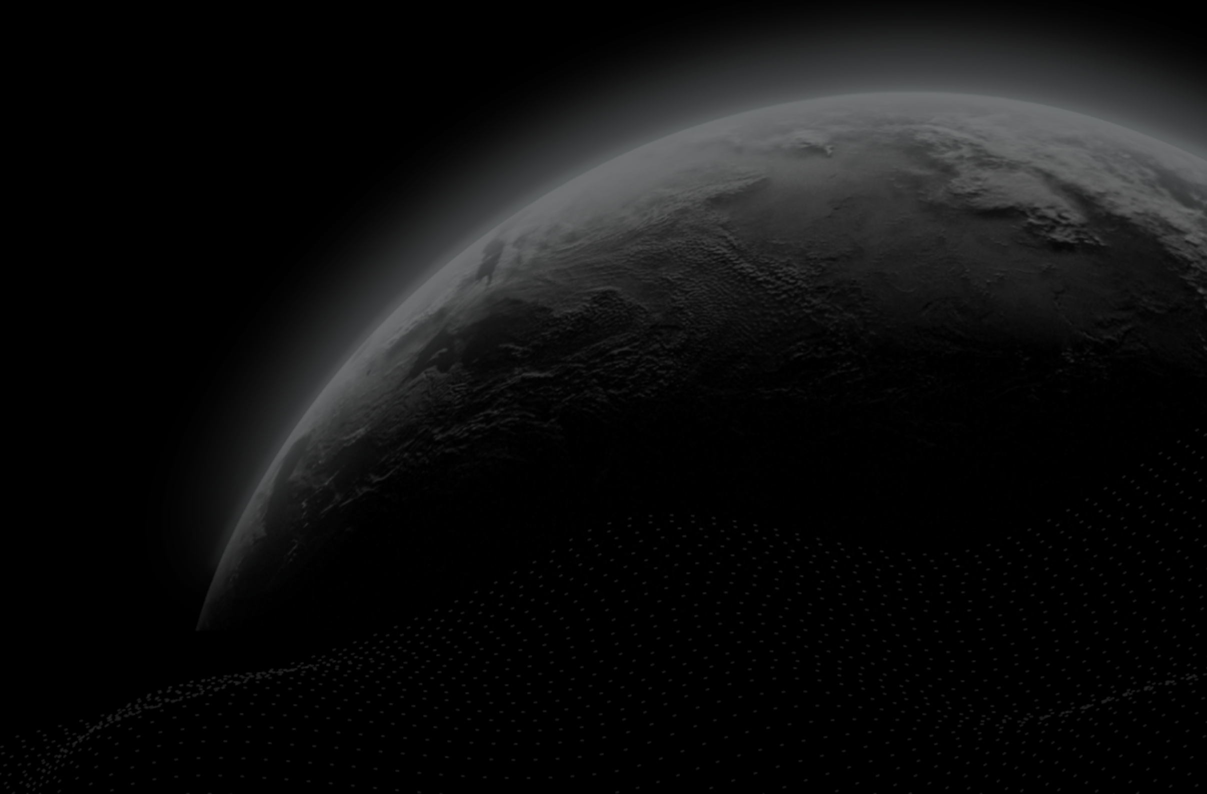
CERTIK

Security Assessment

Megaton Finance - Audit 1

CertiK Verified on Jan 18th, 2023



CertiK Verified on Jan 18th, 2023

Megaton Finance - Audit 1

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEX	TON	Manual Review
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
FunC	Delivered on 01/18/2023	router, lp-minter, lp-wallet, allocator

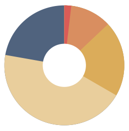
CODEBASE

update 10678c3cca15627161ebf6bc842cbc5e411271b9

base f7776e9ea9495fc0e9aa22a85426838ac2d988dc

...View All

Vulnerability Summary



54	44	0	0	10	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	1	Critical	1 Resolved	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	6	Major	6 Resolved	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	11	Medium	10 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	24	Minor	19 Resolved, 5 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	12	Informational	8 Resolved, 4 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | MEGATON FINANCE - AUDIT 1

Disclaimer

CODEBASE | MEGATON FINANCE - AUDIT 1

Repository

update 10678c3cca15627161ebf6bc842cbc5e411271b9

base f7776e9ea9495fc0e9aa22a85426838ac2d988dc

AUDIT SCOPE | MEGATON FINANCE - AUDIT 1

22 files audited ● 5 files with Acknowledged findings ● 3 files with Resolved findings ● 14 files without findings

ID	File	SHA256 Checksum
● ALL	contracts/amm/allocator.fc	fe88bef43fd42b8f07732cbf900dc09d38e548f e79e4a0e2dd5901323bfa95fb
● LPM	contracts/amm/lp-minter.fc	d9e76e271526122bb21dcd3df7e35d29072b4 6d28fa2ed1dc56c51bedce6cdd5
● ROU	contracts/amm/router.fc	7a15c19c06d10fc9a675f2f63f94314ddb0c425 4f0a74d0a0eab38991e8e98ea
● UTI	contracts/imports/utils.fc	a77e2bc76bcd4f6aaaaeddff1383bf9a7eb0f19 184221d4f368389fec98c8a1d
● JET	contracts/jetton-minter.fc	c1e96cdd08843805bc42a2fe5a7ab2b867cc6 b698b57c6d36ead943641e21ef0
● LPW	contracts/amm/lp-wallet.fc	189bedab2e0072e2f3694d7e731d7825323a5 6e9edf598be20259372be470a98
● OPC	contracts/imports/op-codes.fc	2db5f4e6f0087b8c0ebb63faf4398cd07f78d83 cf1685a8b4a50cbb788f0eaaa
● JEO	contracts/jetton-wallet.fc	d0b14a28428efc117f389936d221c4e2cf6fe3 547206ed494a8ecb931ee6a834
● CON	contracts/imports/constants.fc	91a348fc40806abbeeb407146177f4ab8e7cd 5927fec508496933ebfe8563dcc
● DIS	contracts/imports/discovery-params.fc	d7a3fd5cf6e39c1c1074855c6b525a9c441ea7 34a749c0d0eb5260922f112830
● MES	contracts/imports/message_utils.fc	75ddc7ebf2a0b2006ce5428ea12acf1febd971 2656d57637993d726ac7847871
● JEW	contracts/jetton-wallet.fc	04ea4246fd5ccd290b4189a8da3303da2a1f9f 426f8555a73126531b8d75e2b9
● JEI	contracts/jetton-minter.fc	81d1769a1123c337540ae8f3a1041a4cb5ed4 bba4cd9411eba5ecae35ba339f1
● COT	contracts/imports/constants.fc	fb5763f6806b0f599fc4e6bc2d9b387318e155 348b178525e1c7e75a36257648

