Overview

Volt Protocol core contributors and external auditors separately reviewed the Compound Finance codebase. Compound v2 has held billions of dollars over three and a half years without any large scale losses to lenders, and as a whole the Volt Protocol team feels it is among the safest on chain yield venues at this time.

The Volt Protocol team identified and explored a class of risks involving market manipulation in Aave and Compound style markets. These attacks require significant upfront capital, are illegal, and have uncertain chances of success or profit. As a result, the risk is not considered a blocker to VOLT PCV deposit in Compound. The Volt Protocol team has disclosed this risk category to both Aave and Compound teams, who did not consider it of sufficient concern to take action to pause the relevant markets.

Along with an analysis of the Compound codebase and the market manipulation risk category, this document details measures already in place or planned to reduce the risk to Volt Protocol from venues. There are general failure modes and shared tailed risks in lending pools, so the Volt Protocol team is working on features that will enable market governance participants to more safely allocate funds among these venues. The ongoing (as of 1 September 2022) cETH freeze proves that governance is among the greatest dangers for well-established markets.

The review of the Compound codebase conducted on behalf of Volt Protocol by external auditors can be found here: ⚡ Using Compound as a Yield Venue

Market Manipulation Risk

In reviewing the security of Compound in preparation for PCV deposit, the Volt Protocol team explored the risk of market manipulation attacks. The attack is possible when the amount of a token borrowable on markets like Aave and Compound is large compared to the liquid market. The most notable example is ZRX, which has borrowable liquidity on *each* of these markets comparable to or greater than the usual daily volume across all centralized and decentralized exchanges.

Collateral Withdraw Attack

In the most basic form of the attack, a user borrows a large amount of the available supply of a token, such as ZRX, and sells it across multiple centralized and decentralized exchanges, depressing the open market price. The combined borrowable liquidity for ZRX on Aave and Compound v2s is more than twice the average daily volume currently. Once the oracles which inform Aave and Compound update, the user withdraws most of their original

collateral. With numbers: supply $30 million collateral in stablecoins. Borrow “$20 million” of illiquid token and sell it, depressing the token’s market price by 95% and realizing $7.5m. New market value of the user's debt is $1 million, allowing withdrawal of $28m collateral. Attacker profits $5.5m, leaving underlying market(s) with bad debt.

For this to work, it is necessary that the amount of the token borrowable be sufficiently large to depress the open market price. When the same token is borrowable on both Aave and Compound, as is the case with ZRX, this appears more likely. It is uncertain whether the current amount of borrowable liquidity would be sufficient to carry out the attack, though we believe that it is possible based on our analysis. If carried out in the form described above, bad debt would be confined to the ZRX market and not concern Volt Protocol PCV.

Two-Oracles Attack

There is a more sophisticated form of the attack that relies on differential oracle updates between Aave and Compound, and is in theory possible whenever two markets use different oracle systems. Whenever a Chainlink update gets sent into Compound, its price is compared against the cached 30 minute TWAP of that asset on Uniswap, which can be re-updated every 30 minutes. If the price on Uniswap and the price from chainlink are more than 15% apart, the Compound Oracle system does not accept that update as a valid price input, and will refuse to accept this new market price into the system.

An attacker can take advantage of this difference by, as in the Collateral Withdraw Attack, borrowing a large amount of ZRX from Aave and selling it rapidly across all exchanges to depress the price. Chainlink will “correctly” update the Aave price down, allowing the user to borrow even more against the same collateral, with the end result that *all* of the ZRX on Aave is borrowed. Once there is little to be gained from selling into the market, the attacker can instead deposit ZRX on Compound as collateral and use it to borrow. Since Compound rejected the price update from Chainlink and is still using a higher old TWAP price, the attacker can cash out by borrowing stablecoins or ETH on Compound. This would result in bad debt in the ZRX market on Aave, and bad debt in a stablecoin or ETH market on Compound.

