and 9% respectively. Other types of protocols are usually less attacked. It should be noted that it has been labeled as Services those protocols that provide specialized services and do not comply with any of the other categories.

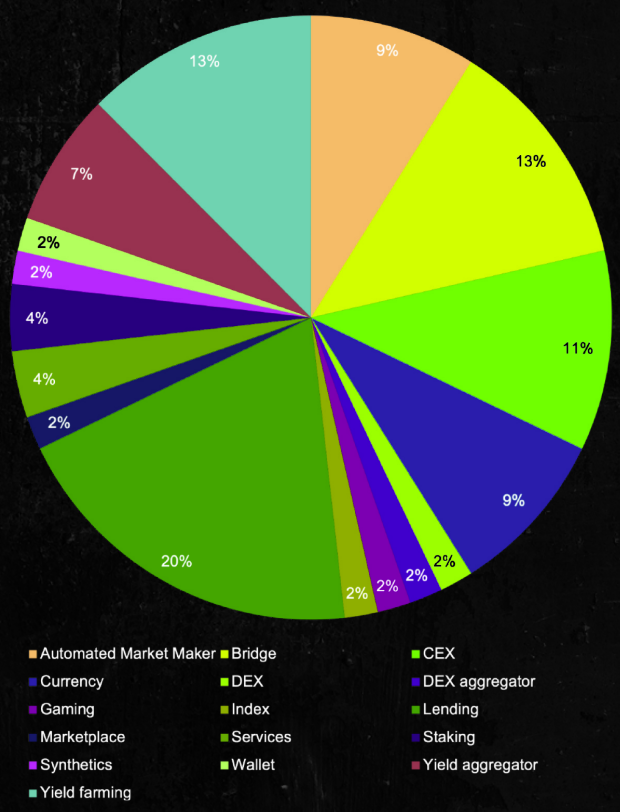

Figure 20: Types of protocols

Comparing the percentage of attacked protocols of each type versus the number of instances of that protocol in the studied sample [4] (Figure 21), we can see that, while DEXs are the most popular protocols in quantity, they are one of the least hacked ones. Something similar happens with yield farming protocols. Lending, Bridges and CEXs are fewer in number in the DeFi space but have been the target of a larger number of attacks in comparison. Thus, it seems these protocols are more vulnerable to attacks. Automated Market Makers and Yield aggregators seems to also be vulnerable but to a lesser extent.

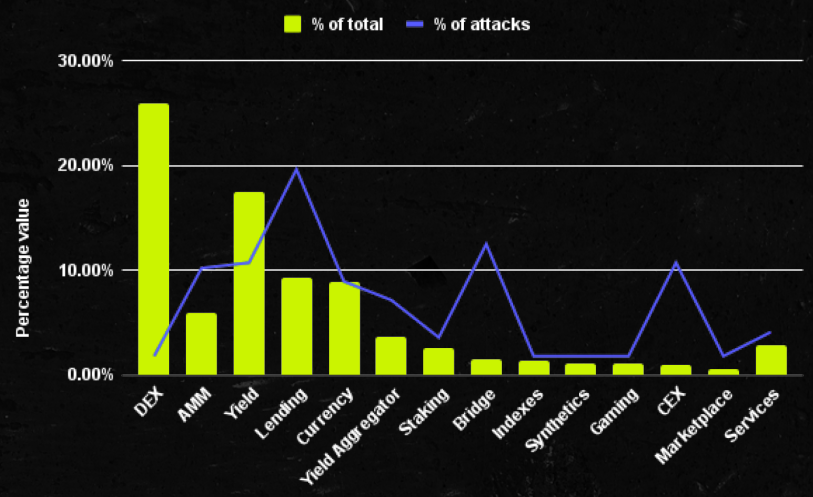

Figure 21: Types of protocols by percentage

Regarding the most attacked protocol per year, it is noticeable that, in 2021, the two most attacked types are Lending protocols and yield farming. However, in 2022, the most attacked protocol types are bridges, pulling far ahead of other protocol types. This is probably due to the growing popularity of this kind of protocol.

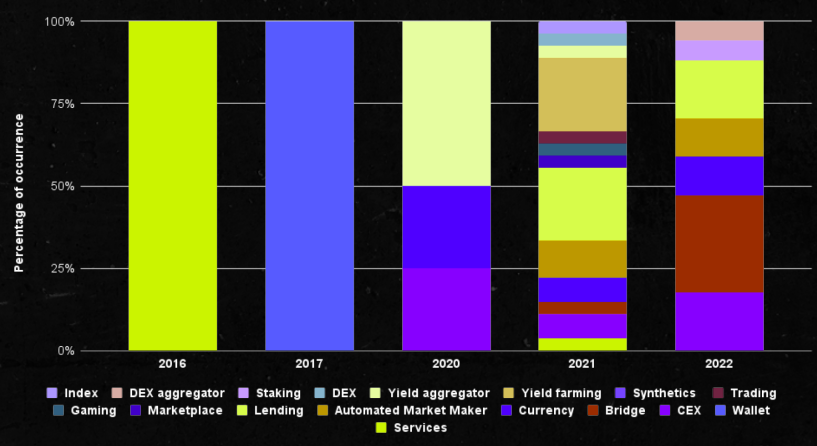

Figure 22: Type of protocols per year

3 Although CEXs can be considered not to be a protocol in some instances, we will categorize them as such for the sake of data normalization and continuity

4 Retrieved from https://defillama.com/docs/api

6.1 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 DAOs is really small (Figure 23). This seems to mean that there is no relation between the type of governance in the protocol and its vulnerability.