ID	File	SHA256 Checksum
DIO	contracts/imports/discovery-params.fc	a7a66da2e83b2c9826ca64d33767e2db1e52063f6abedc3dd290eef6d0fbe919
MEG	contracts/imports/message_utils.fc	03c967c523e9ade43127135a69475fa98308f6a301df608c0b5836f4900dba9a
OPD	contracts/imports/op-codes.fc	0e00b6fdc37ae8b75e697c253cfbd16e83c903320fe265606cce74594c68f040
UTS	contracts/imports/utils.fc	a2c951f76fd58f912f603b2f0ee6252545162cb1b1a4950b56048f6fb086aeb9
ALC	contracts/amm/allocator.fc	5996875adb755efe2a0e3601a529ceaaaefc1741312c5a0ce6fa3f345cfddb26
LPN	contracts/amm/lp-minter.fc	4c712500beff42754afec1891fee077233937ef199d5cce3e4c05273bfe3a377
LPL	contracts/amm/lp-wallet.fc	368e2f62d9136c64cff9217cb96a103f00993fc8cdf9de2e1787c75bad35c992
ROE	contracts/amm/router.fc	e9ffd7a2d83a778811a010d6f730f9e3a21855cf555ebc11f715dfcf765aecb9

APPROACH & METHODS | MEGATON FINANCE - AUDIT 1

This report has been prepared for Megaton Finance to discover issues and vulnerabilities in the source code of the Megaton Finance - Audit 1 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review techniques.

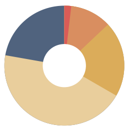
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Perform code refactoring: add functions composing common messages;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | MEGATON FINANCE - AUDIT 1



54	1	6	11	24	12
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Megaton Finance - Audit 1. Through this audit, we have uncovered 54 issues ranging from different severity levels. Utilizing the techniques of Manual Review to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
LPM-01	All Funds Can Be Stolen Via Forged `op::transfer_notification` To `lp-minter`	Control Flow	Critical	● Resolved
LPM-02	`lp-minter` Always Rejects `op::transfer`	Volatile Code	Major	● Resolved
LPM-03	Argument Order Is Incorrect In `save_data()`	Logical Issue	Major	● Resolved
LPM-04	`handle_provide_wallet_address()` Returns Incorrect Address	Logical Issue	Major	● Resolved
LPM-05	`min_amount` Storage Field Is Shadowed And Overwritten By Incoming Argument In `lp-minter::handle_transfer_notification()`	Volatile Code	Major	● Resolved
LPW-01	`lp-wallet` Doesn't Guarantee `pending_balance` Consistency	Logical Issue	Major	● Resolved
LPW-02	Sending `op::init_pending_balance` To `lp-wallet` Wipes The Deposits	Control Flow	Major	● Resolved
JET-01	`jetton-minter::op::mint` Allows To Send Invalid Messages	Logical Issue	Medium	● Resolved
LPM-06	`update_mining_index()` Can Ignore `next_mining_rate_cell`	Logical Issue	Medium	● Resolved
LPM-07	Wrong `response_address` Used For `op::burn` Message In `lp-minter::handle_transfer()`	Inconsistency	Medium	● Resolved

ID	Title	Category	Severity	Status
LPM-08	`msg_value` Is Not Controlled At `lp-minter` On `op::claim`	Inconsistency	Medium	● Resolved
LPM-09	`msg_value` Is Not Controlled At `lp-minter` On `op::check_mintable_notification`	Inconsistency	Medium	● Resolved
LPM-10	Pending Jettons Can Be Returned If `lp_minter` `is_stopped`	Inconsistency	Medium	● Resolved
LPW-03	`lp-wallet` / `lp-minter` Don't Follow TEP-74 Standard	Inconsistency	Medium	● Resolved
ROT-01	`router::handle_change_lp_content()` Is Never Executed	Inconsistency	Medium	● Resolved
ROU-01	Wrong Destination Address Used In Case Of Rejected Swap Request	Logical Issue	Medium	● Resolved
ROU-02	`router` Doesn't Validate The `sender_address` On `op::transfer_notification`	Control Flow	Medium	● Resolved
ROU-03	The Swap Payload From EOA Is Not Properly Validated In `router::handle_transfer_notification()`	Volatile Code	Medium	● Acknowledged
ALL-01	Bounced `op::transfer` Message From `governance_jetton_wallet_address` Is Ignored In `allocator::handle_claim()`	Volatile Code	Minor	● Acknowledged
AMM-01	`end_parse()` Is Missing	Volatile Code	Minor	● Resolved
CON-01	Pull-Over-Push Pattern Is Not Used In Admin Changing	Volatile Code	Minor	● Resolved
CON-02	Token Data Is Not Following TEP-64 Standard	Volatile Code	Minor	● Acknowledged
JEO-01	`msg_value` Is Not Controlled At `jetton-minter` On `op::mint`	Inconsistency	Minor	● Resolved
LPM-11	`parse_std_addr()` Can Be Used To Parse Address	Volatile Code	Minor	● Resolved