Recommended Migitations

Given the large up front cost and substantial risk of the attacks described above, it may not be likely to witness them in practice. However, the growing size and diversity of on-chain lending pools and constantly changing market conditions mean that manipulation or cross-pool attack risks should be given ongoing scrutiny. From the perspective of Aave and Compound, pausing new supply or borrows in less liquid tokens such as ZRX would eliminate the risk of this class of attack. Per asset collateral caps found in Aave v3 and Compound v3’s “single borrowable asset” model both help to reduce the risk of attacks.

Another possible mitigation, though not likely practical for existing markets, would be a rate limit on borrows of assets (or using certain assets as collateral) proportional to their market liquidity depth.

Volt Security Measures

Lending pools like Compound and Aave don't automatically detect bad debt and accordingly update their balances. In the event of a liquidation failure or an attack as described above, bad debt is not uniformly distributed among cToken or aToken holders. Instead, whichever lenders redeem first do not take any loss, and only the portion of lenders who are last to exit and correspond to the percent of bad debt will take a complete loss. Losses are likely small in comparison to total market liquidity unless a truly catastrophic event has occurred, such as an infinite mint exploit of a collateral token.

To help protect VOLT holders from losses, Volt Protocol has a PCV Guard role which can move protocol assets out of a yield venue and back to the control of Governance in an emergency, which gives the protocol a high likelihood of escaping a market that has bad debt without any losses. This role is currently assigned to a set of core team addresses as it could be used to perform a DOS attack on the protocol with unnecessary withdrawals unless permissioned.

Future versions of the Volt system will have more open and permissionless processes to trigger removal of PCV from venues under certain conditions. As an example, a Bad Debt Sentinel would allow any address to provide evidence of bad debt in one of Volt's supported yield venues, and receive a reward for triggering the removal of PCV from the venue. Another example is an Excess Utilization Sentinel, which would have a similar effect but look out for when liquidity in a market is low, another common sign of trouble. This would require a reliable "TWAP" for the utilization in relevant markets, which we expect will be possible after the VOLT yield oracle system is complete. Another sentinel could be created called the Governance Sentinel, which could trigger removal of funds before a new governance proposal goes live in an integrated market. While these measures are being developed in the context of our first PCV deposit into Compound, they will generalize to similar venues like Aave, Euler, or any lending pool style market.

Compound Codebase

The Compound codebase uses a proxy logic pattern that leverages the delegatecall opcode to have proxies implement the logic contract's business logic. The naming convention used in this code base is postfixing smart contracts with either Delegator, which implies that a contract is a proxy and implements no business logic, or Delegate, which implies that the contract contains the business logic that will be delegatecalled by the corresponding Delegator contract. All upgradeable smart contracts were checked to ensure that they weren't vulnerable to Parity Wallet-like attacks where an attacker could take ownership of a contract and then self-destruct the contract, thus bricking the system, or steal funds. No vulnerability of this type was found in the Delegate smart contracts.

Governance

The Compound Protocol is governed by a token weighted voting DAO, currently called the GovernorBravoDelegateG2, which gets delegate called by the GovernorBravoDelegator. An interesting feature of this G2 delegate contract is the whitelist feature, which allows whitelisted accounts to propose to governance without having the minimum 25,000 required COMP tokens for a proposal. The admin of the GovernorBravoDelegator is the Timelock, and the admin of the Timelock is the GovernorBravoDelegator, which means that addresses that can propose without the required COMP can only be added to the whitelist through a governance proposal. The current GovernorBravoDelegateG2 contract was patched into the system with this governance proposal that was executed March 19, 2022.

Timelock

The Compound protocol has a delay of at least 2 days before any governance proposal can be executed after being proposed to the timelock. Combined with the other delays in the system to make changes through the governor, this delay in the timelock is sufficient.

GovernorBravoDelegateG2

The Governor Bravo contract currently in use has a 13,140 block delay before any proposed action can be voted on which translates to 2 days, then an additional 19,710 block delay, or 3 days for voting, and a 2 period in the timelock, which means any proposal will last approximately 7 days before being able to be voted in, which gives VOLT Protocol plenty of time to exit in the case of a malicious governance proposal on Compound. After the merge, when block times decrease from 13 to 12 seconds, so will the amount of time proposals take to pass, however this change is marginal, and the compound team can adjust the number of blocks for these parameters after the merge if they see fit.

CTokens

All the following CTokens have been checked and verified that they have been upgraded to the newest implementation of the CToken contract which uses solidity 0.8.10. cMaker, cLink, cCOMP, cTUSD, cUNI, cDAI, cWBTC2, cUSDT, cAAVE, cSUSHI, cYFI. This logic smart contract is secure from re-initialization as it uses the CErc20Delegate smart contract, which does not set the admin address, thus preventing unauthorized initialization as the check in CToken.sol checks that the msg.sender initializing the contract is in fact the admin. CTokens cUSDC, cZRX and cBAT do not implement the proxy logic pattern, and cannot be upgraded in place, because they were deployed before the Compound codebase switched to an upgradeable model. CEther uses a non upgradeable smart contract that was on the 0.5.8 version of solidity.

Reentrancy

One of the largest security concerns around Compound is reentrancy as there is lots of accounting that happens internally and many different tokens that need to be moved from place to place. A key outcome of this report was verifying that this category of attack was not allowed. CEther is not vulnerable to the same reentrancy attack that happened to Fuse because it uses .transfer instead of a .call, and thus only forwards 2300 gas to the recipient, which eliminates the possibility of reentrancy from this sending of Ether. Additionally, CTokens use the Check-Effects-Interaction pattern to ensure that reentrancy attacks cannot allow an attacker to withdraw more than their pro rata share of assets. Reentrancy attacks are also prevented across all CTokens by having reentrancy locks across all mint and redeem functions.

Accounting

The Compound system relies on an interest index and accrual of interest before each action occurs in a CToken. Interest is calculated with the following formula, where i is accrued interest, Δb is the number of blocks that have passed since the last interest accrual and *ir* is the interest rate per block.

$$i = \Delta b * ir * total\ borrows$$

After interest is accrued, it is then split between the LP's and the Compound platform which takes a percentage of all interest earned. Before a mint, redeem, borrow, repay, liquidation, setting of a new interest rate model, setting of market fees and adding or reducing of reserves, interest is accrued. This prevents flash loan or other economic attacks as it enforces that all users are accessing the market at the current market price, with all interest accrued, thus getting the most up to date price on their CTokens.

CTokens enforce that you cannot send tokens to yourself. This is not considered standard behavior for an ERC20 token, and this can be easily allowed by subtracting tokens from the sender first, and then adding tokens to the recipient after.

After the merge occurs, both supply and borrow rates will increase by at least 8% across all markets. This is because Compound accounting uses blocks to measure interest accrued and makes the assumption that blocks will be 13 and not 12 seconds. Interest rate models assume 2.1m blocks per year, however this is not correct as after the merge, there will be closer to 2.6m blocks per year. This means that the Compound v2 interest rate models will be off by 25% after the merge, causing the interest rates to be higher than intended in the interest rate models. Recommend accounting for the difference by swapping out the interest rate models to have the correct blocks per year which should be around 2.62m blocks per year after the merge.

Comptroller

The comptroller is the controller smart contract that manages all CTokens for a given deployment of Compound. This contract enables liquidations, as well as stores the borrow limits, oracles for each CToken, loan to value ratios, and the CTokens which are a part of that particular version of Compound. In the current version of the comptroller, it can adjust all parameters up to the hardcoded caps. The price oracle can be set, the collateral factor can be set to a maximum of 90%, the liquidation incentive can be set to an unchecked value, CTokens can be added, but not removed as removing a CToken would completely brick the market and all accounting. Borrow limits can be set by the comptroller to minimize borrowing on certain markets. A pause guardian can pause minting new CTokens, transferring CTokens, and additional borrowing of CTokens, but not repaying of loans as that could permanently brick a market and trap users' capital. It seems that there was an ability for the guardian to pause liquidations, but that has since been deprecated.

