Decentralized Finance Protocols: Evaluating Collateralization Mechanisms

Ilya Sinyugin, University of Twente, The Netherlands

Through the appearance of Decentralized Finance (DeFi) protocols, blockchain technology has had a massive impact on the world of finance. Such protocols make use of certain technical mechanisms, specifically collateralization, that are used to control risk and secure value. This study looks at the technical comparison of various collateralization mechanisms used by major DeFi protocols. The research investigates the functionality, effectiveness, and security of such protocols. Preliminary findings suggest that protocols vary significantly in their robustness and efficiency; such differences may have major effects on the durability of DeFi protocols. This study aims to fill a major research gap and give insights to developers, regulators, and users in the growing DeFi field.

Additional Key Words and Phrases: Collateralization, Decentralized Finance (DeFi), Blockchain, Ethereum, Smart Contracts, Compound Finance, Aave.

1 INTRODUCTION

The technological innovations of decentralized finance were built upon the decentralized peer-to-peer electronic cash systems that were first mentioned by Satoshi Nakamoto in 2008 [4]. However, Satoshi Nakamoto's Bitcoin system had several limitations, including a lack of Turing-completeness and Blockchain-blindness, which made it difficult to effectively create decentralized applications. These constraints led to the development of Vitalik Buterin's Ethereum blockchain [3]. Smart contracts and distributed systems on Ethereum allow for less complicated deployment of financial applications; as a result, developers needed a term that would generalize such decentralized applications (dApps) and the term "DeFi" was finally coined in 2018 [7].

The DeFi sector can be considered one of the most fast-paced sectors in blockchain in terms of recent developments. However, such advancements are usually related to the three main functions of DeFi: the translation of monetary banking services, lending and borrowing platforms, and the facilitation of the usage of financial instruments like Decentralized Exchanges (DEX), Derivatives, and others [5]. DeFi's entire functionality is carried out using smart contracts, which act as the foundation for decentralized applications. With such technology, there is no need for any middleman, and all of the functionality is validated via program logic [5]. But the main danger behind smart contracts is that they might get exploited, resulting in cryptocurrency stolen from the protocol without any chance of reimbursement. Therefore, it is crucial to analyze and compare existing DeFi protocols to ensure maximum security of the funds, particularly regarding the lending and borrowing platforms where collateralization is used.

TScIT 39, July 7, 2023, Enschede, The Netherlands

Due to the growing popularity of lending and borrowing platforms, these protocols have accrued a lot of value, peaking at $50 billion in early 2022 [1]. Such platforms frequently attract new users because of their accessibility, lack of legal obstacles, and privacy. As a result of such platforms being decentralized, they rely on collateral to meet the demands of both borrowers and lenders. Particularly because of the high-volatility nature of cryptocurrency, overcollateralization is needed to ensure that the borrower's loan value does not exceed the value of the collateral [1]. Therefore, there exist different platforms that provide solutions for lending platforms that have different visions of how such protocols should be implemented. Due to that, there are various protocols with different methods of implementing functionality such as collateralization ratios and interest rate models.

This research will focus on the collateralization mechanisms of the two major DeFi protocols – Compound Finance and Aave. Such mechanisms are used in DeFi for borrowing certain cryptocurrency by providing an asset as a collateral. Particularly, this study will analyze the technical and functional differences between the two protocols. This study aims to fill a research gap in terms of technical analysis and comparison of collateralization mechanisms in the major DeFi protocols. Furthermore, this paper will enable further research to compare other DeFi protocols with distinct collateralization mechanisms. To fully understand the strengths and weaknesses of the two platforms, a thorough study of these protocols' operational processes will be conducted. This includes liquidation procedures – steps performed by the protocol when a borrower is unable to repay the loan, risk overcollateralization requirements – the proportion of additional collateral needed to reduce risk of liquidation, and risk management techniques – the ways in which potential risks are managed in the protocols.

