# Unbound.Finance Threat Model

Date: 12/11/2020
Author: Peter Kacherginsky

# Introduction

Unbound.Finance is a DeFi platform which relies on the Ethereum blockchain and smart contracts in order to create a decentralized stablecoin called UND. A malicious or accidental exploitation of a vulnerability in the smart contracts or other components of the system presents an existential risk to Unbound.Finance including loss of funds, inability to continue operations, reputation loss, and others.

The document aims to record the results of the threat modeling exercise conducted during the week of November 30th, 2020 in order to identify threats to Unbound.Finance's security and advise on clear next steps to further strengthen it. It is not the aim of a threat modeling exercise to assess a point in time security stance of the system.

The advantage of developing a threat model prior to executing specific mitigating steps (e.g. code audits) is that it allows for rational and clear security decision making while avoiding knee-jerk security actions which may not necessarily improve the security posture of the system.

The threat model includes the following items:

- All critical targets and components of the system.
- Existing security controls and their efficacy.
- Threat agents which may be motivated to attack the application.
- A comprehensive list of threats to each component of the system.
- A prioritized list of mitigations based on threat impact and likelihood.

The following actions were performed as part of the exercise:

- Team interviews to understand processes, components, smart contract and DAPP dependencies, and the existing security practices.
- Targeted source code review with the aim of identifying or validating specific high impact threats inherent to DeFi space (e.g. flash loans, price oracle manipulation, etc.)
- Diagramming the entire Unbound.Finance system and smart contract interactions including trust boundaries, data flows, actors, and other components.

# Tables of Contents



# Engagement Participants

The following individuals have actively supported the creation of the threat model through numerous interviews, code walkthroughs, and threat modeling exercises.

- Peter Kacherginsky - security auditor
- Tarun Jaswani - project lead
- Chetan Badhe - product development
- Adesh Kolte - security researcher

# Threat Summary

The engagement identified **17 unique components** which were used to map **11 threats** with the following severity.

It is important to note that the table below identifies **all threats to the system regardless of their mitigation status** in order to provide the most complete view of the system's threat space. It is specifically **not designed to attest to the point in time risk profile or "safety" of the system**, but to inform which areas need the most security investment and consideration as the project evolves.

The engagement includes a prioritized list of **33 mitigations** which can be used to substantially reduce the severity of the identified threats.

# Scope

## In Scope

- Unbound.Finance Smart Contracts.
    - https://github.com/unbound-finance/UnboundContracts/ (Master:c44b3ec8cc3306a7cea3380b1d0f2d62d5f104b1)
- Project's Github repository:
    - https://github.com/unbound-finance
- DAPP web server and application.
    - https://testnet.unbound.finance/
- Administrator Ethereum key management.
- Administrator web server and github account management.

## Out of Scope

- Ethereum blockchain vulnerabilities. (e.g. reorgs, double spends)
- Ethereum Virtual Machine vulnerabilities.
- AMM Contract vulnerabilities.
- Incorrect usage or misconfigurations by users.
- Self-custody and 3rd party user wallets and key management.
- Economic viability of Unbound.Finance platform.

## Scope Modifications

- Liquidation replacement engine was removed from the original scope since the threat was determined primarily economic rather than security.

# Existing Security Controls

The information collected below is based on both interviews with the development team and manual review of provided artifacts:

- Web service and Github accounts are protected using 2FA accounts.
- Smart Contract testing using Truffle Suite and FireFly coverage verification.
- Passing results from security automation tools (Slither, MythX).
- General security audit using a 3rd party tool.

# Threat Models

Threat models below expose components and their interactions involved in the operation of Unbound.Finance system.