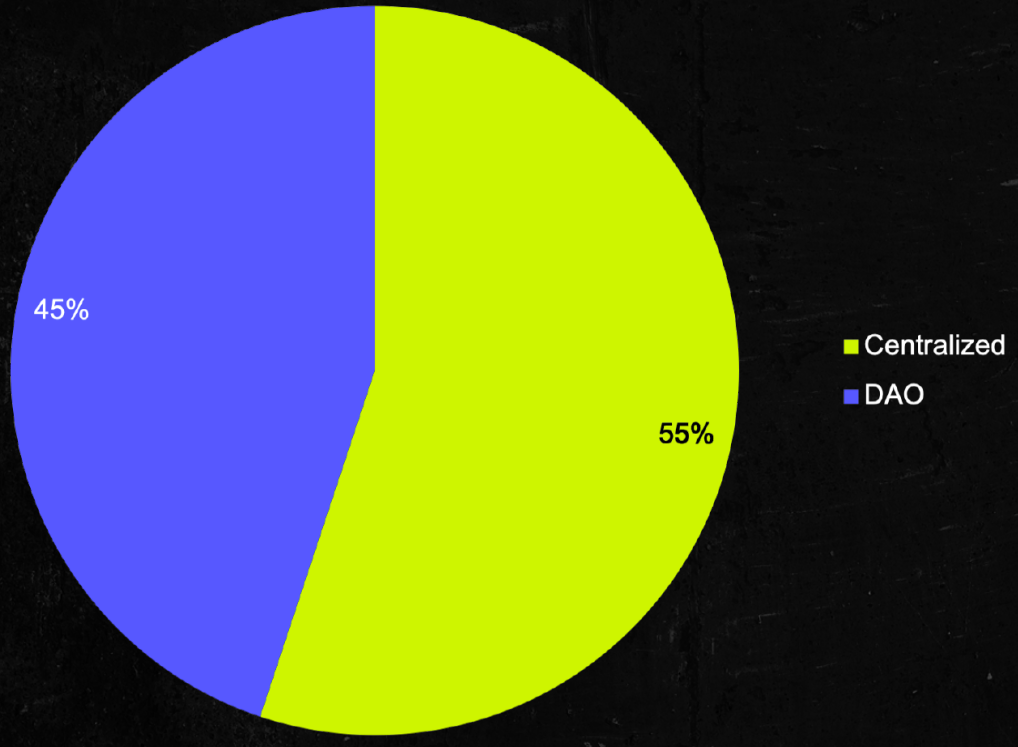

Figure 23: Type of governance

6.2 TYPE OF PROTOCOLS PER CHAINS

Figure 24 shows the distribution of type of protocols per chains. CEXs are the most attacked protocols in well-established chains like Ethereum and Bitcoin.

On BSC, however, the most attacked type of protocol was yield farming. On Polygon, lending protocols were the most targeted. For the other chains, the distribution of protocols is really similar.

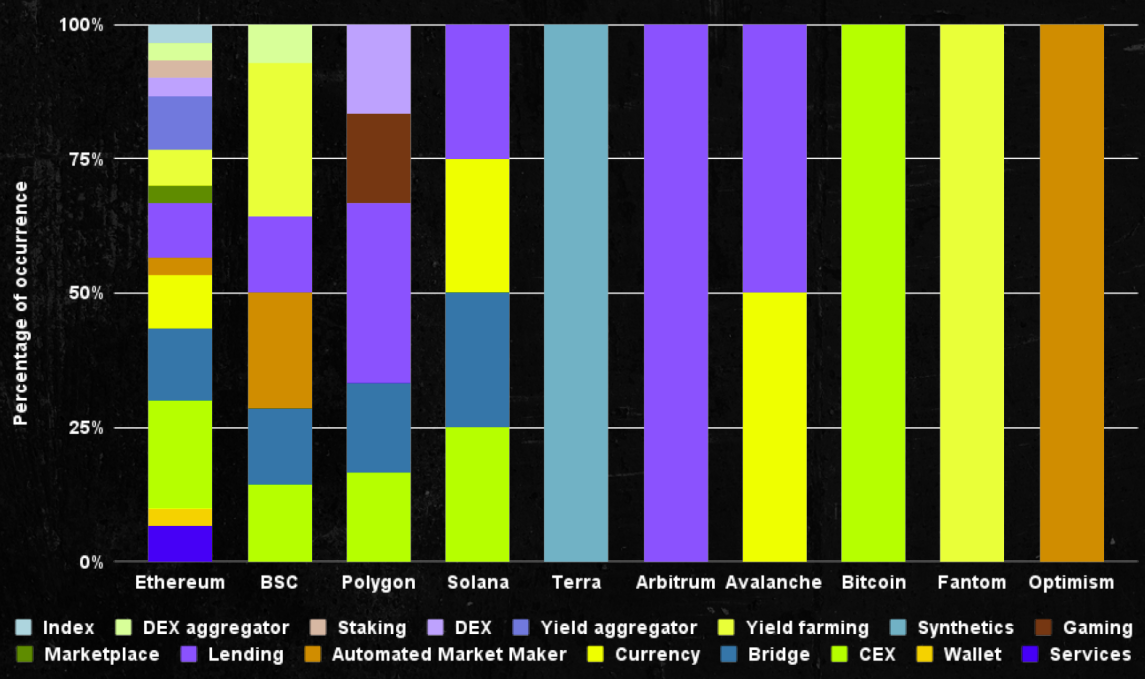

Figure 24: Type of protocols per chains

6.3 TYPE OF PROTOCOLS PER TYPE OF ATTACKS

Figure 25 shows the type of attacks versus protocols. It can be seen that contract exploitation is the most common, or one of the most common, in the majority of the protocols.

However, in yield aggregators, yield protocols and indexes it is equal in number to price manipulation attacks. Another significant detail is that, on CEXs, the most common attack is private key theft or leakage.