2 RESEARCH GOALS

This study aims to evaluate the contributions made by different researchers and developers in the field of decentralized finance, with a focus on collateralization mechanisms in lending protocols. The research questions of this paper are as follows:

1. What are the different approaches to implementing collateralization mechanisms in DeFi lending platforms?

2. How do the collateralization mechanisms compare in terms of efficiency, robustness, and security?

To answer both research questions, the research is split into several sections. First and foremost, a literature review will be conducted, where two of the largest DeFi lending platforms, Compound Finance and Aave, will be analyzed in separate sections. A literature review will support the points made in the rest of the paper, which will be related to the comparison of the two main collateralization mechanisms.

The rest of the paper is organized as follows: Section 3 discusses the literature review regarding the main collateralization mechanisms, Section 4 discusses the analysis between the two collateralization mechanisms, and Section 7 concludes the research.

3 LITERATURE REVIEW

This section discusses two of the most popular DeFi lending platforms: Compound Finance and Aave. Therefore, the subsections will be split into two, where the previously mentioned lending platforms will be described based on their documentation and existing research.

3.1 Compound Finance

Compound Finance is a decentralized protocol that was developed for the purpose of enabling the borrowing of Ethereum tokens and establishing a secure method of having a positive yield while storing assets [8].

Since Compound Finance is a lending protocol, the two main aspects of it are supplying and borrowing assets. In comparison to regular exchanges, the protocol itself aggregates the supply of each individual user; therefore, the assets can be considered fungible. This enables users to withdraw their assets at any point in time. The way this is implemented in Compound Finance protocol is by using ERC-20 tokens called *cToken.* This token represents the original asset supplied by the user plus any interest that accrues over time. Hence, whenever an asset is borrowed, *cTokens* are used as collateral. That functionality enables users to borrow at any time without any specified requirements or negotiations.

Figure 1 illustrates the functionality of supplying assets to the Compound Finance protocol. This is done by first initializing references to both the *ERC20* and *CERC20* contracts, where *CERC20* contract is the Compound Finance's version of the *ERC20* contract, that is a standard for tokens on Ethereum blockchain. This step is necessary to interact with methods from both contracts later in the code. After initializing references, an approval method is called from the *ERC20 underlying* asset. When this step is completed, a user has given permission to Compound Finance to spend a set amount of the given *ERC20* token. The amount is set by the user; it can either be set to a fixed number that is necessary for this exact transaction for security purposes or it can be set to a maximum number for convenience. After the approval takes place, a *mint* function is called, which supplies a given number of *ERC20* tokens to the contract in exchange for the *cToken.*

```
contract MyContract {
 event MyLog(string, uint256);
 function supplyErc20ToCompound(
  address _erc20Contract,
  address _cErc20Contract,
  uint256 _numTokensToSupply
 ) public returns (uint) {
   Erc20 underlying = Erc20(_erc20Contract);
   CErc20 cToken = CErc20(_cErc20Contract);
   underlying.approve(_cErc20Contract, _numTokensToSupply);
   uint mintResult = cToken.mint(_numTokensToSupply);
   return mintResult;
}
```

Figure 1. Compound Finance supply method

Figure 2 displays the functionality of redeeming the *cTokens* back to get the original supplied assets. The way a user redeems the tokens is based on the Boolean condition of *redeemType.* A user has the option of either specifying a quantity of *cTokens* to convert and redeem or specifying a quantity of underlying asset to redeem. In the first case, the *redeemType* variable would be set to *True*, and in the second case it would be set to *False.* Such a condition provides two alternatives for the users in the way they would like to convert their *cTokens.* Some users may prefer to know the exact amount of *cTokens* they would like to redeem, while for others, it could be more beneficial to know the exact amount of the underlying asset they would receive. There is a difference between the two outcomes due to the exchange rate between a *cToken* and an underlying asset.

```
contract MyContract {
 function redeemCErc20Tokens(
  uint256 amount,
  bool redeemType,
  address _cErc20Contract
 ) public returns (bool) {
  CErc20 cToken = CErc20(_cErc20Contract);
  uint256 redeemResult;
  if (redeemType == true) {
   redeemResult = cToken.redeem(amount);
  } else {
   redeemResult = cToken.redeemUnderlying(amount);
  }
  emit MyLog("If this is not 0, there was an error", redeemResult);
  return true;
 }
}
```