## Unbound.Finance System Threat Model

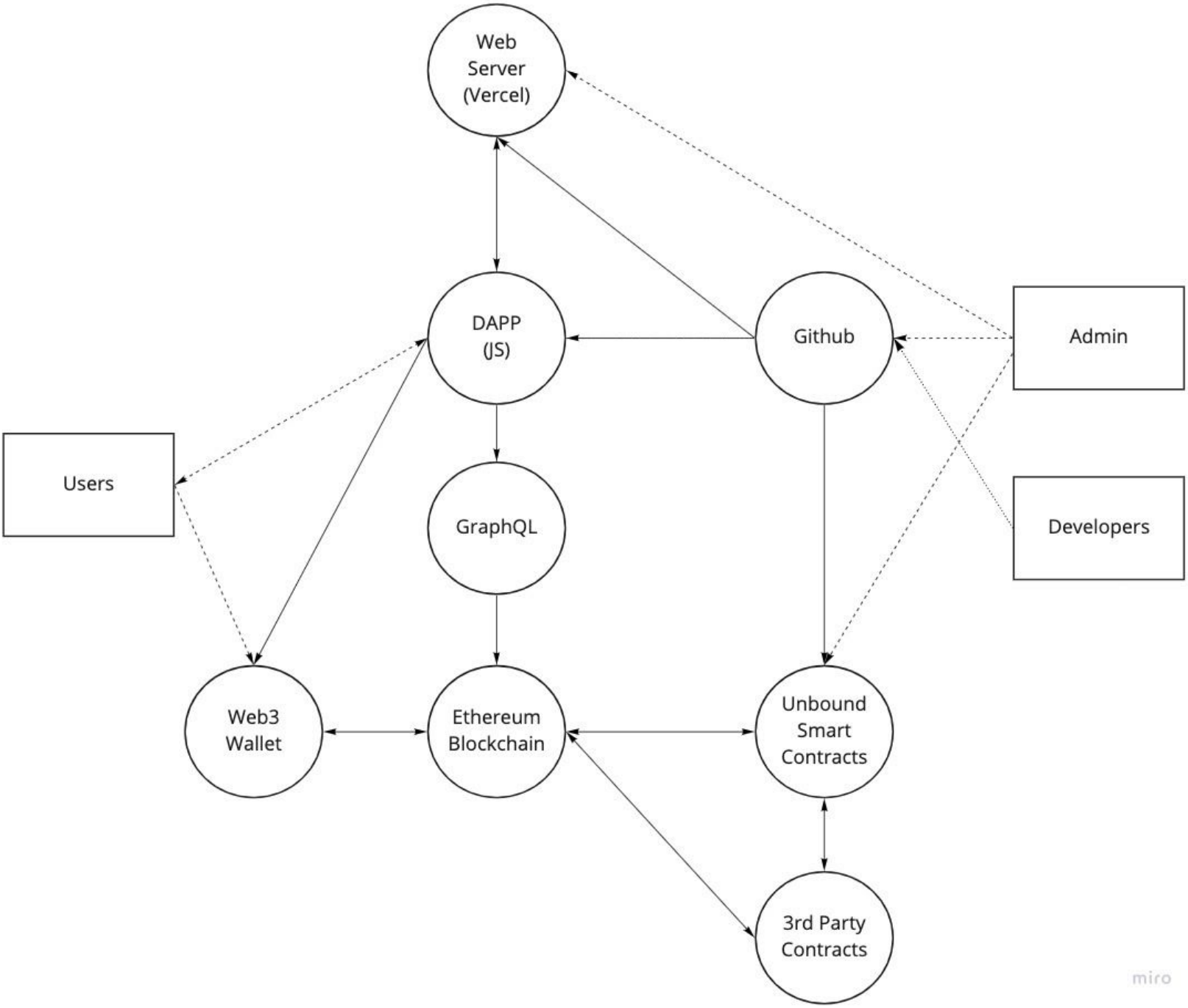
The threat model shows the overview of the overall Unbound.Finance system including developers. The system involves the following key components:

- Web Server holds the static DAPP application which interacts with Ethereum blockchain using GraphQL.
- Unbound Smart Contracts are deployed on Ethereum blockchain and interact with various 3rd party contracts (e.g. Uniswap)
- All smart contract and DAPP source code is stored on Github.
- All smart contract interactions are made either directly on Ethereum blockchain or using a DAPP interface.