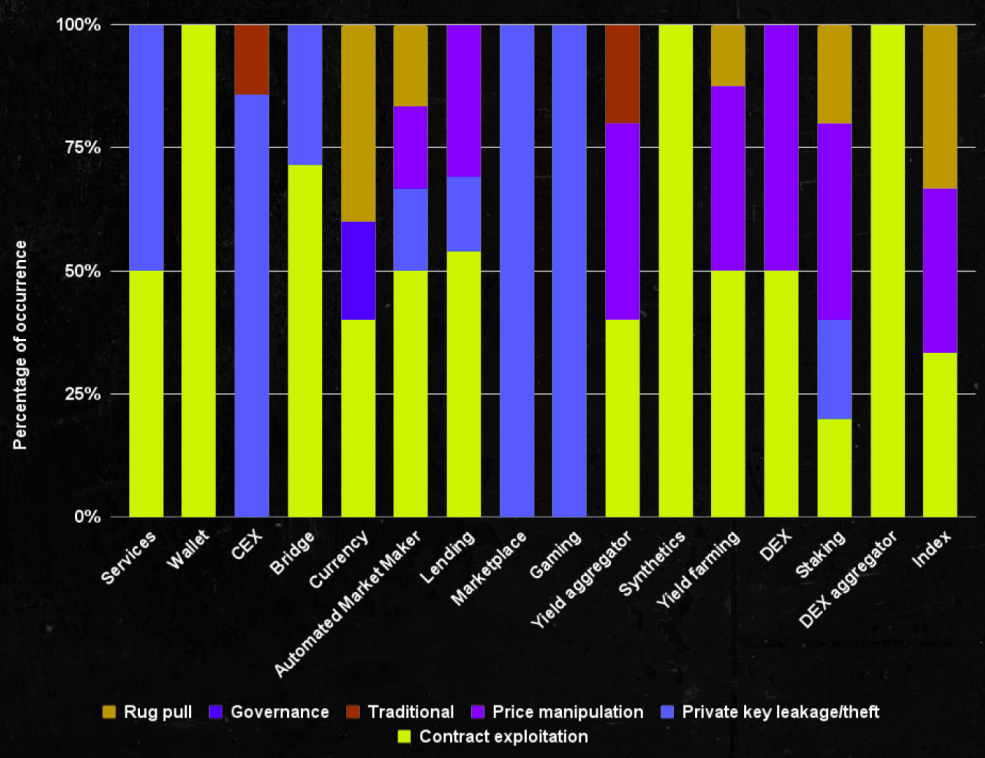

Figure 25: Type of attacks per protocols

TYPE OF FUNCTIONS

When interacting with the protocol 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 and that functions, especially those that are more complex, can fall under more than one category:

- **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.
- **Swap:** The function is used to swap assets. It can call an internal function or protocol to know how much of an asset to swap for another.
- **Mint:** The function is used to mint assets. It can call an internal function or protocol to know how much of an asset to mint.
- **Execute:** The function is used to execute certain functionality in 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.
- **CalculatePrice:** The main purpose of this function is to calculate price of assets that are involved in the transaction.
- **VerifyProof:** The function's objective is to verify certain actions/roles on the blockchain. Especially relevant on bridges

Figure 26 shows the most commonly-attacked types of functions. It can be seen that most of the functions attacked have to do with the withdrawal of funds or assets (31%). The second most common is those that mint assets to an address (16%). After that, functions that swap assets are most attacked (11%). Those in which functionality is executed or that allow users to deposit funds are next on the list by number of attacks (9%). Initialize, upgrade, transferOwnership, calculatePrice, and verifyProof are attacked to a lesser extent.

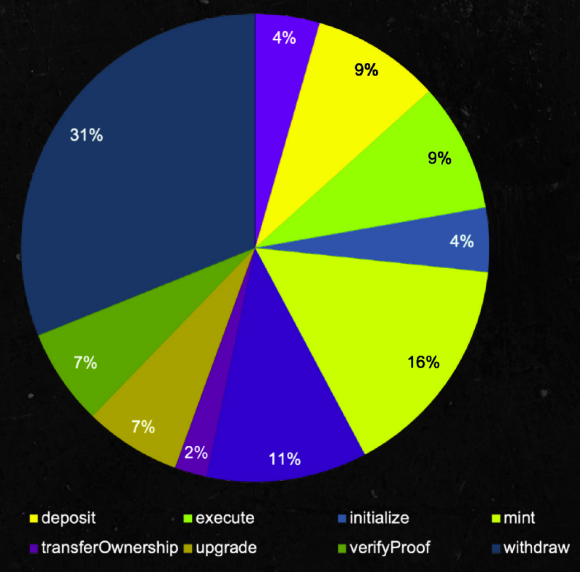

Figure 26: Type of functions

This makes sense given that mint and withdraw are the two most straightforward methods of extracting assets out of a protocol. However, the other types of functions could also help to accomplish this purpose. For example, transferring contract ownership could grant special permissions in the contract and allow the attacker to drain the protocol afterwards; or by upgrading a contract to a malicious implementation.

Figure 27 shows the most commonly-attacked type of function per year. In recent years, the most common is withdraw, followed by mint. It should be noted that, in the first couple of years, the most attacked function varied between execute and initialize. It is remarkable that those related with verify proofs are only present in 2022, probably due to the popularization of bridges.

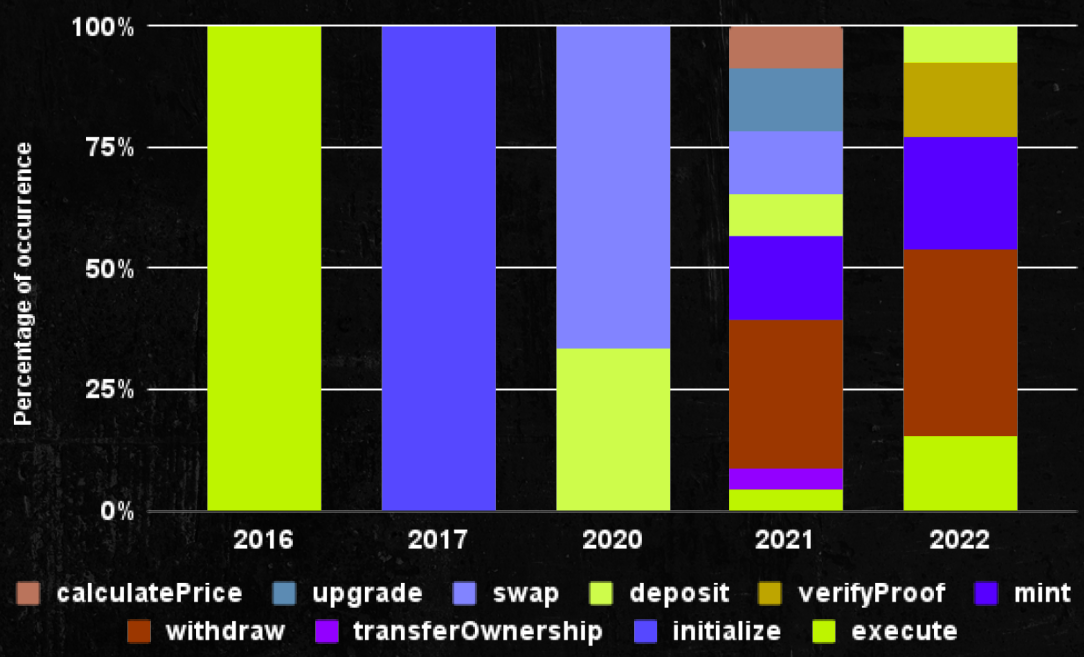

Figure 27: Type of functions per year

7.1 TYPE OF FUNCTIONS VS CHAINS

Figure 28 shows the distribution of vulnerable functions per chain. Most attacked contracts in Ethereum are vulnerable either because of withdraw-like functions or executable-like ones.