Figure 2. Compound Finance redeem method

Users are allowed to borrow assets up to their total borrowing capacity that cannot be exceeded. The total borrowing capacity is based on every asset that is used as collateral by the user. Borrowing capacity (BC) is calculated as follows:

$$BC = \sum(Value\ of\ each\ cToken * Corresponding\ Collaterral\ Factor) \quad (1)$$

Value of *cToken* denotes the current market value of the token, and Collateral Factor denotes the collateral factor associated with that token. The collateral factor can range anywhere from 0 to 1 and represents the proportion of the asset's value that can be borrowed.

The liquidation can occur if the user's borrowing capacity reaches 0, that is, when the borrowing balance has exceeded the total collateral value. Liquidations occur because of searchers, who detect any accounts with a borrowing capacity of 0, in order to take advantage of such an opportunity by buying out a borrower's collateral at a discounted price. On Compound Finance protocol, this process occurs by calling out methods from Figure 3. First, the searchers execute *isLiquidatable* to check whether a certain account is susceptible to liquidation. If it is, then the *absorb* function is called, which moves the collateral to the protocol's balance sheet. This helps the protocol cover the debt by dealing with the loan. Finally, the searcher determines the price of the collateral that can be seized by calling the *quoteCollateral* function. This allows the searcher to obtain the collateral at a discounted price via *buyCollateral,* which is then usually instantly sold due to profitable price discrepancy for the searcher [2].

```
function isLiquidatable(address account) public view returns (bool)
function absorb(address absorber, address[] calldata accounts)
function quoteCollateral(address asset,
 uint baseAmount) public view returns (uint)
function buyCollateral(address asset,
 uint minAmount, uint baseAmount, address recipient) external
```

Figure 3. Compound Finance liquidation methods

The functionality of interest rates in Compound Finance enables users who deposit assets to gain interest, which is derived from the amount of interest paid by borrowers [6]. The interest rate changes dynamically with every change in supply and demand. Therefore, the interest rate index is calculated as follows:

$$Index_{a,n} = Index_{a,(n-1)} * (1 + r * t) \tag{2}$$

where $\mathrm{Index}_{a,n}$ denotes the new interest rate of an asset 'a' at block 'n', $\mathrm{Index}_{a,(n-1)}$ denotes the prior interest rate, r denotes the per-block interest rate, and t represents the time in blocks elapsed since the last time the interest rate was updated [8]. Borrowers also accrue the interest rate in the same way as lenders, and the value derived from the interest rate is added to the amount of their outstanding loan.

3.2 Aave

Aave Protocol is designed for a similar purpose as Compound Finance; hence, users can take part as lenders, borrowers, or liquidators [9].

This protocol functions with the help of a lending pool that holds reserves of various assets and it is usually referred to as a reserve. Aave has a similar approach to tokenization as Compound Finance. When users deposit their assets into a reserve, they receive the same number of derivative tokens called *aTokens*. They are used for providing collateral to the protocol, which then enables to take out a loan.

Upon user's deposit, the function *mintOnDeposit* from Figure 4 is triggered to mint *aTokens*. First, the user's balance is checked via *cumulateBalanceInternal*, which takes into account the current balance and any accrued interest rate over time. Then, if a user wants to redirect the interest to another account, the *updateRedirectedBalanceOfRedirectionAddressInternal* is called; otherwise, the user's balance stays the same and nothing else is updated. Finally, *_mint* is executed in order to mint *aTokens* straight to the user's deposit address.

```
function mintOnDeposit(address _account, uint256 _amount)
 external onlyLendingPool {
 ,
 ,
 uint256 balanceIncrease,
 uint256 index) = cumulateBalanceInternal(_account);
 updateRedirectedBalanceOfRedirectionAddressInternal(_account,
  balanceIncrease.add(_amount), 0);
 _mint(_account, _amount);
 emit MintOnDeposit(_account, _amount, balanceIncrease, index);
}
```

