Scallop

Smart Contract

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

06/30/2023

Scallop Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A lending market on Sui.
Type	Lending
Auditors	MoveBit
Timeline	June 8, 2023 – June 30, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/scallop-io/sui-lending-protocol
Commits	66be3e84e388d9f067f50da47db9d078c7bd68aa 06b6d68d6994b91dc3747af3c269fe8cd018ee8d 2a61f1aef06bc3c48dabb6be4a53d630a7fad0d5 6d08c82aa6f3ccff29c0adbf25cd738ec4dad6c0

1.2 Files in Scope

The following are the SHA1 hashes of the initial reviewed files.

ID	Files	SHA-1 Hash
WHT	libs/whitelist/sources/whitelist.move	0303cd9e9558dd054344daf78595fe4275b0a625

U256	libs/math/sources/u256.move	92d942e8332a8d35b0739f9366b24133108d295b
U128	libs/math/sources/u128.move	883b054814c7bd6f4454dd7d8d8b53c33bf074b9
FP32	libs/math/sources/fixed_point32.move	c6ad7fa16b1477c0c8f7ff5f87946b2a19954193
U64	libs/math/sources/u64.move	efb59129e14d8ba7e050afeba4d1ca663763b36a
CDR	libs/coin_decimals_registry/sources/coin_decimals_registry.move	60e6c24715dbecd1f2b4eb50145453de82f59fc0
OTLV	libs/x/sources/one_time_lock_value.move	0e87437b0db6ce67aa2e493c5845f7c669f00e7d
SUPB	libs/x/sources/supply_bag.move	d28025267eebfbf01f5effd2571e4a8fcff9b65b
WT	libs/x/sources/wit_table.move	c30555d99b98d0d5a47560d3e0660a172750399d
ACT	libs/x/sources/ac_table.move	18c1c199ef8b7e6343fbe392da53e0a7f61fc8cc
OWP	libs/x/sources/ownership.move	3ec2b2c639ba571c71422ed149de051d1cd19aca
BCB	libs/x/sources/balance_bag.move	0f269c662f13aad908e7df061d2b1b5ad723664e
OBGC	protocol/sources/obligation/obligation_collaterals.move	edc88bdd05ea48af9c4c2220f822a8fb59956844
OBG	protocol/sources/obligation/obligation.move	012728bda1d672d0006c85f869555dfccf9a7bbf
OBGD	protocol/sources/obligation/obligation_debts.move	5496850bf52fac0d49d2b10d5a3dac60905c47db
PRC	protocol/sources/evaluator/price.move	535021fb005fb7a6a3fd94acd8731bd92d4ba30d

LQDE	protocol/sources/evaluator/liquidation_evaluator.move	b2456ef7d54028dfe194a651fc94aa14095ae498
CTV	protocol/sources/evaluator/collateral_value.move	85ff554b4a28e6f73b23e0606ef942570f71bd59
BWE	protocol/sources/evaluator/borrow_withdraw_evaluator.move	98a23b1085cacbb428e8e9d63200e4156b7ce307
DTV	protocol/sources/evaluator/debt_value.move	fb0fe747f1a59d5803be546657b3d2866ea7096d
VAC	protocol/sources/evaluator/value_calculator.move	5c0aa59cec899b79fcb0e62db5d9fb406cee7380
APP	protocol/sources/app/app.move	ff7bea02cc1b8bcb69d8f56517df98d58e6e8d2f
BRW	protocol/sources/user/borrow.move	e696ed92757e0b306d965b5a86afd395d85038a7
DPC	protocol/sources/user/deposit_collateral.move	ca5178a53da31dafe4416b72cdce80acefef6249
RDM	protocol/sources/user/redeem.move	9c78ac7f64a03c5a0d6501ac6a49f31d78ecbf48
MIT	protocol/sources/user/mint.move	04df75fe8258c3d578ccb8198579060fd905194d
OPO	protocol/sources/user/open_obligation.move	e32ccce7e9415555a94688e209bf2e5f5b646ff0
WTC	protocol/sources/user/withdraw_collateral.move	6d94adec843a6d38ad669bb1b25877625f4715f1
RPY	protocol/sources/user/repay.move	ebe1b5b503def89c45470ed4bc70006298101065
LQD	protocol/sources/user/liquidate.move	4c388b3865b6fafbfc0b9c1277205046078675f2
FLL	protocol/sources/user/flash_loan.move	6776feedb3a702a6f6fe6ba822bad6d65255cafc