In BSC, however, most of them are either withdraw or mints. Polygon has been attacked exclusively through swap functions. It should be noted that, for most of the other chains with a lower number of attacks, the vulnerable functions have been those used to withdraw funds, except for Fantom, where the most commonly-attacked function has been a deposit one.

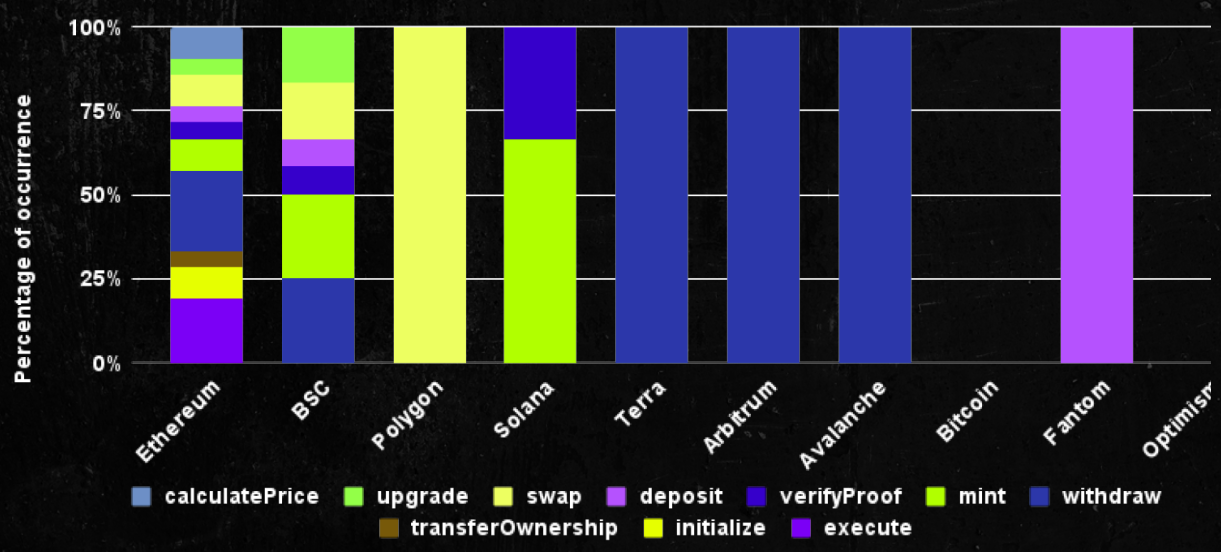

Figure 28: Type of functions per chain

7.2 TYPE OF FUNCTIONS VS TYPES OF ATTACKS

Figure 29 highlights the types of functions used in various types of attacks.

Most contact exploitation attacks are made by exploiting some kind of withdraw-like function. Attacks involving private key theft also exploit this function, using the key to withdraw funds from the protocol. The second most common way to exploit a contract is by the use of initialize functions, followed by mint and verifyProof. Regarding private key theft/leakage, execute and update in the protocols are used. Governance attacks are possible by means of execute-like functions. Price manipulation attacks, however, take advantage of withdraw as well as mint functions and also calculatePrice and swap to a lesser extent but with not a high difference in number. Rug pulls mostly use withdraw functions with upgrade and swap used to a less extent.

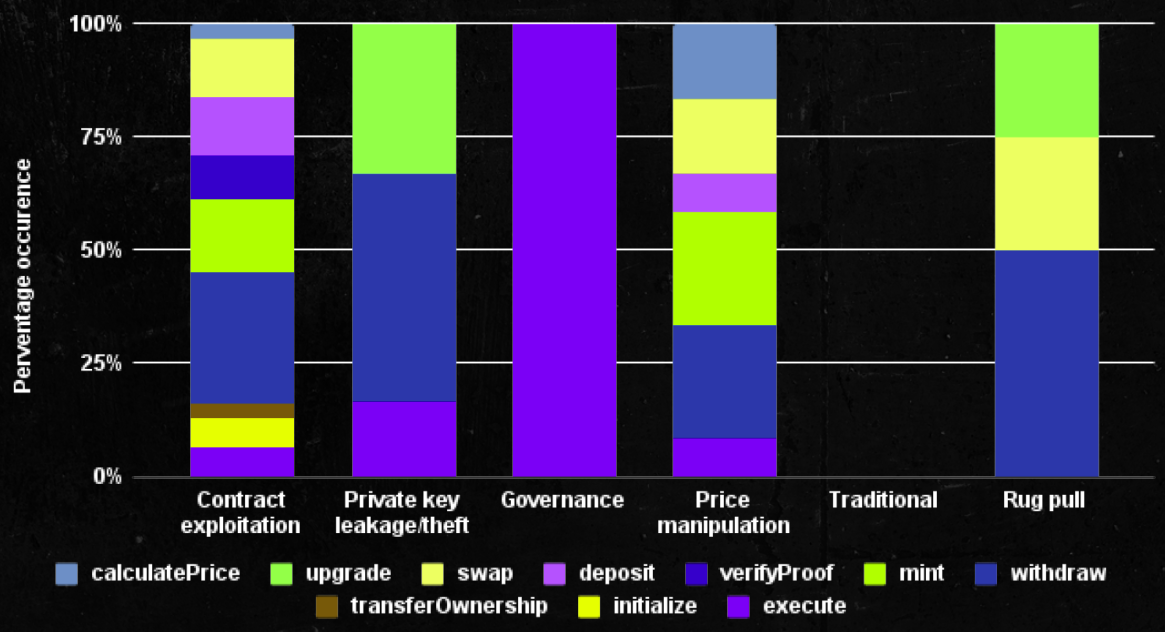

Figure 29: Type of functions versus type of attacks

7.3 TYPE OF FUNCTIONS VS PROTOCOLS

In this section, we want to examine which types of functions are usually used to attack each type of protocol. As expected, Figure 30 shows that the most common function used to attack bridges is verifyProof.

Swaps are the most-used function to attack market makers, yield aggregators, and DEXs and are one of the most common for staking and yield farming protocols. CalculatePrice is mostly used on lending protocols and indexes [5]. Deposit seems to be used mostly on lending protocols. Initialize is used mostly on wallets and bridges. transferOwnership has been used mostly to attack bridges. Mint has been used in various protocols, and it is the second most used function in bridges (together with withdraw), one of the most used in currencies (alongside deposit, withdraw and mint) and yield farming (with withdraw), and it is also used on yield aggregators and lending platforms. Deposit is used only on lending platforms. Upgrade has been used to attack lending platforms and other services.

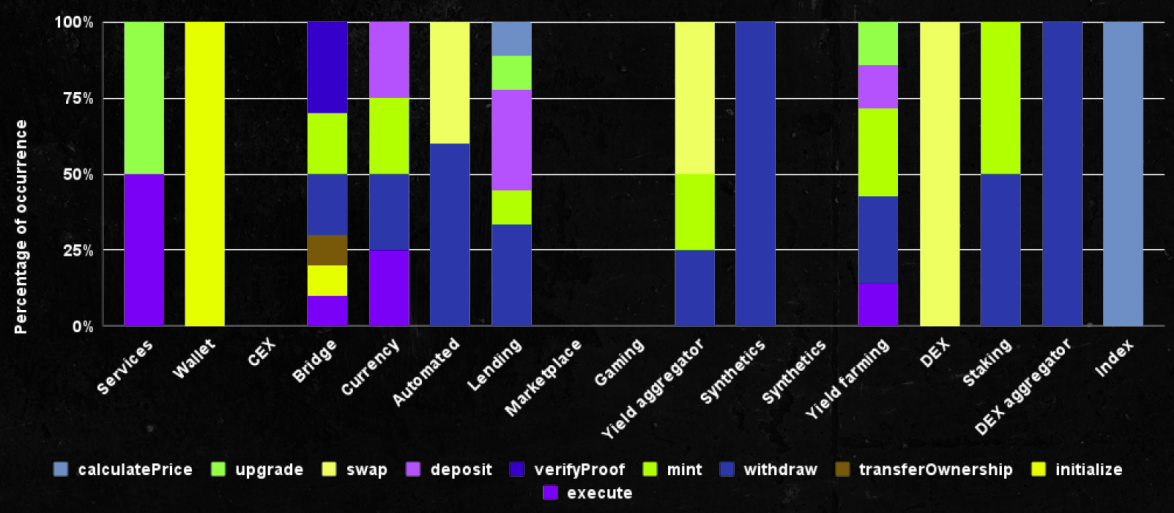

Figure 30: Type of functions versus type of protocols

5 Crypto indices are a financial instrument that tracks the performance of a basket of cryptocurrencies (https://www.defipulse.com/blog/crypto-indices)

WERE THEY AUDITED?

Security audits of protocols' smart contracts can help to prevent attacks. However, it is not a guarantee, as the number of attacked protocols that were audited is still a significant amount (28%) , albeit less than those that weren't (34%) (Figure 31).

There are some circumstances in which the smart-contract does not play a significant role in the attack. This is the case for those that were subjected to traditional attacks, private key leakage/theft and rug pulls (N/A on the figure, 38%). This number is close to the quantity of hacks for those that did not audit their contracts. Therefore, an audit of the protocol as a whole (e.g pen testing and holistic review) is necessary in order to improve the security of the protocol and mitigate risks. It should be also taken into account that we don't have data about those protocols in which an audit did prevent a future attack.

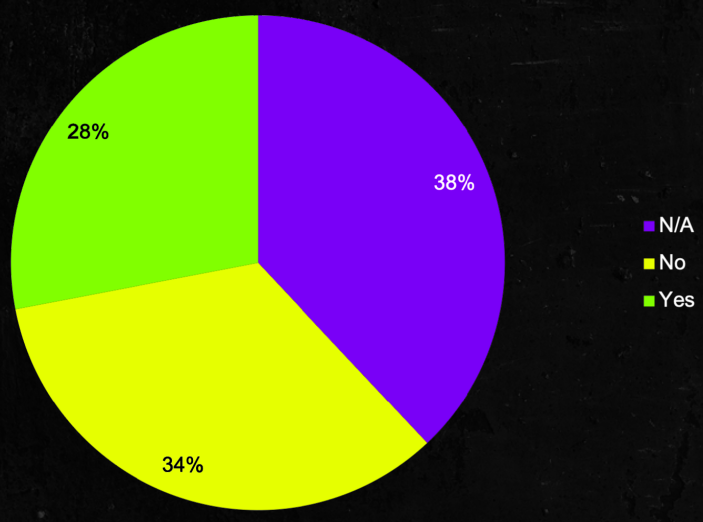

Figure 31: Audited protocols

For the sake of illustration, Figure 32 shows a comparison of audited versus not audited protocols only on Web3-native attacks.

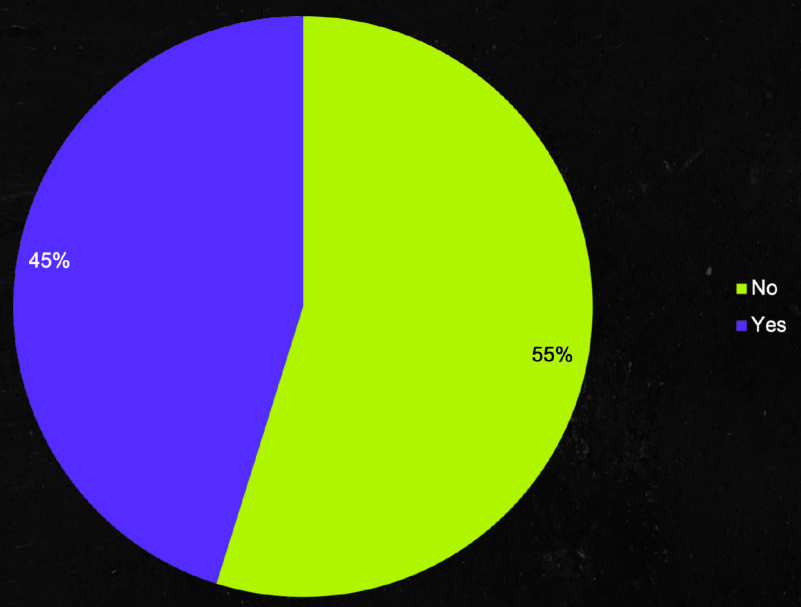

Figure 32: Audited protocols

However, it should be also noted that, among those protocols that were indeed audited, 14% of them included the exploited vulnerability on the report (Figure 33).