The system involves the following actors:

- Users interact with the Unbound.Finance system using web3 wallet software or by visiting the DAPP website to obtain static blockchain data.
- Administrators deploy smart contract code on the Ethereum blockchain, manage smart contracts using owner privileges, manage Github repository and static web server.
- Developers push the smart contract and DAPP code to Github server.

*Smart Contract specific threat model is covered in the next section.*




| Threat ID | Component | Threat Name | Sev. | Prob. | Threat Description | Mitigations |
|---|---|---|---|---|---|---|
| U-001 | Web Server | Web Server is compromised | Med | Med | Compromised web server is used to prevent users from easily interacting with the | - Restrict access to the administrator panel.<br>- Reduce the number |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | DAPP.<br><br>A modified DAPP can serve wrong smart contract addresses to Web3 clients which may be able to steal funds from users. | of people with access.<br>- Audit and revoke unnecessary access (e.g. page designers)<br>- Enforce 2FA access. (no SMS).<br>- Prepare incident playbook<br>- Contact information<br>- Account recovery procedure<br><br>Additional Notes:<br>- Ensure a single source of truth for Unbound smart contract addresses and make sure it's protected from tampering (e.g. Notion, Gitbook). |
| U-002 | GraphQL | GraphQL is compromised | Low | Low | Compromised service may serve wrong blockchain data which may be used in social engineering attacks against users. | - Rely on multiple sources of data. (e.g. CoinGecko)<br>- Monitor and verify that GraphQL is serving correct data. |
| U-003 | Github | Github repository is compromised | High | Low | Compromised Github repository may result in tampering with the following sensitive repositories:<br>- DAPP source code. Smart Contract addresses may be changed to steal funds from users.<br>- Smart Contract source code. Backdoors may be introduced before on-chain deployment. | - Restrict write access to the Github repositories<br>- Reduce the number of people with access.<br>- Audit and revoke unnecessary access (e.g. former developers)<br>- Enforce 2FA access. (no SMS).<br>- Lock master branch to trusted accounts.<br>- Require approvals from multiple accounts before committing to master. |

## References

- https://testnet.unbound.finance/
- https://docs.unbound.finance/introducing-unbound-finance
- https://github.com/unbound-finance