CRV	protocol/sources/version/current_version.move	a2696415941b56107fae127af9d7170b82cb76ad
VSO	protocol/sources/version/version.move	0652904e73b19ef6bd51586bdd125be54144e70e
LMT	protocol/sources/market/limiter.move	5fd33d88038b988d664ba09e7f664cb91ea91ee9
RSV	protocol/sources/market/reserve.move	137b0cab9cdcf81b97fbde42290ff649a3d0914b
RSM	protocol/sources/market/risk_model.move	112f68234fa81be654ac270f39d03962ccc5f820
CLS	protocol/sources/market/collateral_stats.move	e7f8f00bc9c37f3ec9497697bbf07c9c817dd2b2
BWD	protocol/sources/market/borrow_dynamics.move	62efa2d395e54a12056817093d63bfbb27302f0f
MKT	protocol/sources/market/market.move	44f1598cfaa04b316caad47354ee23929059dfd2
ISM	protocol/sources/market/interest_model.move	d1040c3b9db550539ae8b74507b19a1c4aceeede
ERR	protocol/sources/error/error.move	13e0d48636b21bfeb3e744bc1a3306226c583290
SR	sui_x_oracle/supra_rule/sources/supra_registry.move	818d93999641153e27cffefc685871ecf4bef690
SRL	sui_x_oracle/supra_rule/sources/rule.move	ea9eaf9ef11c7c8f6df192e1513d0eafc4b8a6dc
PYA	sui_x_oracle/pyth_rule/sources/pyth_adaptor.move	b0d9aeb76d8f93383e1866c6114cadadea7867d5
PYR	sui_x_oracle/pyth_rule/sources/pyth_registry.move	56c11f4765edbd7d3c0a99a1e252088994b10e1c
PRL	sui_x_oracle/pyth_rule/sources/rule.move	53ba2671a57130e414a75003651e69bb60372889

PRF	sui_x_oracle/x_oracle/sources/price_feed.move	45d1525ee62026419acae6c5d48942c481e9cd39
XOC	sui_x_oracle/x_oracle/sources/x_oracle.move	ac994edf3cb756805ba2e9a60310059bc5246d2a
PUP	sui_x_oracle/x_oracle/sources/price_update_policy.move	1d58d2a24afaf5231103ab17f600990feea1acee
SSR	sui_x_oracle/switchboard_rule/sources/switchboard_registry.move	42a812417f8e889e333d0ce86802a179f51e84b5
SBA	sui_x_oracle/switchboard_rule/sources/switchboard_adaptor.move	3d0046fdbeb7ac28399086ff387a52c4f8a26283
SRU	sui_x_oracle/switchboard_rule/sources/rule.move	e21f5145a2e07f803e94040429ca1d095b20b370

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	15	13	2
Informational			
Minor	9	8	1
Medium	4	4	
Major	2	1	1
Critical			

1.4 MoveBit Audit BreakDown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" and "**Formal Verification**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section **1.2**.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by **Scallop** to identify any potential issues and vulnerabilities in the source code of the **Sui Lending** smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified **15** issues of varying severity, listed below.

ID	Title	Severity	Status
RSV–01	Possible Zero Token Minted in `mint_market_coin` Function	Minor	Fixed
RSV–02	Unable to Withdraw Flash Loan Fees	Major	Fixed
MKT–03	Lack of Reverse Functionality	Minor	Fixed
PRC–04	Lack of Validation for Price Value in `get_price` Function	Medium	Fixed
FLL–05	Lack of Whitelist Control in Flash Loans	Medium	Fixed
FLL–06	Unused Constant	Medium	Fixed
WHT–07	Unnecessary `store` Ability for Event Struct	Minor	Fixed
RPY–08	Incomplete Handling of Fully Repaid Loans in the Loan List	Medium	Fixed

SRL-09	Missing Adapter Implementation in Supra Contract	Minor	Fixed
MIT-10	Missing `entry` in `mint_entry` and `redeem_entry` Functions	Minor	Fixed
RSM-11	Lack of Range Checks for the `create_risk_model_change`	Minor	Fixed
RSM-12	Lack of Events Emit for the `add_risk_model` Function and `add_interest_model`	Minor	Fixed
OBG-13	Incorrect Return Value	Minor	Fixed
GLOBAL-14	Centralization Risk	Major	Acknowledged
GLOBAL-15	Third-Party Dependency	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the **Sui Lending** Smart Contract :

Admin

- Admin can update the interest model change delay through `extend_interest_model_change_delay()` .
- Admin can update the risk model change delay through `extend_risk_model_change_delay()` .
- Admin can update the limiter change delay through `limiter_change_delay()` .
- Admin can add an address into the whitelist for the `Market` through `add_whitelist_address()` .
- Admin can remove a whitelisted address from the `Market` through `remove_whitelist_address()` .
- Admin can create a new interest model change through `create_interest_model_change()` .
- Admin can add an interest model change and register coin for the `Market` through `add_i`

`nterest_model()`.