ID	Title	Category	Severity	Status
LPM-12	`msg_value` Is Not Controlled At `router` On `op::create_pool`	Inconsistency	Minor	● Resolved
LPM-13	`mined` And `current_index` Calculation Can Be Simplified	Coding Style	Minor	● Resolved
LPM-14	`lp-minter::handle_burn()` Doesn't Call `force_chain()`	Volatile Code	Minor	● Resolved
LPM-15	`msg_value` Is Not Controlled At `lp-minter` On `op::burn`	Inconsistency	Minor	● Resolved
LPM-16	`lp-minter` Sends `op::transfer` To `jettonA_wallet_address` In Non-Bounceable Mode	Volatile Code	Minor	● Resolved
LPM-17	Gas Management In `lp-minter::handle_transfer()` Is Inconsistent	Inconsistency	Minor	● Resolved
LPM-18	`to_jetton_address` Is Not Checked In `lp-minter::handle_transfer_notification()`	Volatile Code	Minor	● Acknowledged
LPM-19	`lp-minter` Silently Accepts Incoming LP Transfers	Volatile Code	Minor	● Resolved
LPM-20	`op::claim` Event Emitted In `lp-minter::handle_change_lp_mining_rate()`	Inconsistency	Minor	● Resolved
LPM-21	`min_amount` Is Not Respected By `lp-minter::handle_mintable_notification()`	Inconsistency	Minor	● Resolved
LPM-22	`lp-minter` Accepts Incoming Transfers Of Unrecognized Jettons	Volatile Code	Minor	● Resolved
LPW-04	Wrong `fwd_count` Calculation	Inconsistency	Minor	● Resolved
LPW-05	`jetton_address` Is Not Validated In `lp-wallet::check_mintable()`	Volatile Code	Minor	● Resolved
LPW-06	`lp-wallet::on_bounce()` Is Redundant	Inconsistency	Minor	● Resolved

ID	Title	Category	Severity	Status
ROU-04	`router` Allows `op::pool_created` From `pool_creator_address`	Control Flow	Minor	● Acknowledged
ROU-05	`router::handle_change_lp_mining_rate()` Gas Consumption Is Inconsistent	Volatile Code	Minor	● Resolved
ROU-06	`jettonA_address` / `jettonB_address` Can Be Arbitrary, Irrelevant To Real Jettons	Volatile Code	Minor	● Acknowledged
UTI-01	`mined()` Can Be Simplified	Coding Style	Minor	● Resolved
CON-03	Misleading Comments	Inconsistency	Informational	● Resolved
IMP-01	Unused Code	Inconsistency	Informational	● Resolved
LPM-23	`update_mining_index()` Can Be Refactored	Coding Style	Informational	● Acknowledged
LPM-24	Usage Of Magic Numbers	Coding Style	Informational	● Acknowledged
LPM-25	`in_msg_body` Is Unused In `lp-minter::handle_claim()`	Inconsistency	Informational	● Resolved
LPM-26	`op::change_router` Can't Be Handled Properly By `lp-minter`	Volatile Code	Informational	● Acknowledged
OPC-01	Response Messages `op` Don't Have High-Order Bit Set	Coding Style	Informational	● Resolved
ROU-07	Argument Names Of `router::get_lp_address()` Are Misleading	Coding Style	Informational	● Resolved
ROU-08	`either_forward_payload` Variable Is Unused	Coding Style	Informational	● Resolved
UTI-02	`calculate_jetton_wallet_address()` Can Be Replaced With `calculate_contract_address()`	Inconsistency	Informational	● Resolved
UTI-03	Long And Complicated Message Building Statements Can Be Formatted	Coding Style	Informational	● Acknowledged

ID	Title	Category	Severity	Status
UTI-04	`calculate_jetton_minter_address()` Is Unused And Dangerous	Volatile Code	Informational	● Resolved

LPM-01 | ALL FUNDS CAN BE STOLEN VIA FORGED `op::transfer_notification` TO `lp-minter`

Category	Severity	Location	Status
Control Flow	● Critical	contracts/amm/lp-minter.fc (update6): 842~847	● Resolved

Description

`lp-minter::handle_transfer_notification()` is supposed to handle `op::transfer_notification` messages from `lp-minter` wallets. Those messages carry the data required to perform adding liquidity or swapping operations. However, such messages can be sent by an externally owned account with forged arguments. This allows the extraction of all the funds from the `lp-minter` wallets.

Also, the check

```
840      throw_unless(75, msg_value > const::jetton_transfer_gas_consumption +
fwd_fee);
```

is performed, however, `lp_forward_router_gas_consumption` (0.1 TON) is forwarded to "router". `op::transfer` will not be processed due to not enough gas.

Scenario

1. The attacker directly sends `op::transfer_notification` to `lp-minter`
2. `in_msg_body` is constructed from
    - jetton_amount = jettonA_wallet_address balance
    - from_address = router_address
    - swap_slice = (from_jetton_address = jettonA_address, any to_jetton_address, user_address is the attacker address)
3. msg_value should be bigger than `(const::lp_forward_router_gas_consumption + fwd_fee)` (~0.11 TON) but less than `(const::lp_forward_router_gas_consumption + const::gas_consumption + fwd_fee)` (~0.12 TON)
4. `lp-minter` will send "back" jettons to `router` with the attacker address as the final destination

Recommendation

We recommend sending messages only back to `sender_address` instead of real wallet address to avoid spoofing. We recommend fixing the gas requirements.

LPM-02 | `lp-minter` ALWAYS REJECTS `op::transfer`

Category	Severity	Location	Status
Volatile Code	● Major	contracts/amm/lp-minter.fc (base): 1150~1153	● Resolved

Description

```
if (op == op::transfer) {
  handle_transfer(query_id, in_msg_body, sender_address);
}
```

There is no `return ()` in this case, so `throw(0xffff)` will be executed discarding all the uncommitted changes.

Recommendation

We recommend adding `return ();` .

LPM-03 | ARGUMENT ORDER IS INCORRECT IN `save_data()`

Category	Severity	Location	Status
Logical Issue	● Major	contracts/amm/lp-minter.fc (base): 379	● Resolved

Description

```
379    save_data(total_supply + lp_amount, min_amount, swap_fee, ...
```

`save_data()` accepts `swap_fee` as the second argument, `min_amount` as the third.

