CONSENSYS
Diligence

Atomic Loans Audit

Date	September 2019
Auditors	Steve Marx

1 Summary

ConsenSys Diligence conducted a second security audit on the Atomic Loans smart contract system. The original report can be found here: https://github.com/ConsenSys/atomic-loans-audit-report-2019-07. Please refer to that original report for an overview and explanation of the system.

2 Changes Covered in This Audit

The major changes covered by this audit are:

1. The introduction of Compound into the `Funds` contract. Now lenders have the option of having their funds earn interest via Compound when they haven't been lent out

haven't been fleshed out.

2. The introduction of a global default parameters that are used by all loans by default. Notably, there's an automatically adjusting global interest rate. "Custom" loans can still choose to use other values for these parameters.
3. Liquidation sales are no longer auctions. Rather, the buyer purchases collateral at a 7% discount based on the fair market value according to the oracles.
4. The collateral swap during a sale has a slightly different protocol to work around a Bitcoin script limitation. In the new protocol, the collateral is moved to a P2SH *before* "back" signatures are created. This necessitates adding another secret, provided by the buyer, to complete the collateral sale.

3 Audit Scope

The audit team evaluated that the system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three broad categories:

1. **Security:** Identifying security related issues within the contract.
2. **Architecture:** Evaluating the system architecture through the lens of established smart contract best practices.
3. **Code quality:** A full review of the contract source code. The primary areas of focus include:
    - Correctness
    - Readability
    - Scalability
    - Code complexity
    - Quality of test coverage

4 Issues

Each issue has an assigned severity:

- **Minor** issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- **Medium** issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.

These should be addressed unless there is a clear reason not to.

- **Major** issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- **Critical** issues are directly exploitable security vulnerabilities that need to be fixed.

4.1 Intentional secret reuse can block borrower and lender from accepting liquidation payment Major ✓ Fixed

Resolution
This is fixed in AtomicLoans/atomicloans-eth-contracts#65.

Description

For Dave (the liquidator) to claim the collateral he's purchasing, he must reveal secret D. Once that secret is revealed, Alice and Bob (the borrower and lender) can claim the payment.

Secrets must be provided via the `Sales.provideSecret()` function:

code/ethereum/contracts/Sales.sol:L193-L200

```
        function provideSecret(bytes32 sale, bytes32 secret_) externa
                require(sales[sale].set);
                if      (sha256(abi.encodePacked(secret_)) == secretHa
        else if (sha256(abi.encodePacked(secret_)) == secretHashes[sa
        else if (sha256(abi.encodePacked(secret_)) == secretHashes[sa
        else if (sha256(abi.encodePacked(secret_)) == secretHashes[sa
        else
        }
```

Note that if Dave chooses the same secret hash as either Alice, Bob, or Charlie (arbiter), there is no way to set `secretHashes[sale].secretD` because one of the earlier conditionals will execute.

For Alice and Bob to later receive payment, they must be able to provide Dave's

secret:

code/ethereum/contracts/Sales.sol:L218-L222

```
        function accept(bytes32 sale) external {
        require(!accepted(sale));
        require(!off(sale));
                require(hasSecrets(sale));
                require(sha256(abi.encodePacked(secretHashes[sale].se
```

Dave can exploit this to obtain the collateral for free:

1. Dave looks at Alice's secret hashes to see which will be used in the sale.
2. Dave begins the liquidation process, using the same secret hash.
3. Alice and Bob reveal their secrets A and B through the process of moving the collateral.
4. Dave now knows the preimage for the secret hash he provided. It was revealed by Alice already.
5. Dave uses that secret to obtain the collateral.
6. Alice and Bob now want to receive payment, but they're unable to provide Dave's secret to the `Sales` smart contract due to the order of conditionals in `provideSecret()`.
7. After an expiration, Dave can claim a refund.

Mitigating factors

Alice and Bob *could* notice that Dave chose a duplicate secret hash and refuse to proceed with the sale. This is not something they are likely to do.

Recommendation

Either change the way `provideSecret()` works to allow for duplicate secret hashes or reject duplicate hashes in `create()`.

4.2 There is no way to convert between custom and non-custom funds Medium Won't Fix

Resolution

Users who want to switch between custom and non-custom funds can create a new address to do so. This is not actually a big burden because lenders need to use agent software to manage their funds anyway. That workflow typically involves generating a new address because the private key needs to be given to the agent software.