- Admin can update the interest model change for the `Market` through `update_interest_model()`.
- Admin can create a new risk model change through `create_risk_model_change()`.
- Admin can add a risk model and register collateral for the `Market` through `add_risk_model()`.
- Admin can update the risk model for the `Market` through `update_risk_model()`.
- Admin can add a `limiter` for the Market through `add_limiter()`.
- Admin can update the params of the `limiter` through `create_limiter_params_change()`.
- Admin can update the outflow segment params of `limiter` through `apply_limiter_params_change()`.
- Admin can update the outflow limit params of the `limiter` through `apply_limiter_limit_change()`.
- Admin can set the incentive reward factor through `set_incentive_reward_factor()`.
- Admin can set the flash loan fee through `set_flash_loan_fee()`.
- Admin can set the base asset active state through `set_base_asset_active_state()`.
- Admin can set the collateral active state through `set_collateral_active_state()`.
- Admin can withdraw revenue through `take_revenue()`.
- Admin can add a lock key to the ObligationAccessStore through `add_lock_key()`.
- Admin can remove the lock key from the ObligationAccessStore through `remove_lock_key()`.
- Admin can add the reward key to the ObligationAccessStore through `add_reward_key()`.
- Admin can remove the reward key from the ObligationAccessStore through `remove_reward_key()`.

Whitelist

- Whitelist can borrow from the `Market` through `borrow_entry()`.
- Whitelist can deposit collateral into `Obligation` through `deposit_collateral()`.
- Whitelist can borrow flash loans and repay flash loans from the `Market` through `borrow_flash_loan()` and `repay_flash_loan()`.
- Whitelist can use the `Coin<T>` to mint the `MarketCoin` through `mint_entry()`.
- Whitelist can use `MarketCoin` to redeem the `Coin<T>` through `redeem_entry()`.

- Whitelist can repay the debt of the `Market` through `repay()` .
- Whitelist can withdraw their collateral from the `Obligation` through `withdraw_collateral_entry()` .

4 Findings

RSV-01 Possible Zero Token Minted in `mint_market_coin` Function

Severity: Minor

Status: Fixed

Code Location: protocol/sources/market/reserve.move#L138

Descriptions: As the code below, in the calculation of `mint_amount` , it checks if the `balance_sheet.market_coin_supply` is greater than 0. If it is, it calculates `mint_amount` by dividing `underlying_amount` by the ratio of `balance_sheet.market_coin_supply` to the sum of `balance_sheet.cash` and `balance_sheet.debt` .The `mint_amount` calculation takes into account the ratio of `balance_sheet.market_coin_supply` to the sum of `balance_sheet.cash` and `balance_sheet.debt` , which includes accrued interest.

If `balance_sheet.cash + balance_sheet.debt` is greater than `balance_sheet.market_coin_supply` and `underlying_amount` is relatively small, resulting in a `mint_amount` of 0. This can lead to a situation where the user deposits funds (underlying_balance), but no MarketCoin shares are minted, resulting in the user not receiving any shares for their deposit.

Suggestion: Assert when `mint_amount` is 0:

```
assert!(mint_amount > 0, "Zero mint amount in mint_market_coin function");
```

Resolution: The client has followed our suggestion and fixed this issue.

RSV-02 Unable to Withdraw Flash Loan Fees

Severity: Major

Status: Fixed

Code Location: protocol/sources/market/reserve.move#L182

Descriptions: According to the code logic, the `borrow_flash_loan` method is used to borrow flash loans from the `Market` . It incurs a certain fee based on the amount of the flash loan. The `repay_flash_loan` method is used to repay the fee along with the flash loan back to the `Market` . However, in the contract, the `redeem` method converts `MarketCoin` for `Coin` from the `Market` based on the `debt` and `cash` calculations. There are no other methods in the contract to extract the fees. As a result, the fees may remain locked in the contract indefinitely.

Suggestion: It is recommended to add fees to `balance_sheet.cash` for user distribution or implement an alternative mechanism for withdrawing the fees to avoid locking.

Resolution: The client has introduced a withdrawal mechanism to handle the fees.