Recommendation

We recommend fixing the argument order.

LPM-04 | `handle_provide_wallet_address()` RETURNS INCORRECT ADDRESS

Category	Severity	Location	Status
Logical Issue	● Major	contracts/amm/lp-minter.fc (base): 583~584	● Resolved

Description

```
583       msg = msg.store_slice(calculate_user_jetton_wallet_address(owner_address,
my_address(), lp_wallet_code));
```

`handle_provide_wallet_address()` is supposed to provide `lp-wallet` address. However, an incorrect `jetton-wallet` address is returned.

Recommendation

We recommend fixing the code this way:

```
583       msg = msg.store_slice(calculate_user_lp_wallet_address(sender_address,
my_address(), lp_wallet_code, jettonA_address, jettonB_address));
```

LPM-05 | `min_amount` STORAGE FIELD IS SHADOWED AND OVERWRITTEN BY INCOMING ARGUMENT IN `lp-minter::handle_transfer_notification()`

Category	Severity	Location	Status
Volatile Code	● Major	contracts/amm/lp-minter.fc (base): 733~734, 832~833	● Resolved

Description

`lp-minter` has `min_amount` storage field with a minimal allowed LP amount for each account. The function `handle_transfer_notification()` gets the `min_amount` argument from `in_msg_body` practically shadowing the storage field. Moreover, the shadowing value is saved to the storage instead. This allows the end user to set any `min_amount` for any `lp-minter` at will.

Recommendation

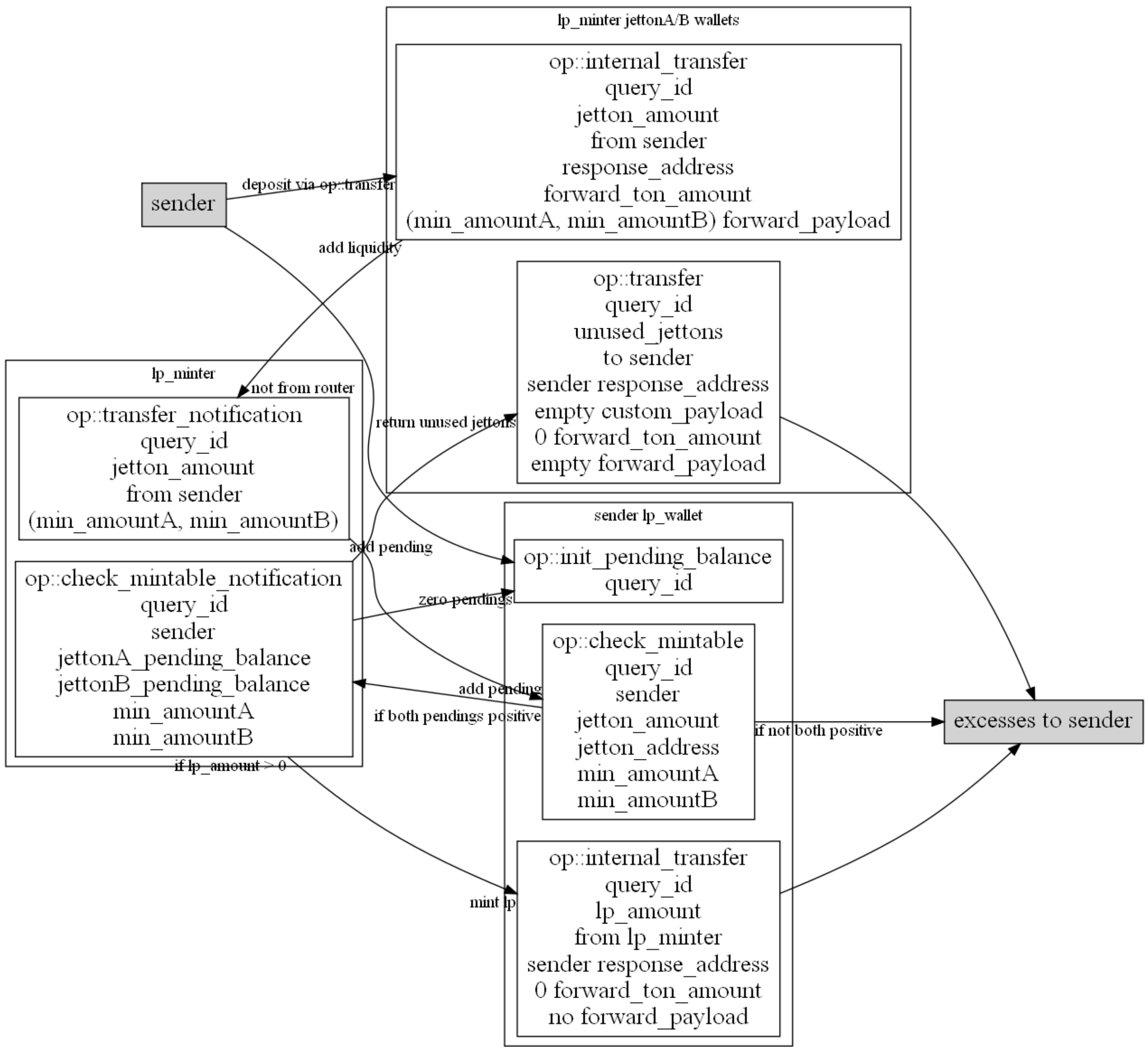
We recommend renaming the argument variable to avoid shadowing.

LPW-01 `lp-wallet` DOESN'T GUARANTEE `pending_balance` CONSISTENCY

Category	Severity	Location	Status
Logical Issue	● Major	contracts/amm/lp-wallet.fc (base): 62~63	● Resolved

Description

`lp-wallet` allows minting via `op::check_mintable` and canceling via `op::check_pending_jetton` at the same time. This leads to double-spending.