Figure 4. Aave mint method

The final step of the deposit process before the user can take out a loan is to enable the deposited funds to be used as collateral. This is done with the code in Figure 5. Where the Boolean value *useAsCollateral* is set to *True*, which means that the asset is allowed to act as collateral in case a user decides to borrow. Finally, the *setUserUserReserverAsCollateral* method is called with the aforementioned Boolean value and the address of the deposited asset. Then, if the user would like to borrow a specific asset, there are two options: stable or variable rate modes. This gives freedom of choice for the user since a certain rate mode could be more suitable for them depending on their strategy. Variable rates are based on the overall supply and demand in Aave protocol, while stable rates are usually fixed in short-term, however, there is a possibility of them rebalancing depending on the market conditions [9].

```
LendingPoolAddressesProvider provider =
 LendingPoolAddressesProvider(address(
 0x24a42fD28C976A61Df5D00D0599C34c4f90748c8));
LendingPool lendingPool = LendingPool(provider.getLendingPool());
// mainnet DAI adderss
address daiAddress = address(
 0x6B175474E89094C44Da98b954EedeAC495271d0F);
bool useAsCollateral = true;
lendingPool.setUserUseReserveAsCollateral(daiAddress, useAsCollateral);
```

Figure 5. Aave use as collateral method

Similar to Compound Finance, the derivative tokens, in this case *aTokens*, can be redeemed for the underlying asset at any time, unless the tokens are currently being used as collateral. Figure 6 illustrates the process of redeeming the underlying asset.

```
/// Instantiation of the aToken address
aToken aTokenInstance = AToken("/*aToken_address*/");
/// Input variables
uint256 amount = 1000 * 1e18;
/// redeem method call
aTokenInstance.redeem(amount);
```

Figure 6. Aave redeem method

As in Compound Finance, there is a limit to how much a user can borrow, which in Aave is determined by a health factor. The health factor determines whether an asset is undercollateralized or not. If it is, then a loan can be liquidated. This occurs when the health factor drops below 1. Health factor *Hf* is calculated as follows:

$$H_f = \frac{TotalCollateralETH * L_q^a}{TotalBorrowsETH + TotalFeesETH} \tag{3}$$

where L_q^a represents the average liquidation threshold, which is constantly updated to keep up with the volatility of assets. If the ratio of borrowed assets surpasses the liquidation threshold, then the position may be liquidated [9]. The liquidation threshold is calculated with the following equation:

$$Liquidation\ Threshold = \frac{\Sigma Collateral_i\ in\ ETH * Liquidation\ Threshold_i}{Total\ Collateral\ in\ ETH} \tag{4}$$

If the user's health factor falls above 1, then their position is prone to liquidation. This is when searchers check a user's health factor by calling the function *getUserAccountData* from Figure 7, which returns several values, one of which is *healthFactor*. Then, by receiving the *debtToCover, debtAsset* and *healthFactor* values from the previous function call, the searcher can liquidate the position by calling *liquidationCall*. Similar to Compound Finance, the searcher receives an amount of liquidated collateral at a discount price, which makes it profitable to search for accounts with a low health factor. Furthermore, Aave enables the searcher to specify which asset they would like to receive by setting *receiveAToken* to either *True* for *aToken* or *False* for a regular underlying asset.

```
function getUserAccountData(address user)
function liquidationCall(address collateral, address debt,
 address user, uint256 debtToCover, bool receiveAToken)
```

Figure 7. Aave liquidation methods

The interest rate model in Aave is a two-slope linear function that determines the interest rates:

$$R_v = \begin{cases} R_{v0} + \frac{U}{U_{optimal}} R_{slope1}, if\ U \leq U_{optimal} \\ R_{vo} + R_{slope1} + \frac{U - U_{optimal}}{1 - U_{optimal}} R_{slope2}, if\ U > U_{optimal} \end{cases} \quad (4)$$