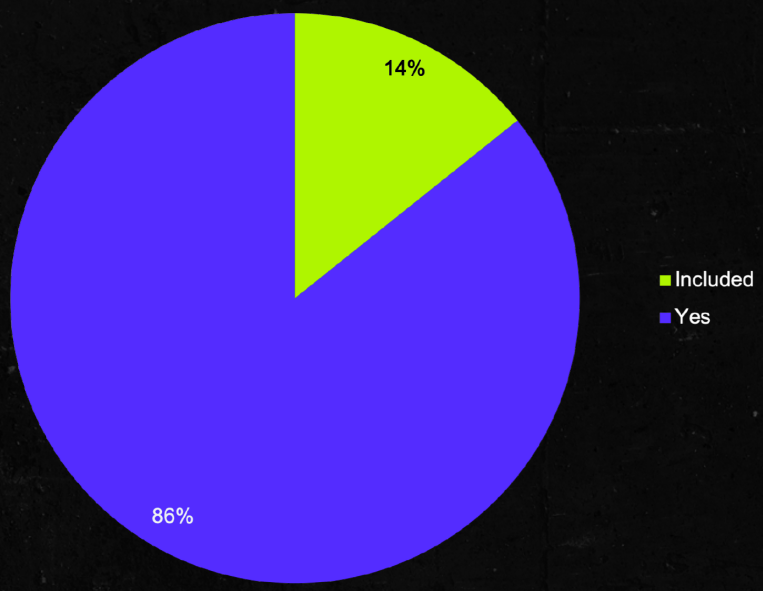

Figure 33: Audited protocols and percentage of vulnerabilities included on the report

In 2021, there was a spike in protocols attacked that had an audit conducted compared to those that were not audited. However, in 2020 and 2022, the number of protocols whose smart contracts were not audited is higher than those whose contracts were (Figure 34). This hopefully will result in a trend where audited protocols experience fewer hacks as time passes.

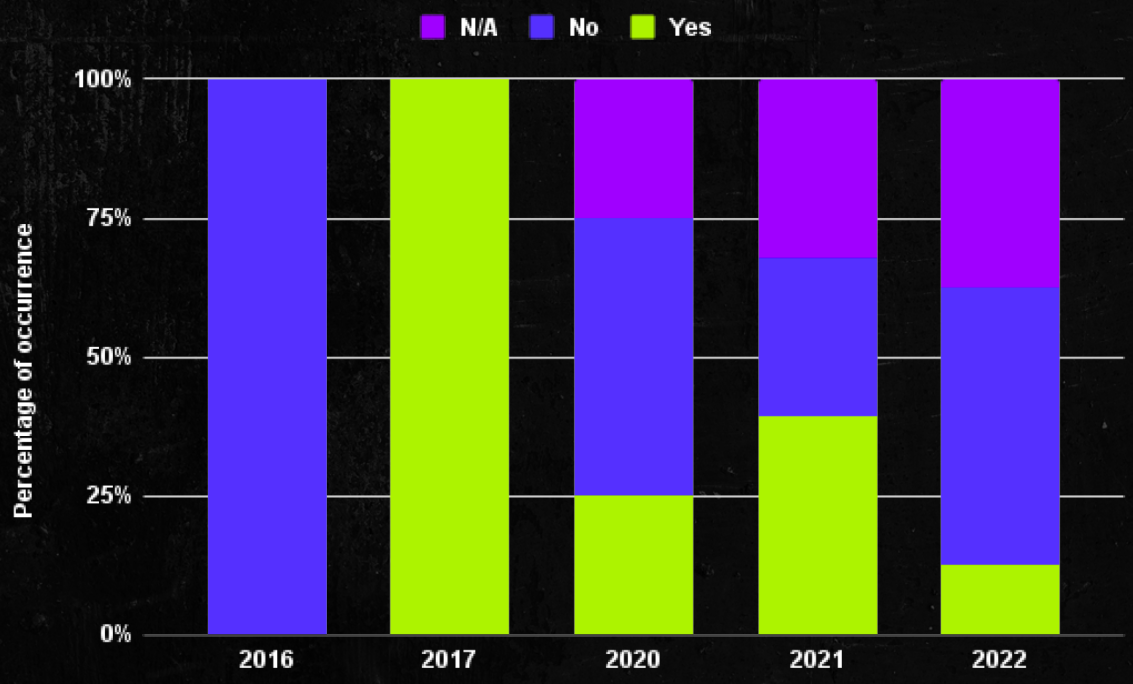

Figure 34: Audited protocols per year

8.1 AUDITED PROTOCOLS BY CHAIN

While most chains have seen more attacks on non-audited protocols, there are exceptions, such as Solana, Avalanche, and Fantom. However, Fantom only has a single audited protocol.

Other chains that also have a single protocol vary between not audited and N/A (Figure 35).

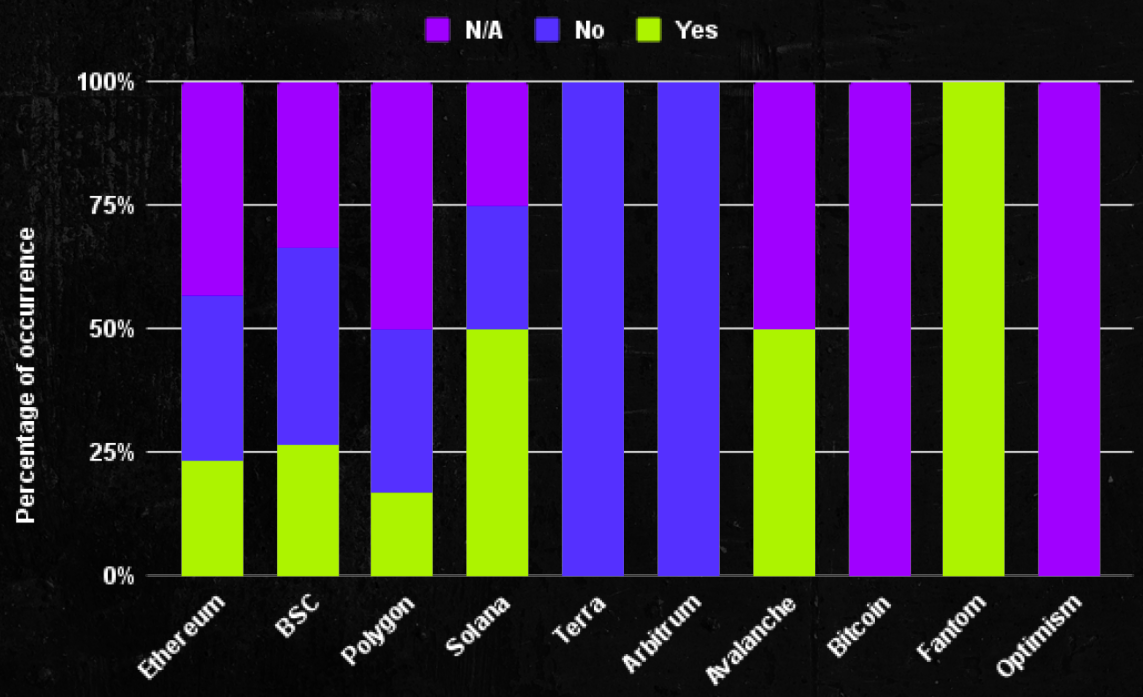

Figure 35: Audited protocols per chain

8.2 AUDITED PROTOCOLS BY TYPE OF ATTACK

Most attacks on smart contracts (e.g., contract exploitation, governance, and price manipulation) involved protocols that had not been audited (Figure 36).

Price manipulation, however, is an exception: most of the hacks were on audited contracts. This could mean that just auditing the code is not enough, auditing the whole environment and how the protocol would interact with other DeFi applications is necessary. In the cases where the smart contract is not part of the attack vector (N/A), and an audit of the security and generation of the private keys and traditional Web2 security audits on the whole system of the protocol (e.g. webapp audits) should be carried out.

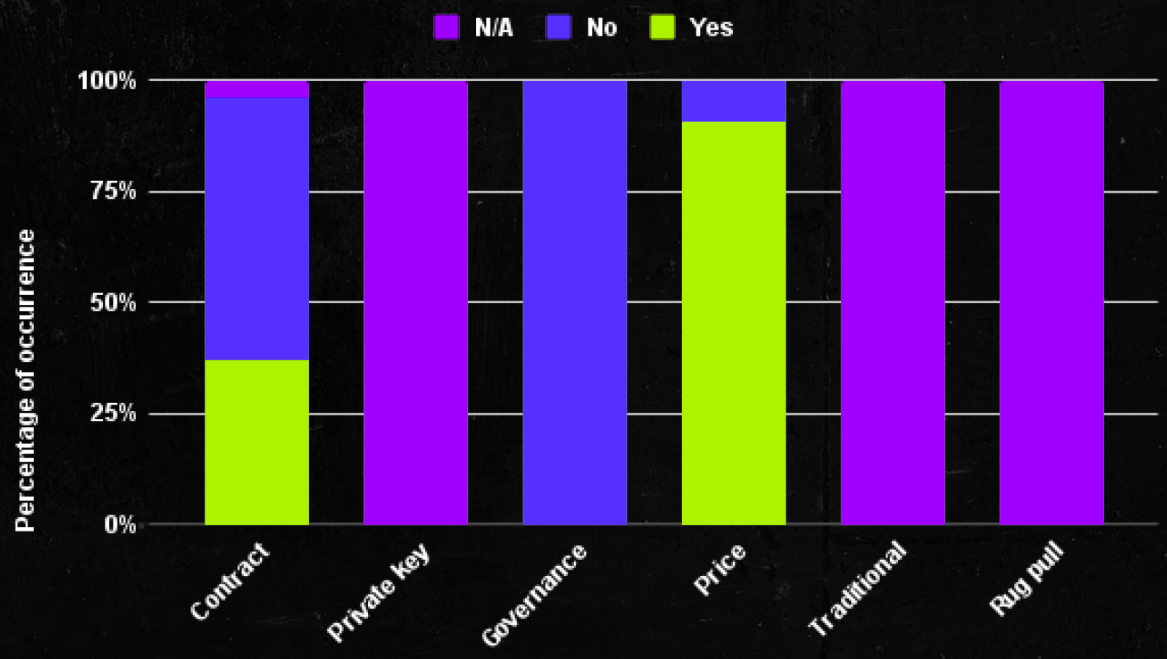

Figure 36: Audited protocols per type of attack

8.3 AUDITED PROTOCOLS BY TYPE OF PROTOCOL

Most protocols attacked seem not to have been audited. Furthermore, in some cases, their attack couldn’t have been prevented even with a smart-contract audit (e.g., private key leakage).

However, there are a few (wallets, yield aggregators and farming, DEXs and Staking) that were attacked even though they were audited (Figure 37). This raises the question of why this would happen. If we remember Section 6.3, almost all of them were also the most prone to price manipulation attacks. This kind of attack is difficult to detect in an audit if it only centers on the current code being audited and not the whole ecosystem.