Description

Each fund is created using either `Funds.create()` or `Funds.createCustom()`. Both enforce a limitation that there can only be one fund per account:

code/ethereum/contracts/Funds.sol:L348-L355

```
function create(
    uint256  maxLoanDur_,
    uint256  maxFundDur_,
    address  arbiter_,
    bool     compoundEnabled_,
    uint256  amount_
) external returns (bytes32 fund) {
    require(fundOwner[msg.sender].lender != msg.sender || msg.sender =
```

code/ethereum/contracts/Funds.sol:L383-L397

```
function createCustom(
    uint256  minLoanAmt_,
    uint256  maxLoanAmt_,
    uint256  minLoanDur_,
    uint256  maxLoanDur_,
    uint256  maxFundDur_,
    uint256  liquidationRatio_,
    uint256  interest_,

    uint256  penalty_,
    uint256  fee_,
    address  arbiter_,
```

```
    bool     compoundEnabled_,
    uint256  amount_
) external returns (bytes32 fund) {
    require(fundOwner[msg.sender].lender != msg.sender || msg.sender =
```

These functions are the only place where `bools[fund].custom` is set, and there's no way to delete a fund once it exists. This means there's no way for a given account to switch between a custom and non-custom fund.

This could be a problem if, for example, the default parameters change in a way that a user finds unappealing. They may want to switch to using a custom fund but find themselves unable to do so without moving to a new Ethereum account.

Recommendation

Either allow funds to be deleted or allow funds to be switched between custom and non-custom.

4.3 `Funds.maxFundDur` has no effect if `maxLoanDur` is set

Medium **✓ Fixed**

Resolution
This is fixed in AtomicLoans/atomicloans-eth-contracts#68.

Description

`Funds.maxFundDur` specifies the maximum amount of time a fund should be active. It's checked in `request()` to ensure the duration of the loan won't exceed that time, but the check is skipped if `maxLoanDur` is set:

code/ethereum/contracts/Funds.sol:L510-L514

```
if (maxLoanDur(fund) > 0) {
    require(loanDur_       <= maxLoanDur(fund));
} else {
    require(now + loanDur_ <= maxFundDur(fund));
}
```

Examples

If a user sets `maxLoanDur` (the maximum loan duration) to 1 week and sets the `maxFundDur` (timestamp when all loans should be complete) to December 1st, then there can actually be a loan that ends on December 7th.

Recommendation

Check against `maxFundDur` even when `maxLoanDur` is set.

4.4 In `Funds`, `maxFundDur` is misnamed Minor ✓ Fixed

Resolution
This is fixed in AtomicLoans/atomicloans-eth-contracts#66.

Description

This is a timestamp, not a duration.

Recommendation

Rename to something with “timestamp” or perhaps “expiration” in the name.

4.5 `Funds.update()` lets users update fields that may not have any effect Minor ✓ Fixed

Resolution
This is fixed in AtomicLoans/atomicloans-eth-contracts#67.

Description

`Funds.update()` allows users to update the following fields which are only used if `bools[fund].custom` is set:

- `minLoanamt`
- `maxLoanAmt`
- `minLoanDur`
- `interest`
- `penalty`
- `fee`
- `liquidationRatio`

If `bools[fund].custom` is not set, then these changes have no effect. This may be misleading to users.

Examples

code/ethereum/contracts/Funds.sol:L454-L478

```
function update(
    bytes32  fund,
    uint256  minLoanAmt_,
    uint256  maxLoanAmt_,
    uint256  minLoanDur_,
    uint256  maxLoanDur_,
    uint256  maxFundDur_,
    uint256  interest_,
    uint256  penalty_,
    uint256  fee_,
    uint256  liquidationRatio_,
    address  arbiter_
) external {
    require(msg.sender == lender(fund));
    funds[fund].minLoanAmt       = minLoanAmt_;
    funds[fund].maxLoanAmt       = maxLoanAmt_;
    funds[fund].minLoanDur       = minLoanDur_;
    funds[fund].maxLoanDur       = maxLoanDur_;
    funds[fund].maxFundDur       = maxFundDur_;
    funds[fund].interest         = interest_;
    funds[fund].penalty          = penalty_;

    funds[fund].fee              = fee_;
    funds[fund].liquidationRatio = liquidationRatio_;
    funds[fund].arbiter          = arbiter_;
```

```
}
```

Recommendation

This could be addressed by creating two update functions: one for custom funds and one for non-custom funds. Only the update for custom funds would allow setting these values.

Appendix 1 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself

remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent

that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., "third parties") – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.