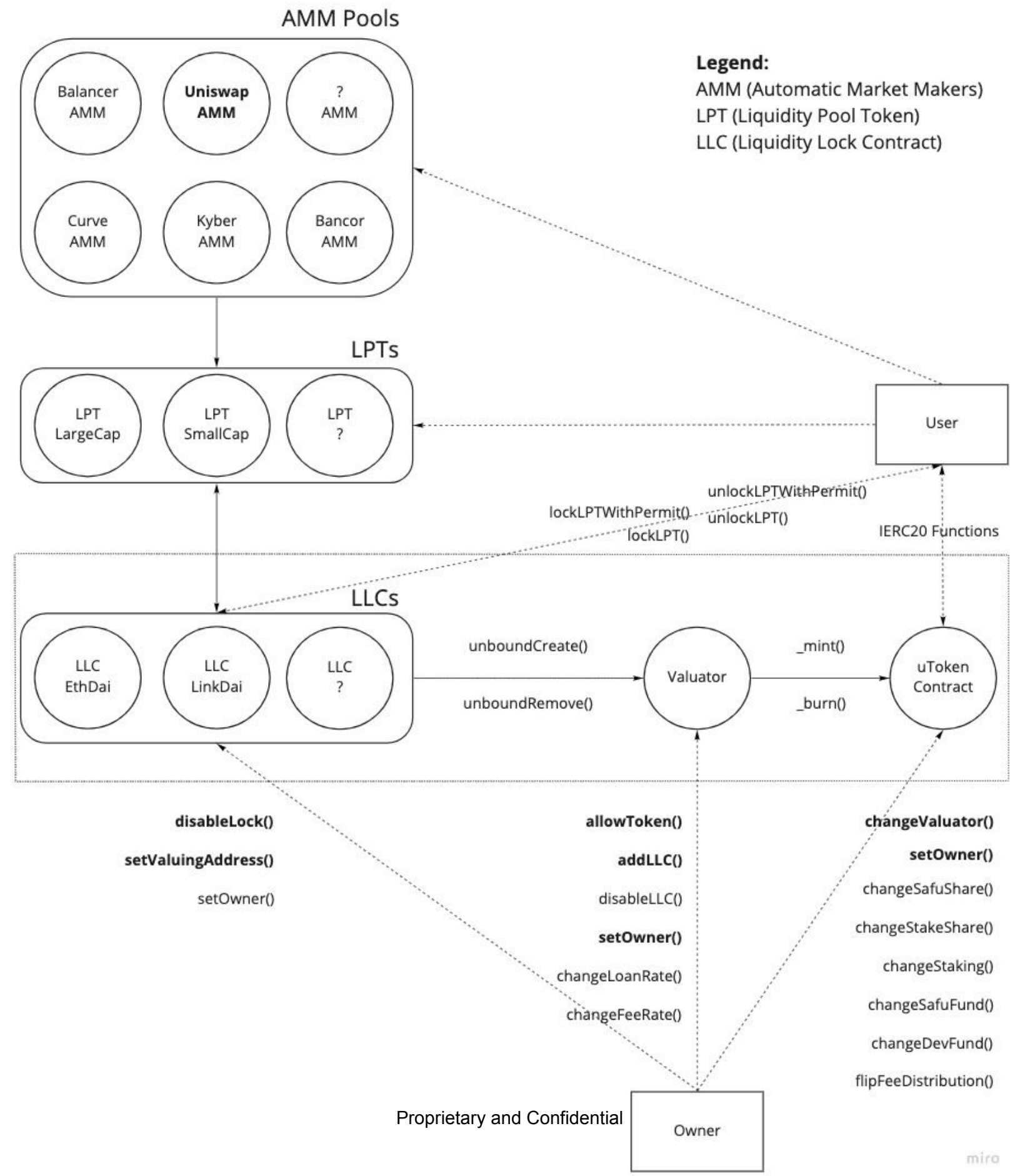
# Unbound.Finance Smart Contracts Threat Model

The threat model below focuses on the smart contract interactions within the Unbound.Finance system. Unbound.Finance has three key components:

- Liquidity Lock Contracts (LLCs) provide unique logic for each Liquidity Pool Token (LPT) pair to be locked as a collateral in return for the configurable amount of Unbound tokens. Each LPT has a unique LLC associated with it.
- Valuator contract is responsible for managing multiple LLC contracts, managing lending parameters, and issuing Unbound tokens to the borrower.
- uToken is an ERC-20 token borrowed by the LPT owners.

The smart contract system includes two actors:

- Users interact with the Liquidity Lock Contracts (LLCs) to lock their Liquidity Pool Tokens (LPTs) and manage uToken assets.
- Owner account deploys, manages, and interacts with LLCs, Valuator, and uToken contracts by adding new trusted LLCs and uToken contracts to the system, toggling an emergency lock on the LLCs, adjusting loan and fee rates, and other administrative actions.