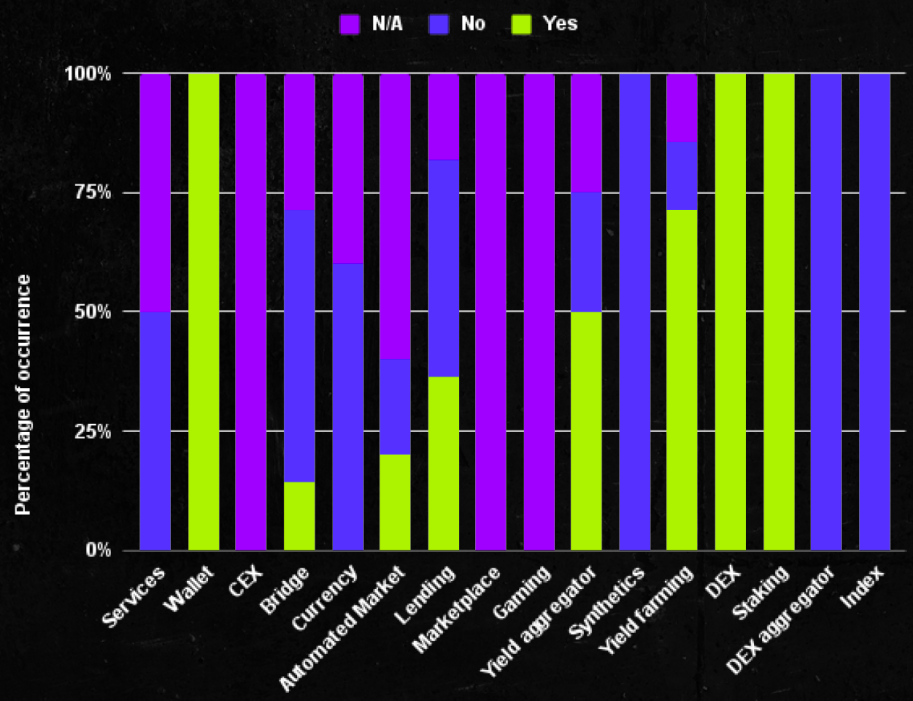

Figure 37: Audited protocols per type of protocol

8.4 AUDITED PROTOCOLS BY TYPE OF FUNCTION

Figure 38 shows whether each type of vulnerable function was audited. It should be noted that, in most cases, the majority of functions were not audited.

One special case is mint functions, where the quantity of audited ones surpasses slightly those which weren't. This is likely due to their use in price manipulation attacks.

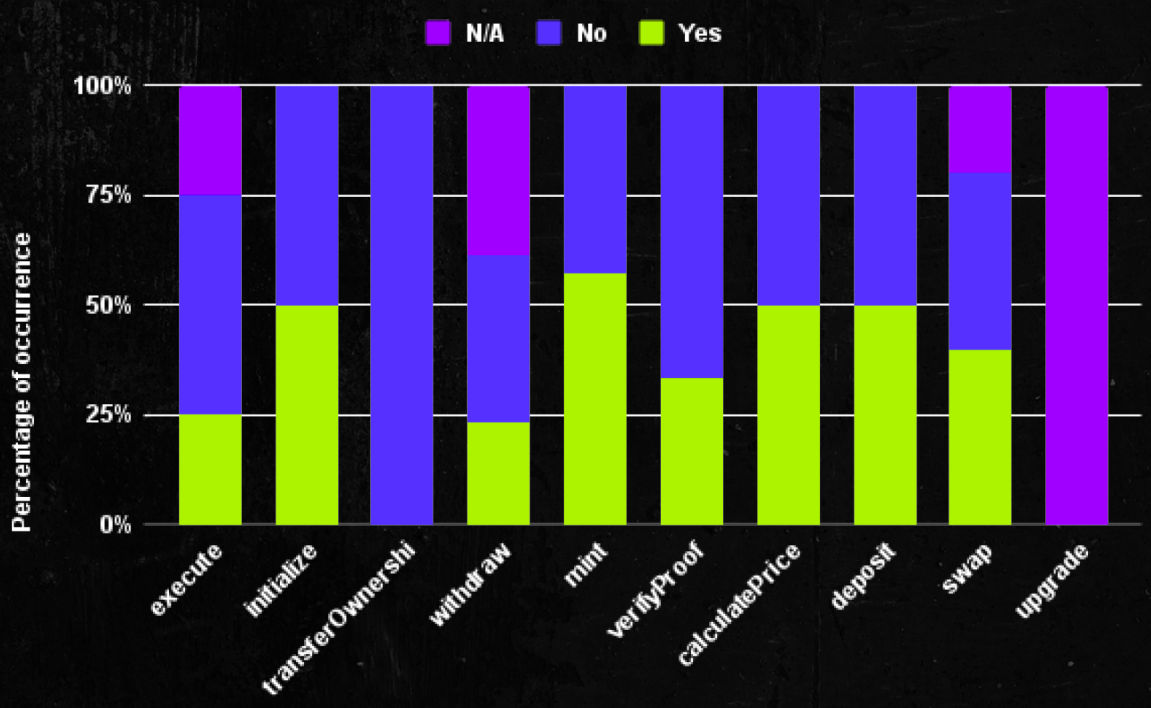

Figure 38: Audited protocols per type of function

ACTIONABLE TAKEAWAYS

Based on the analyzed data and key findings (Section 2), the following actions are recommended

- Be especially cautious when using Solana and make sure all contracts are properly tested and audited.
- 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 performed complete audits and not only smart contract ones.
- Be especially careful with the logic and input validation of the contracts. Test and review your code carefully, considering different use cases and make sure to perform proper input validation in each function.
- Consider using multi-signature, MPC, and cold wallets. Using a cold wallet reduces the chance of private keys being stolen. Furthermore, using multi-signature or MPC wallets to perform permissioned and administrative actions on the protocol helps minimize the chance of major damage if one key gets 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 bad oracles. Use reputable, multi-source, decentralized, and incentive-driven oracles like Chainlink.
- Be careful with Lending protocols, Bridges and CEXs. As a protocol owner, be careful of 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 containing funds, are secure. As a user, be careful with these types of protocols, make sure they were properly audited and consider using alternatives (like DEXs) instead.
- Pay special attention when programming functions whose functionality coincides with those defined in Section 8.
- Use available tools to enhance the security of likely vulnerable functions. We have been able to categorize all vulnerable functions into ten types based on with their main purpose. The most vulnerable are those with withdraw and mint capabilities. Using tools like OpenZeppelin Sentinels or Forta bots to actively monitor those functions once the contract is deployed would help to minimize losses in case of an attack. Furthermore, using https://seraph.co to protect them could help stop the attack from occurring.

// Visit Halborn.com For More

HALBORN