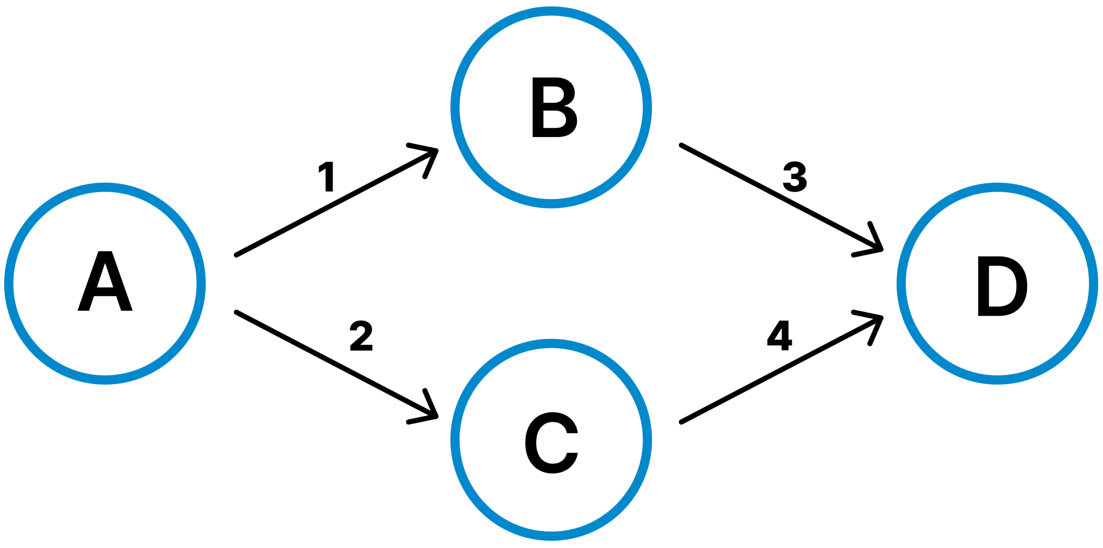
Minting LP is currently working this way:

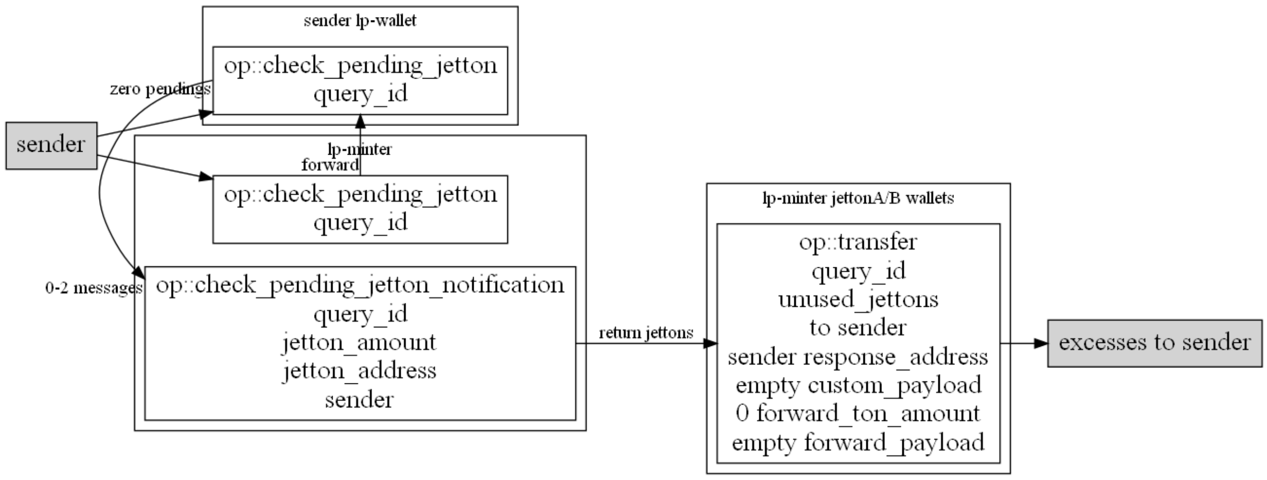
1. Sender deposits jettonA via sending `op::transfer` to `lp-minter walletA`.

2. `lp-minter walletA` sends `op::transfer_notification` to `lp-minter`.
3. `lp-minter` sends `op::check_mintable` to `sender lp-wallet`.
4. `sender lp-wallet` sends `op::check_mintable_notification` to `lp-minter` if both pending amounts are positive
5. `lp-minter` sends `op::init_pending_balance` to `sender lp-wallet`. Pending amounts are zeroed.
6. `lp-minter` sends `op::internal_transfer` to `sender lp-wallet`. Sender's LP balance is increased.

However, between steps 3 and 5, the `sender lp-wallet` can get and execute another `op::check_pending_jetton` and extract both pending jetton deposits.

According to Message delivery guarantees we can't be sure which message, 3 or 4 will be delivered first.





The attack scenario:

1. Sender deposits jettonB. `sender lp-wallet::jettonB_pending_balance` is updated.
2. Sender deposits jettonA. `sender lp-wallet op::check_mintable` is executed. Since both pending balances are positive, `op::check_mintable_notification` is sent to `lp-minter`.
3. Sender sends `op::check_pending_jetton` to `sender lp-wallet`.

4. `sender lp-wallet` sends 2 `op::check_pending_jetton_notification` to `lp-minter`. Pending balances are zeroed.
5. `lp-minter` returns Sender's deposits to their wallets.
6. `op::check_mintable_notification` is delivered to `lp-minter`. A new LP is minted on Sender's wallet, their zero pending balances are zeroed again.

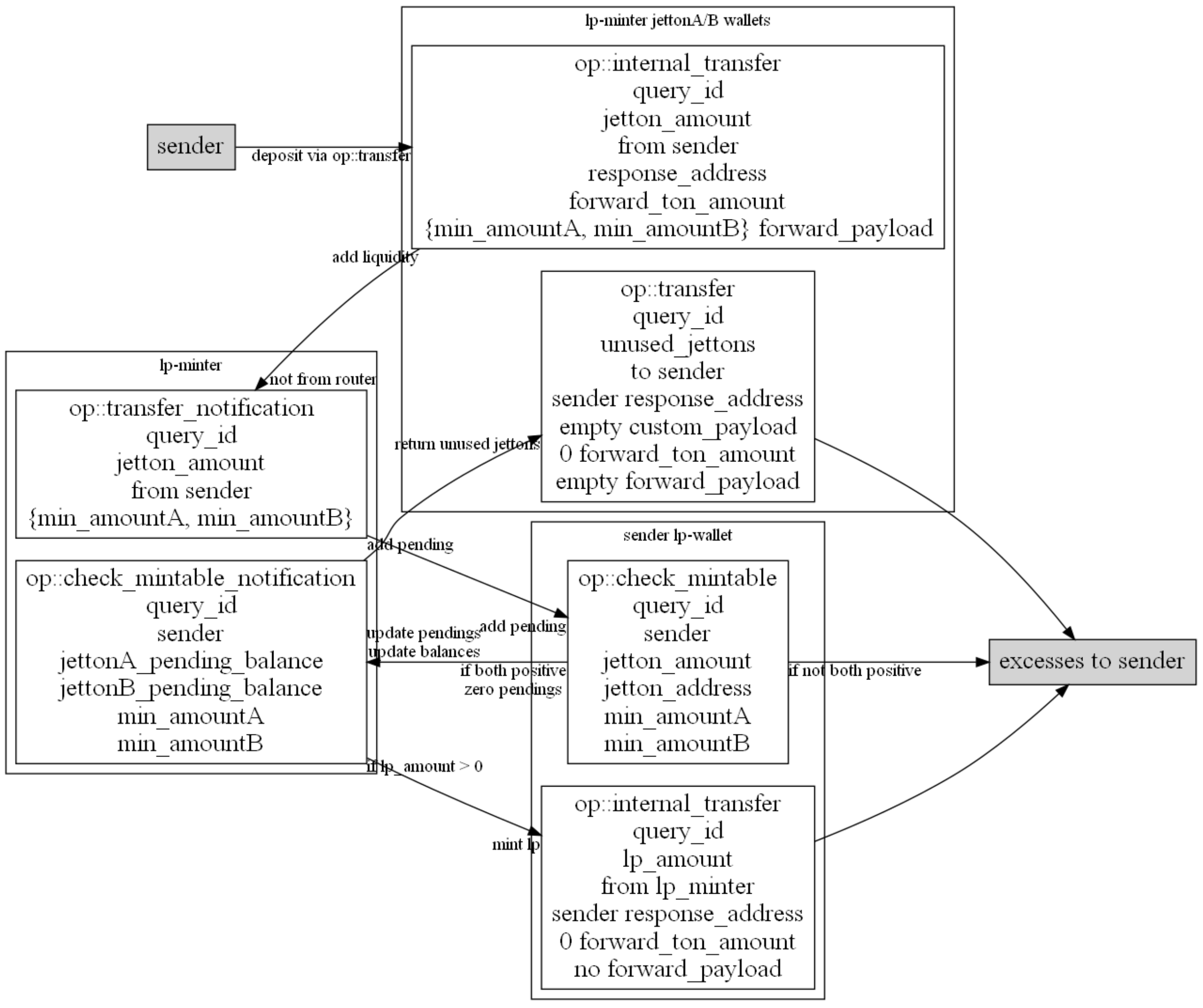
As a result, Sender extracted deposited jettons in step 5 and minted the corresponding LP in step 6.

Recommendation

We recommend dropping of `lp-wallet` support and managing pending balances in `lp-minter` directly. We recommend decreasing the balances before transaction action phase.

Alleviation

`sender lp-wallet` now zeroes pending balances before sending `op::check_mintable_notification` to `lp-minter` if both pending amounts were positive. New workflow:



LPW-02 | SENDING `op::init_pending_balance` TO `lp-wallet` WIPES THE DEPOSITS

Category	Severity	Location	Status
Control Flow	● Major	contracts/amm/lp-wallet.fc (base): 299~300	● Resolved

Description

`op::init_pending_balance` in `lp-wallet` zeroes the user's jetton pending balances. It can be sent directly by the user or as part of `op::check_mintable` flow.

Sending it directly wipes users' jetton deposits and makes `lp-minter` pending balances inconsistent.

`in_msg_body` argument is not used by `init_pending_balance()`.

Recommendation

We recommend allowing `op::init_pending_balance` to be processed only if received from `lp-minter`. We recommend dropping of unused arguments. We recommend merging the handler with `lp-wallet::receive_tokens()`.

JET-01 | `jetton-minter::op::mint` ALLOWS TO SEND INVALID MESSAGES

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/jetton-minter.fc (base): 76~77	● Resolved

Description

```
71    if (op == op::mint) {
72      throw_unless(73, equal_slices(sender_address, minter_address));
73      slice to_address = in_msg_body~load_msg_addr();
74      cell master_msg = in_msg_body~load_ref();
75      slice master_msg_cs = master_msg.begin_parse();
76      master_msg_cs~skip_bits(32 + 64); ;; op + query_id
77      int jetton_amount = master_msg_cs~load_coins();
78
79      mint_tokens(msg_value, to_address, jetton_wallet_code, master_msg);
```

`jetton-minter::op::mint` is supposed to allow `minter_address` to mint new jettons to `to_address` via sending of `op::internal_transfer` message. However, `master_msg` is not validated:

- any `op` can be used
- the `op::internal_transfer` message format is not validated
- the `forward_ton_amount` argument is not respected, `min_tons_for_storage` is not provided
- `msg_value` is not controlled, `CARRY_REMAINING_GAS` mode is not used
- the bounced message is not handled, `total_supply` is not decreased back in case of failure

Recommendation

We recommend checking all the required arguments of `op::internal_transfer` message, we recommend handling of bounced message.

LPM-06 | `update_mining_index()` CAN IGNORE `next_mining_rate_cell`

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/amm/lp-minter.fc (base): 151~153	● Resolved

Description

If `next_mining_rate_cell` was set, the function `update_mining_index()` is supposed to calculate `first_mined` for the first period with the old mining rate and `second_mined` for the second period with the updated mining rate.

However, if `first_mined <= last_mined` , the `second_mined` will not even be checked, despite the fact it can be bigger than `last_mined` . This can lead to loss of the reward.

Recommendation

We recommend checking if `second_mined` is bigger than `last_mined` even if `first_mined` is not.

LPM-07 | WRONG `response_address` USED FOR `op::burn` MESSAGE IN `lp-minter::handle_transfer()`

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/lp-minter.fc (base): 683~684	● Resolved

Description

`lp-minter::handle_transfer()` sends a `op::burn` message to ex-lp-owner-wallet. The `sender_address` is specified as a `response_address` argument, however, the `sender_address` is ex-lp-owner-wallet, not the ex-lp-owner. It is reasonable to send `op::excesses` to the originator of the transaction.

Recommendation

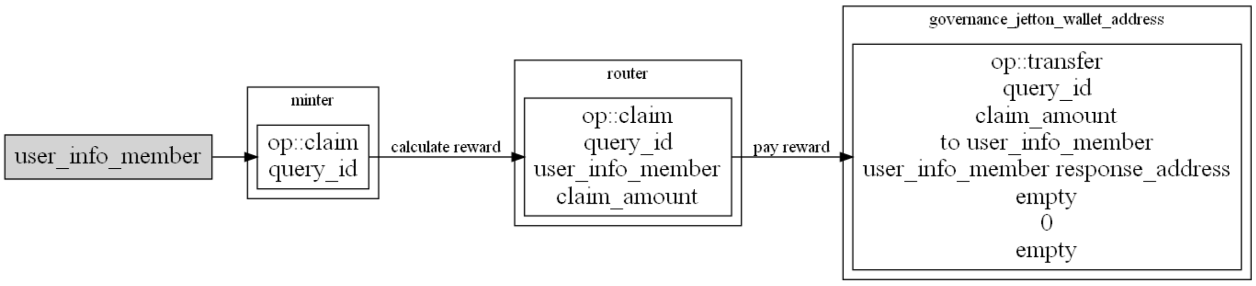
We recommend using of `from_address` as a `response_address` to return unused fees.

LPM-08 | `msg_value` IS NOT CONTROLLED AT `lp-minter` ON `op::claim`

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/lp-minter.fc (base): 254~255	● Resolved

Description

`lp-minter::handle_claim()` doesn't check that `msg_value` is enough.



Claiming reward works this way:

1. `user_info_member` sends `op::claim` to `lp-minter`. `msg_value` is not checked.
2. `lp-minter` sends `op::claim` to `router`, forwards 0.04 TON, and pays for processing and forwarding.
3. `router` sends `op::transfer` to `governance_jetton_wallet_address`, forwards 0.04 TON, and pays for processing and forwarding.
4. `governance_jetton_wallet_address` returns excesses to `user_info_member`.

As a result, if `router` has zero balance, the `op::transfer` will not be sent due to out-of-gas, and the reward will be lost. If `router` and `lp-minter` have enough gas, up to 0.04 TON can be stolen from `lp-minter` per each `op::claim`.

Recommendation

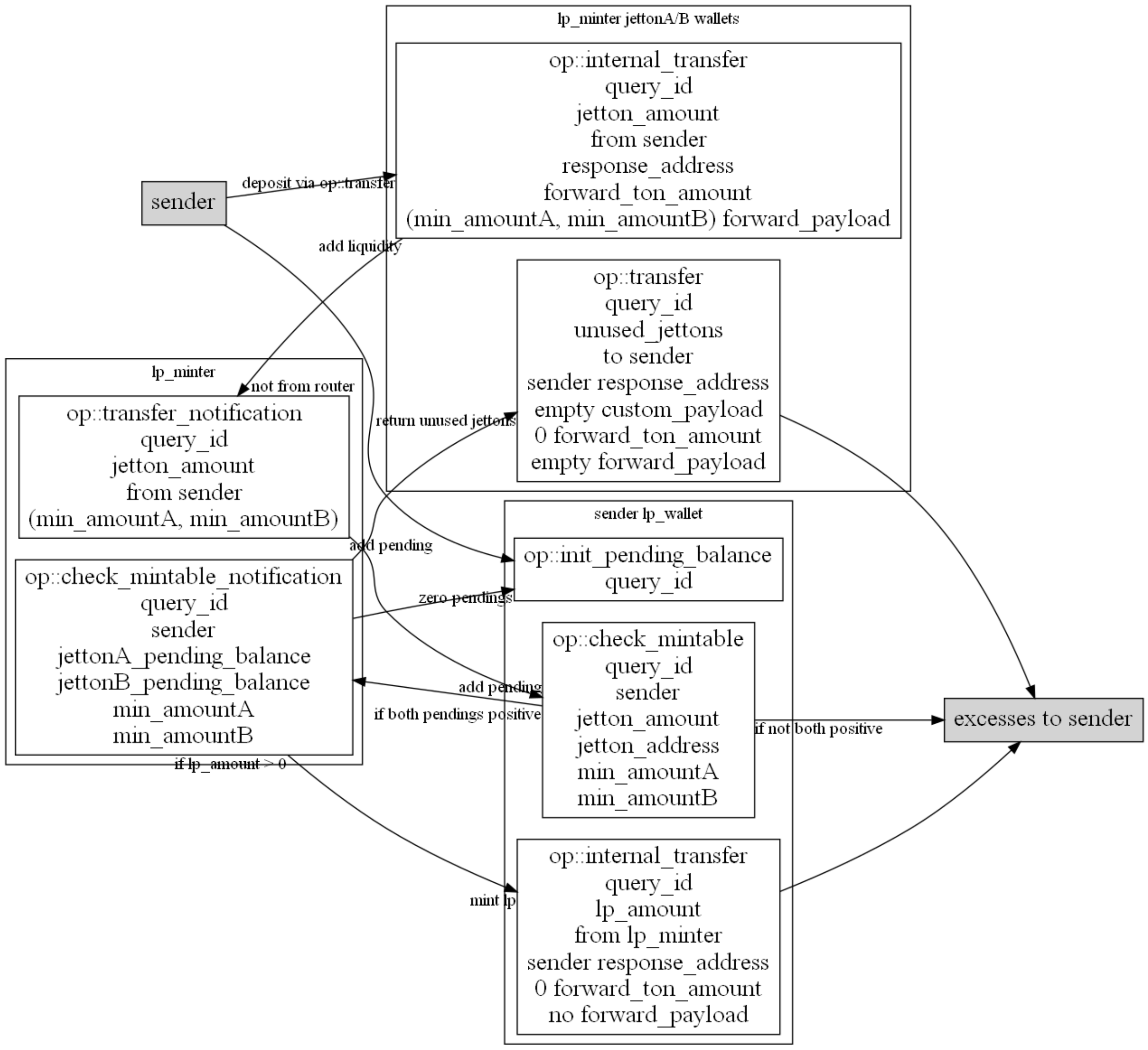
We recommend explicitly checking in `minter::handle_claim()` that `msg_value` is at least `2 * const::gas_consumption + 2 * fwd_fee + 0.04` and forwarding to `router` `const::gas_consumption + fwd_fee + 0.04`. We recommend to `CARRY_REMAINING_GAS` in `router::handle_claim()`.