| Threat ID | Component | Threat Name | Sev. | Prob. | Threat Description | Mitigations |
|---|---|---|---|---|---|---|
| U-004 | Liquidity Lock Contracts (LLCs) | Incorrect uToken minting calculation | High | Med | *lockLPT()* and l*ockLPTWithPermit()* calculate the amount of *uTokens* to mint based on the value of the user's locked LPTs. Incorrect calculation of the value may lead to an invalid amount getting credited. | Ensure correctness of getValue() call under all conditions and every future AMM/LP addition:<br>- Implement tests with 100% code coverage for handling of the decimal matching logic.<br>- Implement testing with 100% coverage for correct calculation of the LP value used to mint UND, uETH, or other synthetic assets.<br>- Perform a targeted audit of the uToken amount calculation. |
| U-005 | Liquidity Lock Contracts (LLCs) | Flash Loan assisted arbitrage exploitation | High | High | Unbound.Finance price calculation logic depends on a unique AMM pool implementation which may be susceptible to arbitrage opportunities. These can be further enhanced with a large flash loan deposit to break the reference token peg. | - Implement monitoring and alerting on transactions beyond a safe threshold. (e.g affecting more than 50% of a given LP pair).<br>- Limit the ability to perform multiple sensitive actions in a single transaction (e.g both minting and burning, future governance actions, etc.).<br>- Introduce delays between multiple sensitive actions (e.g. must wait N blocks before unlocking funds).<br>- Use trusted 3rd party price oracles to calculate LPT prices which are resilient to flash price fluctuations. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | Sensitive Actions Include:<br>- Locking and unlocking LPTs<br>- Minting and burning uTokens |
| U-006 | Liquidity Pool Tokens (LPTs) | Vulnerable or malicious LPT pair | Med | Med | A vulnerability or a dangerous functionality in an LPT pair breaks Unbound locking/unlocking and other system logic resulting in the loss of funds. | Review every new Liquidity pool token to ensure reliable calculation of the following:<br>- Correct reporting of the base asset reserves.<br>- Correct reporting of the total balance.<br>- Pool specific actions which may affect the value of locked LPTs. (e.g. USDC/USDT freezing/blacklisting) |
| U-007 | Liquidity Pool Tokens (LPTs) | Insufficient LP liquidity | Med | Med | Insufficient LP liquidity may make LPTs susceptible to price manipulation. | Establish a minimum acceptable liquidity tolerance for all newly onboarded pools. |
| U-008 | AMM Platforms | Vulnerable or malicious AMM platform | Med | Med | A vulnerability in an AMM platform breaks Unbound locking/unlocking and other system logic resulting in the loss of funds. | Review every new AMM platform to ensure the following:<br>- Strong platform security security features, maturity, and reliability (e.g. presence of audits performed, major incidents in the past, mitigations applied, etc.).<br>- The ability to adversary affect |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | liquidity pool's availability. (e.g. pausing) |
| U-009 | Valuator | Valuator uToken confusion | High | Low | Deployed LLCs specify which uToken to use for minting which can be abused when the wrong type is selected (e.g. UND vs uETH or other future assets). | - Create 1:1 coupling of LLC to have a single uToken address permitted to use in the Valuator contract (e.g. create a uToken entry in the LLCStruct). |
| U-010 | Owner | Compromised Owner keys | High | Med | A malicious actor with Owner keys can perform the following dangerous actions:<br>- Permanently disable the system.<br>- Substantially alter the behavior of the system.<br>- Burn tokens from an arbitrary account.<br>- Mint arbitrary amounts of UND tokens. | - Minimize superuser privileges.<br>- Protect Owner keys with ⅗ multi-sig.<br>- Use hardware wallets for key storage.<br>- Incident and Disaster recovery plan.<br>- Key backup in a secure location.<br>- Future:Implement DAO to manage all dangerous functions. |
| U-011 | LLC, Valuator, uToken | Unknown Vulnerabilities | Med | Med | Vulnerabilities discovered in any of the Unbound.Finance system contracts. | - Implement an upgrade mechanism:<br>- LLC can update the Valuing address.<br>- Valuing can update the uToken address.<br>- Prepare an incident plan with different scenarios |

# References


- https://github.com/unbound-finance
- https://docs.unbound.finance/documentation/whitepaper

# Threat Mitigations

Threat mitigations below provide specific actions to mitigate threats described in the previous sections. All mitigations are prioritized based on their impact and likelihood (e.g. complexity, opportunity to exploit). Threat severity levels are defined as follows:

- **Critical Threats** - both high impact and likelihood of occurrence.
- **Severe Threats** - high security impact, but less likely to occur.
- **Substantial Threats** - high security impact, but very unlikely to occur.
- **Moderate Threats -** medium security impact and high to medium likelihood of occurrence.
- **Low Threats** - medium security impact and unlikely to occur.

In addition to threat severity, each mitigation has a cost parameter which approximates the amount of engineering time necessary to implement:

- **High -** ~1 month
- **Med** - ~ 1 week
- **Low** - ~ 1 day

The above can be used to quickly identify high impact and easy to mitigate threats to target in the immediate future while planning for larger efforts in accordance with their severity levels.

## Critical Threats

Critical threats have both high impact and high likelihood of occurrence. These threats are most likely to cause significant financial and/or reputational damage.

| Threat ID | Component | Threat Name | Mitigating Action | Cost | Status |
|---|---|---|---|---|---|
| U-005 | Liquidity Lock Contracts (LLCs) | Flash Loan assisted arbitrage exploitation | Limit the ability to perform multiple sensitive actions in a single transaction (e.g both minting and burning, future governance actions, etc.).<br><br>Sensitive Actions Include:<br>- Locking and unlocking LPTs<br>- Minting and burning uTokens | Low | |
| | | | Introduce delays between multiple sensitive actions (e.g. must wait N blocks before unlocking funds). | Low | |

| | | | | | |
|---|---|---|---|---|---|
| | | | Use trusted 3rd party price oracles to calculate LPT prices which are resilient to flash price fluctuations. | Med | |
| | | | Implement monitoring and alerting on transactions beyond a safe threshold. (e.g affecting more than 50% of a given LP pair). | High | |
| | | | Optional: Introduce upper limit for uToken minting. | Low | |

## Severe Threats

Severe threats have a high security impact; however, less likely to occur. These threats may cause significant financial and/or reputational damage, but less likely to happen.

| Threat ID | Component | Threat Name | Mitigating Action | Cost | Status |
|---|---|---|---|---|---|
| U-004 | Liquidity Lock Contracts (LLCs) | Incorrect uToken minting calculation | Implement tests with 100% code coverage for handling of the decimal matching logic. | Med | Partial:<br><br>Increase code coverage to 100%. |
| | | | Implement testing with 100% coverage for correct calculation of the LP value used to mint UND, uETH, or other synthetic assets. | Med | Partial:<br><br>Increase code coverage to 100%. |
| | | | Perform a targeted audit of the uToken amount calculation. | Med | |
| U-010 | Owner | Compromised Owner keys | Minimize superuser privileges. | Med | |
| | | | Protect Owner keys with ⅗ multi-sig. | Low | |
| | | | Use hardware wallets for key storage. | Low | |
| | | | Incident and Disaster recovery plan. | Med | |
| | | | Key backup in a secure location. | Med | |
| | | | Implement DAO to manage all dangerous functions. | High | |

## Substantial Threats

Substantial threats have a high security impact; however, very unlikely to occur. These threats may cause significant financial and/or reputational damage, but are very unlikely to happen.

| Threat ID | Component | Threat Name | Mitigating Action | Cost | Status |
|---|---|---|---|---|---|
| U-003 | Github | Github repository is compromised | Restrict write access to the Github repositories | Low | |
| | | | Reduce the number of people with access. | Low | |
| | | | Audit and revoke unnecessary access (e.g. former developers) | Low | |
| | | | Enforce 2FA access. (no SMS). | Low | |
| | | | Lock master branch to trusted accounts. | Low | |
| | | | Require approvals from multiple accounts before committing to master. | Low | |
| U-009 | Valuator | Valuator uToken confusion | Create 1:1 coupling of LLC to have a single uToken address permitted to use in the Valuator contract (e.g. create a uToken entry in the LLCStruct). | Low | |