MKT-03 Lack of Reverse Functionality

Severity: Minor

Status: Fixed

Code Location: protocol/sources/market/market.move#L155-L164

Descriptions: The contract currently supports the functionality of registering coins and collateral assets for the protocol. However, it lacks the ability to remove or unregister coins and collateral assets. This means that if a registered coin or collateral asset experiences a vulnerability or depegging issue, the protocol does not have built-in mechanisms to prevent users from interacting with those specific assets.

The same situation also applies to the user whitelist, where users can only be added to the whitelist but there is no functionality to remove users from the whitelist.

Suggestion: Implementing the reverse functionality as mentioned above.

Resolution: The client has followed our suggestion and fixed this issue.

PRC-04 Lack of Validation for Price Value in `get_price` Function

Severity: Medium

Status: Fixed

Code Location: protocol/sources/evaluator/price.move#L13-L34

Descriptions: In the `get_price` function, there is a potential issue where the `price_value` is not validated as being zero. This can lead to incorrect calculations when the price is zero, impacting functions such as `borrow_withdraw_evaluator` , `collateral_value` , and `debt_value` , as well as `liquidation_evaluator` .

Suggestion: Assert when `price_value` is 0.

Resolution: The client has followed our suggestion and fixed this issue.

FLL-05 Lack of Whitelist Control in Flash Loans

Severity: Medium

Status: Fixed

Code Location: protocol/sources/user/flash_loan.move#L26

Descriptions: The `borrow_flash_loan` function does not have any whitelist control for the flash loan operation. This means that any borrower can initiate a flash loan without any restrictions or authorization checks. This lack of whitelist control poses a potential security risk as it allows unauthorized or malicious actors to exploit the flash loan functionality. It is important to implement proper whitelist controls to ensure that only authorized borrowers can access the flash loan feature and mitigate potential security vulnerabilities.

Suggestion: Adding a whitelist check in the flash loan function.

Resolution: The client has followed our suggestion and fixed this issue.

FLL-06 Unused Constant

Severity: Minor

Status: **Fixed**

Code Location: protocol/sources/user/flash_loan.move#L11, L12

Descriptions: Certain variables declared in the contract are not referenced or utilized in any of the contract's functions or logic. These unused variables add unnecessary complexity to the codebase and can potentially confuse developers or auditors trying to understand the contract's functionality.

Suggestion: Unless there are specific plans for utilizing these variables in future updates or additions, it is advisable to remove them to improve code readability and maintainability.

Resolution: The client has followed our suggestion and fixed this issue.

WHT-07 Unnecessary `store` Ability for Event Struct

Severity: Minor

Status: **Fixed**

Code Location: libs/whitelist/sources/whitelist.move#L19-L36

Descriptions: The event structure in Sui needs to have the ability to `copy` and `drop`, and does not need the `store` ability.

Suggestion: Delete the attribute of the structure `store`.

Resolution: The client has followed our suggestion and fixed this issue.

RPY-08 Incomplete Handling of Fully Repaid Loans in the Loan List

Severity: Medium

Status: **Fixed**

Code Location: protocol/sources/user/repay.move#L25

Descriptions: In the `repay` method, when a user fully repays all loans, the borrowed asset still remains in the loan list with a loan amount of 0. Consequently, in subsequent user operations, the system unnecessarily performs interest calculations for these assets that have been fully repaid. This issue indicates a flaw in the loan list maintenance and interest calculation process, as it fails to remove fully repaid loans from the loan list, resulting in redundant interest calculations.

Suggestion: It is recommended to modify the `repay` method to update the loan list and remove loans with a loan amount of 0.

Resolution: The client has followed our suggestion and fixed this issue.

SRL-09 Missing Adapter Implementation in Supra Contract

Severity: Minor

Status: Fixed

Code Location: protocol/sources/sui_x_oracle/supra_rule/rule.move

Descriptions: The Supra contract lacks implementation for the adapter. Without the adapter, the contract cannot effectively communicate or interact with the external environment, limiting its functionality and interoperability.

Resolution: The client has followed our suggestion and fixed this issue.

MIT-10 Missing `entry` in `mint_entry` and `redeem_entry` Functions

Severity: Minor

Status: Fixed

Code Location: protocol/sources/user/mint.move#L24, protocol/sources/user/redeem.move#L25

Descriptions: The functions `mint_entry` and `redeem_entry` are missing the `entry` keyword in their declarations. In the Move language, the `entry` keyword is used to define a function that can be called from outside of the module.

Suggestion: It is recommended to add `entry` keywords for these functions.

Resolution: The client has followed our suggestion and fixed this issue.