Where R_{v0} represents the base variable rate, U represents the utilization rate of the deposited asset, $U_{optimal}$ represents the target utilization rate that is calculated by the model, R_{slope1} represents the interest rate slope below $U_{optimal}$ and R_{slope2} represents the interest rate slope above $U_{optimal}$ [9]. The interest rate function is split into two parts to make sure that the liquidity risk will not become a problem for the protocol. This can occur when the utilization rate is high, which refers to the proportion of the total borrowed amount of a specific asset. For example, if users have deposited 1000 USDC and 750 USDC have been borrowed, the utilization rate is 75%. Therefore, the higher the utilization rate, the more liquidity risk there is. It is crucial for the protocol to have a low liquidity risk; otherwise, the protocol may not satisfy further withdrawals or loans. Therefore, when the utilization rate is low, the interest rate increases slowly, and it encourages users to borrow more of the asset. Meanwhile if the utilization rate is high, the interest rate increases significantly faster which encourages borrowers to repay their loans as it becomes more expensive to borrow due to the high interest rate.

4 ANALYSIS

This section will go over the collateralization methods described in the literature review section and make a comparative analysis between Compound Finance and Aave lending protocols. This research primarily relies on the examination of whitepapers of the two protocols, existing literature, and documentation. Furthermore, to evaluate efficiency, robustness, and security of the collateralization mechanisms, real-world data from the Ethereum blockchain will be used, which will then be examined and converted into user-friendly graphs using the Dune analytics service. The data will illustrate the key metrics of the two protocols, such as total value locked, interest rates and liquidations.

4.1 Comparative Analysis

4.1.1 *Loan mechanisms.* The lending and borrowing mechanisms of Compound Finance for the most part, may seem quite similar to the user, yet there are a number of important technical differences to consider when interacting with such protocols.

Upon supplying an asset to the pool, in both protocols the user receives a derivative token, e.g., a user supplies DAI and receives cDAI in Compound Finance or aDAI in Aave. In both protocols, users can earn interest upon supplying an asset, however, there is a difference in how the interest is received. The fundamental difference is that in Aave, the accrued interest is added to the supplied balance; hence, the balance of *aTokens* is incrementing over time. Meanwhile, in Compound Finance, the interest must be claimed manually by calling the function from Figure 8. This adds restrictions due to extra payments that need to be made by the user, since by calling the *claim* function, the user must pay gas fees in order to claim the accrued interest.

```
function claim(address comet, address src, bool shouldAccrue) external
```

Figure 8. Compound Finance claim function

When borrowing assets, both protocols use metrics to calculate the maximum allowed loan. As mentioned in the literature review, Compound Finance uses borrowing capacity, while Aave uses the health factor. Furthermore, Compound protocol uses a collateral factor to determine how much a user can borrow, while Aave uses a liquidation threshold that determines when a loan becomes undercollateralized.

4.1.2 *Tokenization.* One of the main differences between the two protocols is how tokenization is implemented.

In Compound Finance, there is a model with an increasing exchange rate over time. This means that users receive a fixed amount of derivative *cTokens,* which never changes. However, the exchange rate between the tokens increases as interest accrues. Therefore, it is profitable for the user to exchange the *cTokens* back into the underlying asset as the interest has increased. It is also important to note, that whenever a token is launched, its initial exchange rate always begins at 0.02 and each user has always the same *cToken* exchange rate to establish fairness [8].

On the other hand, Aave protocol takes care of the interest rate by increasing the initial amount of *aTokens.* This implies that whenever users deposit certain tokens, they receive their equivalent of *aTokens* at a 1:1 ratio. The interest rate increases the amount of *aTokens* in the account over time, therefore users continuously see a change in the number of their assets. This is different from Compound Finance, where the users only see the effect of the exchange rate when they exchange their *cTokens* for the underlying asset, while in Aave, users are able to track the change in real-time. Such functionality adds a more user-friendly and a more intuitive approach to earning interest from holding a certain derivative asset. Specifically, for new users of decentralized finance, a more visual representation would be preferred to the Compound protocol's approach.

The difference between the models that Compound Finance and Aave implemented is more evident when it comes to minting and/or redeeming tokens. For users, it is more convenient to know that they are receiving their underlying asset at a ratio of 1:1, than when they need to make a calculation with an exchange rate that is continuously updated.

For both protocols, there are the same risks and limitations when redeeming derivative tokens. If a user is currently borrowing an asset by providing collateral in the form of a derivative token, then they might be susceptible to liquidation if they attempt to borrow more derivative tokens. In Compound Finance, the user is at risk of decreasing the collateralization ratio, and in Aave, the user is at risk of decreasing the health factor. Both may lead to the liquidation of a certain asset that becomes undercollateralized.

4.1.2 *Interest rate models.* In Compound Finance, there is an interest rate model that depends on demand and supply from all users in the protocol. It starts at 0% and grows with the utilization rate. This is done to create incentives for lenders to supply even more of their assets once utilization is high. However, one of the drawbacks of the Compound protocol is that the model is at risk of sudden interest rate changes when utilization rates change drastically. For example, Figure 10 illustrates the current utilization rate vs. annual percentage yield (APY) for Ether and Figure 11 illustrates the 90%

utilization rate vs. APY for Ether [10]. Purple line represents borrow APY and green line represents supply APY. If someone were to take a large loan in Ether and drive the utilization rate from 5% to 90%, then borrow APY would increase from 3.17% to 14.74% and supply APY would increase from 0.13% to 10.41%. Therefore, borrow APY would experience an approximate increase of 365%, and supply APY would experience an approximate increase of 7907%. This is a very drastic change that is driven by the fact that the supply interest rate increases with the square of the utilization rate, while the borrow interest rate increases linearly.

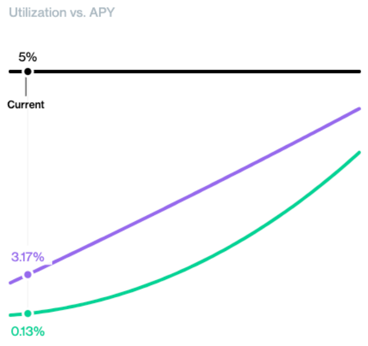

Figure 9. Compound APY against current (5%) utilization for Ether

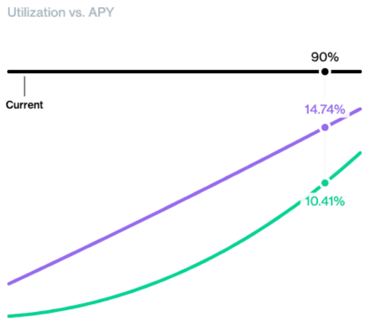

Figure 10. Compound APY against 90% Utilization for Ether

The interest rate model of Aave is different from the one used in Compound Finance. The model for the interest rate includes two separate equations, see equation 4. Unlike Compound Finance, in Aave, the interest rate model is adjusted to each asset individually. The $U_{optimal}$ value is calibrated for every separate asset based on their risk profile [9]. This value is managed by the Aave team, since it's important to adjust the interest rate model based on how risky the asset is. For instance, if a certain asset is very volatile, the $U_{optimal}$ value can be set lower, which will discourage over-borrowing. With such parameters, the users are protected from liquidations that may occur with sudden price drops. Furthermore, Aave allows two different interest rate models: variable and stable, whereas Compound Finance only has a stable-rate model. Such functionality allows Aave users to be more flexible with regards to their capital since they can switch between variable and stable rates at any time. Figure 12 illustrates the interest rate curves of APY against utilization rate in Aave protocol for Ether.

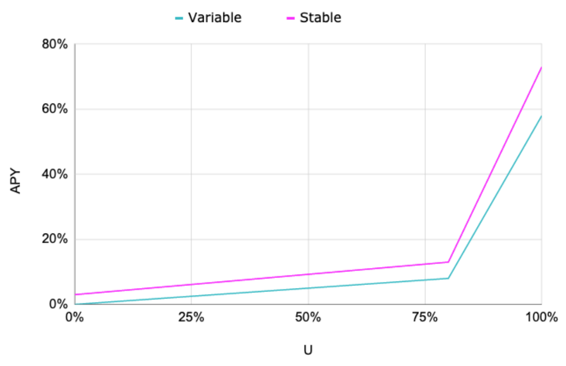

Figure 11. Aave APY against Utilization for Ether

4.1.2 *Liquidation processes.* Both protocols approach the liquidation process in a similar fashion, where a user's position may be liquidated if the value of their borrowed assets surpasses the allowed threshold.

To further explore the liquidation processes of Aave and Compound Finance, the following graphs were generated. They demonstrate the daily volume of liquidations on both protocols over the course of 30 days. The data for the graphs was collected directly from the Ethereum blockchain with the use of the Dune Dashboard service to model the data into user-friendly graphs [11], see Figure 13 and Figure 14.

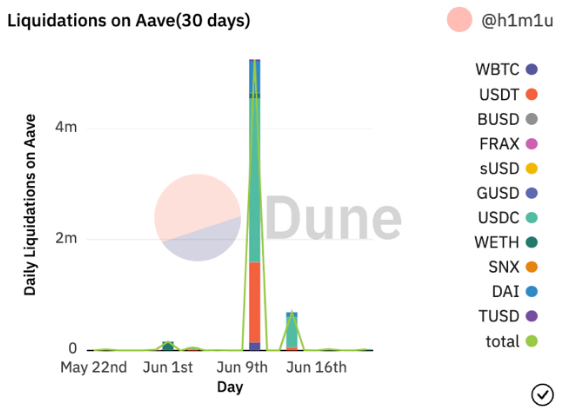

Figure 12. Liquidations on Aave (30 days)

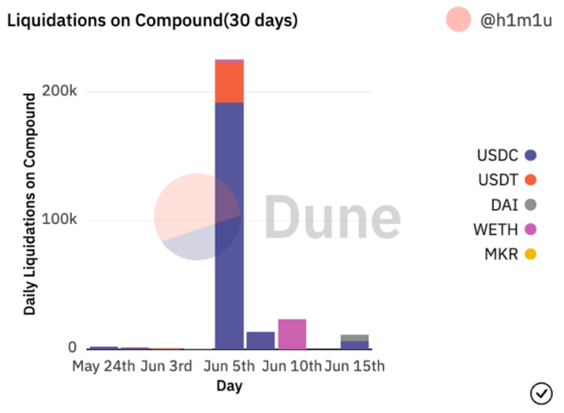

Figure 13. Liquidations on Compound (30 days)

As can be seen from Figures 13 and 14, Aave surpasses Compound Finance in the number of liquidations for the past 30 days. Additionally, it is important to note the number of different tokens, for which positions were liquidated on Aave and on Compound. For Compound, there were liquidations related to 5 tokens only, however, for Aave there were liquidations for over 11 tokens. This is an important distinction that illustrates the aforementioned advantage of Aave in terms of the interest rate model, hence, users tend to take loans by providing various assets as collateral.

In addition to the liquidation volume, it is important to consider the total value locked (TVL), since more TVL implies a greater possibility for liquidations to occur. Figure 15 illustrates the TVL for Aave for the past year, and Figure 16 illustrates the TVL for Compound Finance for the past year.

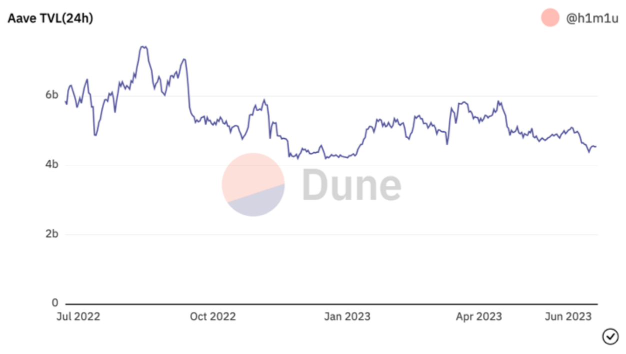

Figure 14. Aave TVL against time (24h)

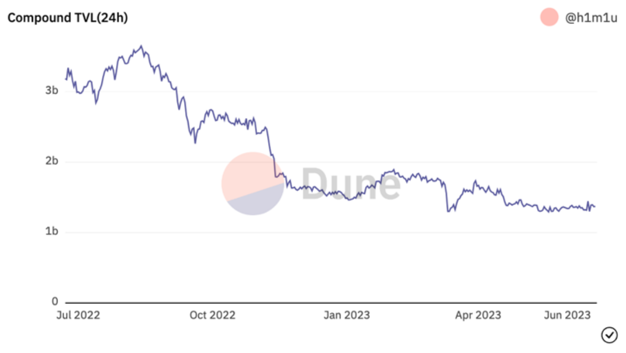

Figure 15. Compound Finance TVL against time (24h)

TVL indicates the overall volume of assets in the protocols. Therefore, a higher TVL implies a presence of a greater volume of loans in the protocol, and consequently, more positions are at risk of being undercollateralized and ultimately liquidated. However, it is crucial to mention that a larger number of liquidations does not necessarily qualify as a negative aspect of a platform. It demonstrates the protocol's ability to effectively manage risk and ensure the safety of funds within the protocol. Especially, if the protocol's functionality is not affected by large liquidations.

5 CONCLUSION

This paper compares the collateralization mechanisms of the two largest decentralized finance platforms that allow users to lend and borrow assets. The approach to implementing collateralization mechanisms and their functionality were studied in the literature review section of the paper, where the main technical aspects of lending and borrowing were studied. Furthermore, a comparative analysis was conducted in the analysis section of the paper, where Compound Finance and Aave protocols were compared based on their approaches to loan mechanisms, tokenization, interest rate models, and liquidity processes. Based on the findings, in terms of efficiency, robustness, and security, Aave protocol demonstrates an advantage with several features such as automatic reinvestment of interest, two-slope interest rate models, and individualized asset risk profiles. However, the decentralized finance space is constantly evolving, and new protocols are being developed each year. Hence, it is feasible to perform further research to compare new protocols with existing platforms such as Aave or Compound Finance. Furthermore, the relevant data of the two platforms can be reproduced by running the previously mentioned Dune dashboard [11].

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my supervisor, Dr. Georgiana Caltais, for helping me with coordinating my research project. Additionally, I would like to thank my track chair, Dr. Peter Lammich, and students from the software technology and formal methods track for providing useful feedback during track meetings.

REFERENCES

[1] Sirio Aramonte, Sebastian Doerr, Wenqian Huang, and Andreas Schrimpf. 2022. DeFi lending: intermediation without information? *BIS Bull.* (June 2022). Retrieved May 1, 2023 from https://ideas.repec.org//p/bis/bisblt/57.html
[2] Adam Bavosa. 2022. The Compound III Liquidation Guide. *Compound Community Forum.* Retrieved June 17, 2023 from https://www.comp.xyz/t/the-compound-iii-liquidation-guide/3452
[3] Vitalik Buterin. 2014. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. *Ethereum.org.* Retrieved May 31, 2023 from https://ethereum.org/en/whitepaper/
[4] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008).
[5] Andrei-Dragoş Popescu. 2020. DECENTRALIZED FINANCE (DEFI) – THE LEGO OF FINANCE. *Ed. Sitech* 7, 1 (2020), 321–349.
[6] Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. 2021. An Empirical Study of DeFi Liquidations: Incentives, Risks, and Instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference*, 336–350. DOI:https://doi.org/10.1145/3487552.3487811
[7] Camila Russo. 2022. What Is Decentralized Finance?: A Deep Dive by The Defiant. *CoinMarketCap Alexandria.* Retrieved May 31, 2023 from https://coinmarketcap.com/alexandria/article/what-is-decentralized-finance
[8] Compound.Whitepaper.pdf. Retrieved June 10, 2023 from https://compound.finance/documents/Compound.Whitepaper.pdf
[9] Aave. Retrieved June 18, 2023 from https://docs.aave.com/hub/
[10] Compound Finance. Retrieved June 20, 2023 from https://compound.finance
[11] Compound Finance and Aave Dashboard. Retrieved July 2, 2023 from https://dune.com/h1m1u/compound-finance-and-aave/3cf16ce9-14ea-457c-9f14-df073854c4cf