## Moderate Threats

Moderate threats have a medium security impact and have a high to medium likelihood of occurrence. These threats pose some financial and/or reputational damage and are somewhat likely to happen.

| Threat ID | Component | Threat Name | Mitigating Action | Cost | Status |
|---|---|---|---|---|---|
| U-001 | Web Server | Web Server is compromised | Restrict access to the administrator panel. | Low | |
| | | | Reduce the number of people with access. | Low | |
| | | | Audit and revoke unnecessary access (e.g. page designers) | Med | |
| | | | Enforce 2FA access. (no SMS). | Low | |

| | | | | | |
|---|---|---|---|---|---|
| | | | Prepare incident playbook<br>- Contact information<br>- Account recovery procedure | Med | |
| | | | Ensure a single source of truth for Unbound smart contract addresses and make sure it's protected from tampering (e.g. Notion, Gitbook). | Low | |
| U-006 | Liquidity Pool Tokens (LPTs) | Vulnerable or malicious LPT pair | Review every new Liquidity pool token to ensure reliable calculation of the following:<br>- Correct reporting of the base asset reserves.<br>- Correct reporting of the total balance.<br>- Pool specific actions which may affect the value of locked LPTs. (e.g. USDC/USDT freezing/blacklisting) | Med | |
| U-007 | Liquidity Pool Tokens (LPTs) | Insufficient LP liquidity | Establish a minimum acceptable liquidity tolerance for all newly onboarded pools. | Low | |
| U-008 | AMM Platforms | Vulnerable or malicious AMM platform | Review every new AMM platform to ensure the following:<br>- Strong platform security security features, maturity, and reliability (e.g. presence of audits performed, major incidents in the past, mitigations applied, etc.).<br>- The ability to adversary affect liquidity pool's availability. (e.g. pausing) | Med | |
| U-011 | LLC, Valuator, uToken | Unknown Vulnerabilities | Implement an upgrade mechanism.<br><br>References:<br>https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable | High | Partial:<br><br>The contract system can be upgraded as follows:<br>- LLCs can be upgraded to a new Valuator address.<br>- Valuator can add or remove new uToken and LLC contracts.<br>- uToken can be pointed to a new |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Valuator contract. |
| | | | Prepare an incident plan with different scenarios | High | |

## Low Threats

Low threats have medium to low security impact. These threats pose minimal financial and/or reputational damage.

| Threat ID | Component | Threat Name | Mitigating Action | Cost | Status |
|---|---|---|---|---|---|
| U-002 | GraphQL | GraphQL is compromised | Rely on multiple sources of data. (e.g. CoinGecko) | Med | |
| | | | Monitor and verify that GraphQL is serving correct data. | Med | |

# Appendix A - References

https://github.com/crytic/building-secure-contracts/tree/master/development-guidelines

https://github.com/securing/SCSVS

https://consensys.github.io/smart-contract-best-practices/

# Appendix B - Auditor Qualifications

Peter Kacherginsky is an active member of the Blockchain Security community where he presents at Defcon and other conferences, publishes a weekly Blockchain Threat Intelligence newsletter, and leads the Capture the Coin CTF. Professionally he is employed at a major cryptocurrency exchange where he spends his day assessing security of standalone blockchains and smart contracts, developing blockchain monitoring and automation solutions. Before getting bit by the cryptocurrency bug, Peter worked as a reverse engineer, penetration tester, and incident responder. In the past, he taught malware analysis classes, developed a number of tools used to aid in malware analysis and penetration testing tasks. Peter created IDA Sploiter which won the IDA Pro plug-in contest in 2014. He also developed a number of open source security tools such as FLARE VM, FakeNet-NG, DNSChef, PACK (Password Analysis and Cracking Kit), SEAT (Search Engine Assessment Tool), WiCrawl and others. A number of these tools are included in the Kali Linux and other security distributions.