RSM-11 Lack of Range Checks for the `create_risk_model_change`

Severity: Minor

Status: Fixed

Code Location: protocol/sources/market/risk_model.move#L42-L68

Descriptions: The function `create_risk_model_change` lacks reasonable range checks for `collateral_factor`, `liquidation_factor`, `liquidation_penalty`, and `liquidation_discount`. Even in a trusted role system, there still exists the possibility of inputting typos and creating the wrong `risk_model` for the markets.

For example, if the `collateral_factor` is set to bigger than 100%, then surely users will deposit as much collateral as possible and lend coins as quickly as possible. The user gain is the market loss. The same rule applies to other factors. Or if the scale is set to 0, then it will lead to division by zero and abort.

Suggestion: Add assertions to make sure those values are in the reasonable range.

Resolution: The client has followed our suggestion and fixed this issue.

RSM-12 Lack of Events Emit for the `add_risk_model` Function and `add_interest_model`

Severity: Minor

Status: Fixed

Code Location: protocol/sources/market/risk_model.move#L70-L88, protocol/sources/market/interest_model.move#L82-L99

Descriptions: The function `add_risk_model` lacks events emitted after the new risk model is added. In the best practice, there should be events to notify users that the risk models have been changed. Otherwise, they may deposit according to the old risk model and be surprised.

Also found in `add_interest_model`.

Suggestion: It is recommended to emit events for these actions.

Resolution: The client has followed our suggestion and fixed this issue.

OBG-13 Incorrect Return Value

Severity: Minor

Status: Fixed

Code Location: protocol/sources/obligation/obligation.move#L42-L48

Descriptions: In the `obligation_key_uid_mut` and `obligation_uid_mut` functions, the return value should be mutable.

Suggestion: Modify the return value `&UID` to `&mut UID`.

Resolution: The client has followed our suggestion and fixed this issue.

GLOBAL-14 Centralization Risk

Severity: Major

Status: Acknowledged

Descriptions: There are some centralization risks in the contract:

- Admin can update the interest model change delay through `extend_interest_model_change_delay()`.
- Admin can update the risk model change delay through `extend_risk_model_change_delay()`.
- Admin can update the limiter change delay through `limiter_change_delay()`.
- Admin can add an address into the whitelist for the `Market` through `add_whitelist_address()`.
- Admin can remove a whitelisted address from the `Market` through `remove_whitelist_address()`.
- Admin can create a new interest model change through `create_interest_model_change()`.

- Admin can add an interest model change and register coin for the `Market` through `add_interest_model()`.
- Admin can update the interest model change for the `Market` through `update_interest_model()`.
- Admin can create a new risk model change through `create_risk_model_change()`.
- Admin can add a risk model and register collateral for the `Market` through `add_risk_model()`.
- Admin can update the risk model for the `Market` through `update_risk_model()`.
- Admin can add a `limiter` for the Market through `add_limiter()`.
- Admin can update the params of the `limiter` through `create_limiter_params_change()`.
- Admin can update the outflow segment params of `limiter` through `apply_limiter_params_change()`.
- Admin can update the outflow limit params of the `limiter` through `apply_limiter_limit_change()`.
- Admin can set the incentive reward factor through `set_incentive_reward_factor()`.
- Admin can set the flash loan fee through `set_flash_loan_fee()`.
- Admin can set the base asset active state through `set_base_asset_active_state()`.
- Admin can set the collateral active state through `set_collateral_active_state()`.
- Admin can withdraw revenue through `take_revenue()`.
- Admin can add a lock key to the ObligationAccessStore through `add_lock_key()`.
- Admin can remove the lock key from the ObligationAccessStore through `remove_lock_key()`.
- Admin can add the reward key to the ObligationAccessStore through `add_reward_key()`.
- Admin can remove the reward key from the ObligationAccessStore through `remove_reward_key()`.

Suggestion: It is recommended to take some measures to mitigate centralization risk.

GLOBAL-15 Third-Party Dependency

Severity: Minor

Status: Acknowledged

Descriptions: During the audit process, we discovered that the system relies on third–party services for certain functionalities, such as an oracle. However, please note that this audit does not cover the third–party dependencies, including the oracle. We assume that the data provided by the oracle is accurate and properly handled by the system.

Suggestion: It is recommended to utilize audited and widely adopted third–party dependencies whenever possible. Necessary security measures should be implemented to address potential issues that may arise from these dependencies. Additionally, proactive monitoring of the third–party services is essential during the operational phase to promptly detect and mitigate any potential risks and avoid potential losses.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non–exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed**: The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer