Dai

Security Assessment

Dai Smart Contracts
March 23, 2018

Prepared For:
Rain Break | *DappHub*
rain@dapp.org

Prepared By:
JP Smith | *Trail of Bits*
jp@trailofbits.com

Ben Perez | *Trail of Bits*
benjamin.perez@trailofbits.com

Executive Summary

From October 15th to November 16th, Trail of Bits assessed the multi-collateral Solidity smart contracts in the Dai platform. Two engineers conducted this assessment over the course of 6 person-weeks.

Since the contracts being assessed had already been formally verified using the K Framework, the audit focused on attack vectors not covered by the verification process such as denial of service attacks and other blockchain-level attacks. We also initiated the development of a new tool which is capable of statically proving certain algebraic invariants hold after a function is called. This tool is discussed in Appendix D.

During the course of the audit, Trail of Bits discovered two potential issues with the Dai codebase. The first is a potentially too low liveness assumption in the auction code, and the second is a potential failure of the auction system during eclipse attacks. Neither are of high severity, and the only mitigation recommended in the short term is modifying a single parameter.

Engagement Goals & Scope

The goal of the engagement was to evaluate the security of the multi-collateral Dai smart contracts with a focus on answering the following questions:

- Did formal verification of the Solidity code successfully catch function-level bugs?
- Are there inter-contract vulnerabilities not detected by formal verification?
- What blockchain-level attacks are possible?

Coverage

We reviewed:

- bite.sol
- drip.sol
- flap.sol
- flip.sol
- flop.sol
- frob.sol
- heal.sol
- join.sol
- move.sol
- tune.sol

We looked for common Solidity flaws, such as integer overflows, re-entrancy vulnerabilities, and unprotected functions. We also looked for more nuanced flaws, such as logical errors, denial of service attacks, and race conditions. We also checked that the fundamental equations of Dai always hold.

Project Dashboard

Application Summary

Name	Dai
Type	Ethereum smart contract
Platform	Solidity

Engagement Summary

Dates	October 15th to November 16th
Method	Whitebox
Consultants Engaged	2
Level of Effort	6 person-weeks

Vulnerability Summary

Total High Severity Issues	0	
Total Medium Severity Issues	0	
Total Low Severity Issues	2	■■
Total Informational Severity Issues	2	■■
Total Undetermined Severity Issues	0	
Total	4	

Category Breakdown

Denial of Service	1	■
Race Condition	1	■
Code Quality	2	■■
Total	4	

Recommendations Summary

This section aggregates all the recommendations made during the engagement. They are split into short-term recommendations, which address issue's immediate causes, and long-term recommendations, which include adjustments in the development process.

Short Term

❑ **Increase `ttl in flip/flap/flop contracts`.** Auctions might settle for less than the fair market value due to Ethereum network congestion.

Long Term

❑ **Be conservative with Ethereum blockchain liveness assumptions.** When the Ethereum network is congested, confirming transactions in a timely manner is difficult.

❑ **Be aware of eclipse attacks and other unintended consequences of network partitioning.** Users might be able to create malicious transactions in such circumstances that undermine the economic stability of the Dai platform.

Findings Summary

#	Title	Type	Severity
1	Timestamp dependence in auctions enables denial of service	Denial of Service	Low
2	Auction identification system enables eclipse attacks	Race Condition	Low

1. Timestamp dependence in auctions enables denial of service

Severity: Low
Difficulty: High
Type: Denial of Service
Finding ID: TOB-DSS-001
Target: Flip, Flap, Flop

Description
When auctioning off assets, bids are finalized after some interval of time, `ttl`, has passed. During periods of extreme network congestion, it can be challenging to confirm a transaction within three hours, especially in the presence of large numbers of unconfirmed transactions with too-low gas prices.

Exploit Scenario
While an auction is taking place, cryptokitties launches a sequel. The Ethereum network begins undergoing incredible congestion, and common APIs also are shut down due to denial of service. Only sophisticated users who can craft their own transactions are able to use the auction, and the high gas price makes bidding extremely expensive regardless of whether these bids win. Auctions thus settle for less than the fair market value.

Recommendation
Increase `ttl`, or have a mechanism for governance to extend auction time.

Be conservative with Ethereum blockchain liveness assumptions.

2. Auction identification system enables eclipse attacks

Severity: Low
Difficulty: High
Type: Race Condition
Finding ID: TOB-DSS-002
Target: Flip, Flap, Flop

Description
Auctions are identified via the `kicks` variable, an integer that starts at zero and increments monotonically. An attacker with the ability to conduct a temporary eclipse attack can exploit this to create two auctions in two temporary forks of the Ethereum chain, then take valid bids on one and apply them to the other.

This attack is mostly mitigated by the inclusion of both `bid` and `lot` parameters in all bidding function types, but multiple-auction interactions could still be used to potentially defraud bidders. Preventing this in the current Ethereum execution environment is not, however, necessarily possible.

Exploit Scenario
Alice has the ability to temporarily partition the Ethereum network. On the Ethereum mainnet, she initiates a large volume of collateral auctions of a particular type, as cost of each will be relatively low (approximately the liquidation fee). She groups them in two “waves” separated by closing time.