Interest Rate Models

The interest rate paid in Compound is variable and determined by the ratio between borrowers and lenders in a given market. These interest rate models have a slope and a kink, which is a point on the slope at which the slope changes. This allows borrowing costs to increase more quickly if too much of the reserves are borrowed. The BaseJumpRateModelV2 contract allows an owner to update the kink, jump multiplier per year, base rate per year, and multiplier per year without first calling accrueInterest on the CToken. This could cause an issue where a malicious governance proposal changes the values on this interest rate model, thus causing interest that has not already been accrued to jump, which could cause a liquidation or a transfer of all assets from borrowers to lenders by forcing a liquidation. Because the BaseJumpRateModelV2 is owned by the timelock across all CTokens, a malicious proposal that would force up interest rates and cause liquidations would leave users and VOLT Protocol plenty of time to exit the market before this change went into effect.

Multiple implementations of interest rate models were observed in production such as JumpRateModelV2, LegacyJumpRateModelV2, and WhitepaperInterestRateModel. All of their owners, if they had one, were the Compound timelock, and the respective ages of these smart contracts were all over 1 year old.

Oracles

The Compound system uses Chainlink oracles to fetch price data on underlying assets of CTokens. The system compares the price from Chainlink to values recorded on a Uniswap V2 TWAP oracle to ensure that the Chainlink oracle is not reporting an incorrect value. Values not directly matching the TWAP oracle are allowed from chainlink as long as they are within 15% of the current Uniswap TWAP price. This system is secure as long as there is sufficient liquidity in

the referenced Uniswap pools, and there is sufficient liquidity for that token on other DEX's, which will allow the price to be arbitraged back to its fair market value by MEV bots. The UniswapAnchoredView contract in the Compound open oracle repo on branch master is different from the deployed contract at address https://etherscan.io/address/0x65c816077c29b557bee980ae3cc2dce80204a0c5 that is used by the comptroller. Consider updating the repository code to match the live version in production.

Admin Upgrades

The Compound codebase uses a two step process for changing the admin of any given contract. The first step in this process is to set the pending admin, the second step is for the pending admin to call into the contract, accept this pending admin position, and thus become the admin. This methodology is more secure than a simple transfer that does not require acceptance, and it means if the admin is accidentally given to an invalid address, the system doesn't lose the ability to upgrade itself because the invalid address cannot become the admin, and the ownership does not get transferred.

Solidity Versions

The Compound protocol has multiple versions of solidity running in their contracts on mainnet. The first version of solidity is 0.5.8, which is what all proxy smart contracts were deployed with. The second version of solidity is greater than or equal to 0.8.0. Even if the smart contracts were to be upgraded without adding additional variables, this upgrade would not cause an issue as long as existing storage offsets were not modified as different versions of solidity don't change how storage offsets are calculated by bytecode.

Testing Infrastructure

Compound was created at a time when the only framework available for Solidity development was Truffle. While truffle worked and was an improvement over using the solidity compiler directly, it was a nightmare to integrate with, took a very long time to run tests, and didn't allow much developer customization. To work around this, the Compound development team created their own framework, Saddle, to handle the complexities of their larger and more complex codebase. Saddle served the Compound team well, allowing them to have features not available in truffle. However, at this point, there are much more advanced frameworks for smart contracts such as foundry and hardhat. If further upgrades are going to be applied to the Compound V2 codebase after this Uniswap V2 TWAP rollback, Volt Protocol recommends adding an additional integration testing framework that is more lightweight and developer friendly. Some governance proposals with code changes have passed without integration tests, this is an antipattern and all proposals should have thorough integration tests that assert the health of markets immediately after passing of a proposal.

Conclusion

The Compound v2 codebase is thoroughly battle-tested, governance follows a lengthy process that helps make it difficult for malicious proposals to be passed, and control of the system is limited to reduce centralization risk. Guardians are in place to step in and pause contracts should issues arise, and the proxy logic pattern is used, which allows logic upgrades in the smart contracts if bugs were discovered. Interest accruing faster after the merge is not a concern to the Volt Protocol as this will increase earnings on PCV in Compound until borrowers react to the rate change.

While there are some concerns such as market manipulation in certain tokens and governance errors, appropriate mitigations within Volt Protocol will minimize the risk to VOLT holders compared to holders of “raw” cUSDC or cDAI. We recommend that vulnerable markets such as ZRX be deprecated on both Aave v3 and Compound v2, and that Compound v2 be hardened as much as possible so that protocols like Index Coop, Morpho, and Volt can safely build on top.