As the first wave of auctions approach closing time, she partitions the network and creates incredibly high bids for all of them. This convinces observers (particularly automated ones) of a lack of demand for Dai (compared to the underlying asset), and rational observers will update their priors regarding that trading pair. This effect is amplified when the underlying asset has low capitalization or is illiquid for other reasons.

Once the second wave of auctions approach closing time, these observers will bid higher than they normally would. These bids can be translated over to mainnet, netting Alice some extra money. While it’s possible she lost money in liquidation fees, she could still make enough back to compensate, and the observers lose money regardless.

Recommendation
Be aware of eclipse attacks and other unintended consequences of network partitioning. Consider sealed-bid auctions, should they become more practical in the Ethereum execution model.

References
- Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network (*N.B.*: the attack described in the paper is no longer viable)

A. Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for client's reputation, moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details or must discover other weaknesses in order to exploit this issue

B. Code Quality Recommendations

The following recommendations are not associated with specific vulnerabilities. However, they enhance code readability and may prevent the introduction of vulnerabilities in the future.

- In the `flip`/`flap`/`flop` contracts, the `kicks` variable is never initialized
- `Join.sol` uses Solidity 0.4.20 while the other contracts use 0.4.24

Appendix C: On-chain Governance

This project intends to make high-impact decisions via on-chain governance. This is an extremely challenging endeavor from both an engineering and community relations standpoint. While the governance system itself is outside the scope of this audit, this appendix outlines several general concerns with on-chain governance system that could introduce material risk to this product.

Threat Model

It is reasonable to expect some incentive misalignment with regards to voting. Suppose some DAO is choosing between investing in two tokens of similar value. The value of voting correctly to each DAO stakeholder is very small, but the value to someone holding a large amount of one of the tokens under consideration is very large. We might then expect to see influence campaigns by such token holders, and mitigating such campaigns is nontrivial.

Ideally, our system is secure both cryptographically and economically. Cryptographic security has to do with the voting protocol itself, and economic security has to do with incentive alignment for voters. A failure in either domain can cause consequences from erosion of trust to total organizational failure, as governance decisions cease to be made rationally (from the point of view of the organization).

Cryptographic security

On-chain voting processes typically create a cryptographically verifiable ledger of who voted for which option. This allows for coercion, where an adversary can promise to observe the final voter roll and dole out rewards to voters they agree with and punishment to voters they disagree with.

To avoid this, cryptographers design electronic voting systems with a property called "receipt-freeness." This means that no one can produce a "receipt" stating they voted a particular way, and thus the above coercive strategies cannot work (or at least, cannot scale). However, such systems should also be "verifiable" (that is, everyone can know their vote counted), "correct" (each authorized voter gets exactly one vote, which only they control), and anonymous.

Several solutions for voting protocols are known (see first reference), but it is unclear how to migrate these onto the Ethereum blockchain. It may even be necessary to run a sidechain strictly for governance, and reify its results onto the Ethereum blockchain via state channels or something similar. To our knowledge, no operationalized solutions exist.

Economic Security

Another issue with specifically token-based blockchain voting systems is the separation of equity and voting rights. In “real life” shareholder voting, you must possess voting shares to vote, and this possession implies full exposure to the movements in share value. In blockchain voting systems, this is not the case.

One strategy an adversary can take is creating a voting smart contract, which locks tokens for some amount of time, uses them to vote, and then returns them with a premium paid. Users can inspect the smart contract code and be sure they will recover their tokens after lending them, and in cases like the investment decision discussed above, can be convinced to vote irrationally.

Even more sophisticated attacks are possible. For instance, an attacker could use trusted computing technology (such as Intel SGX) to create programs that can only access cryptographic secret material for voting, not share transfer. This means that bought votes appear indistinguishable from real ones, and organization of vote buying can happen in a distributed, indetectable manner. Phil Daian of Cornell’s IC3 has outlined this topic in much more depth in his work on “Dark DAOs” (second reference).

Mitigating these threats is even harder than mitigating the cryptographic ones, and there are few known theoretical solutions, none of which (to our knowledge) have ever been operationalized. Perhaps by using cryptographic multi-party computation it’s possible to inseparably couple voting rights and transfer rights, but building such a system is highly nontrivial.

References

- https://crypto.stanford.edu/pbc/notes/crypto/voting.html
- http://hackingdistributed.com/2018/07/02/on-chain-vote-buying/

Appendix D: Static Invariant Prover

Trail of Bits has begun development of a tool which tests a given set of algebraic relationships to ensure they remain true before and after the invocation of a given function. This allows an operator to statically determine if these invariants hold.

Currently the tool is in the early stages of development and cannot handle functions with control flow, but Trail of Bits is in the process of expanding the scope of this tool to handle functions with arbitrary control flow using a SAT solver. Additionally, the tool is based on SlithIR, which does not allow for the analysis of inline assembly or external calls. Despite this limitation, a work-around is possible through hard-coding functionality within the tool during setup. Future updates to this tool aims to address these limitations, and further expand usability.