LPM-09 | `msg_value` IS NOT CONTROLLED AT `lp-minter` ON `op::check_mintable_notification`

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/lp-minter.fc (base): 399~400	● Resolved

Description

`lp-minter::handle_mintable_notification()` doesn't check that `msg_value` is enough. It can lead to funds draining or incomplete execution.



Handling of `op::check_mintable_notification` at `lp-minter` works this way:

1. `sender` deposits jettons to `lp-minter` and uses some `forward_ton_amount`.

2. `lp-minter` gets `op::transfer_notification` . `msg_value` is not checked.
3. `lp-minter` sends `op::check_mintable` to `sender lp-wallet` , and forwards all remaining gas.
4. `sender lp-wallet` sends `op::check_mintable_notification` to `lp-minter` , and forwards all remaining gas.
5. `lp-minter` during the processing of `op::check_mintable_notification` sends 0.03 TON with `op::init_pending_balance` , 0.05 TON during the return of unused jettons (maybe twice), 0.2 TON to `lp-wallet` during minting.

As a result, if not enough `forward_ton_amount` , the deposit will not be handled properly leading to inconsistent pending balances. If `lp-minter` has enough balance, up to 0.25 TON can be stolen per each deposit.

Recommendation

We recommend:

1. explicitly checking in `lp-minter` `op::transfer_notification` handler, that `msg_value` is enough to finish the workflow
2. trying to return jettons if `msg_value` is not enough or the payload is invalid
3. avoiding failure in the action phase
4. returning excesses in `op::init_pending_balance` handler or removing this message completely.

LPM-10 | PENDING JETTONS CAN BE RETURNED IF `lp_minter` `is_stopped`

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/lp-minter.fc (update6): 1227~1228	● Resolved

Description

`lp-minter::handle_pending_jetton()` allows the user to return jettons deposited to add liquidity. The operation will fail if `is_stopped` is set. However, `handle_pending_jetton_notification()` can be triggered via a direct `op::check_pending_jetton` message to `lp-wallet` . This essentially allows skipping the check.

Also, `router` processes `op::claim` requests even if `is_stopped` . It is unclear if that behavior is intended.

Recommendation

We recommend disallowing processing of `op::check_pending_jetton` messages in `lp-wallet` , if they are received directly from the wallet owner, or ignoring of `is_stopped` flag in `lp-minter::handle_pending_jetton()` . We recommend clarifying the intended behavior of `router::handle_claim()` in the case of `is_stopped` via code comments.

Alleviation

[**CertiK**]: `router::handle_claim()` behavior is left intact in the case of `is_stopped` . It is possible to stop the router and keep `lp-minter` s unstopped. Claim requests will still be processed in this case.

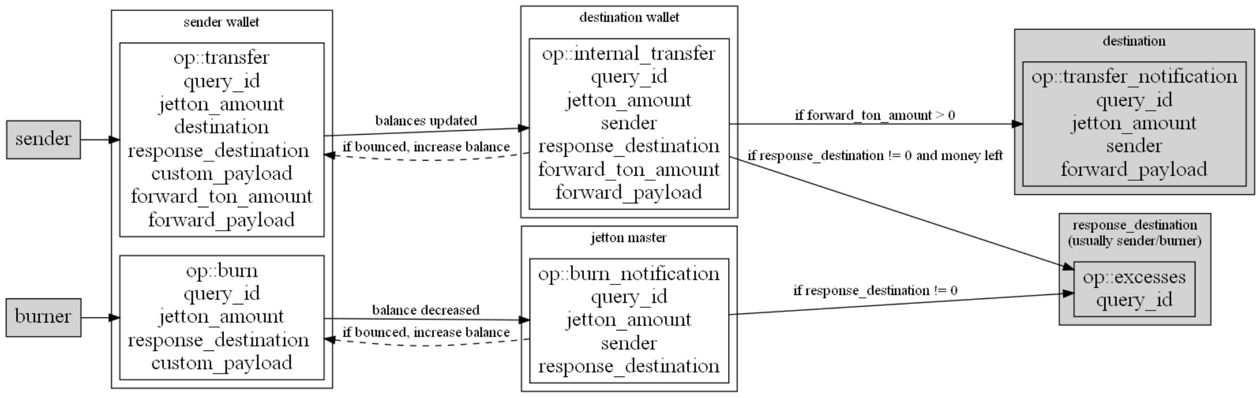
[**Megaton Finance**]: We add cheking `is_stopped` flag in `handle_mintable_notification()` . So now, every case to call the router's `op::claim` is blocked with `is_stopped` flag in `lp-minter` .

LPW-03 `lp-wallet` / `lp-minter` DON'T FOLLOW TEP-74 STANDARD

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/lp-wallet.fc (base): 92~93	● Resolved

Description

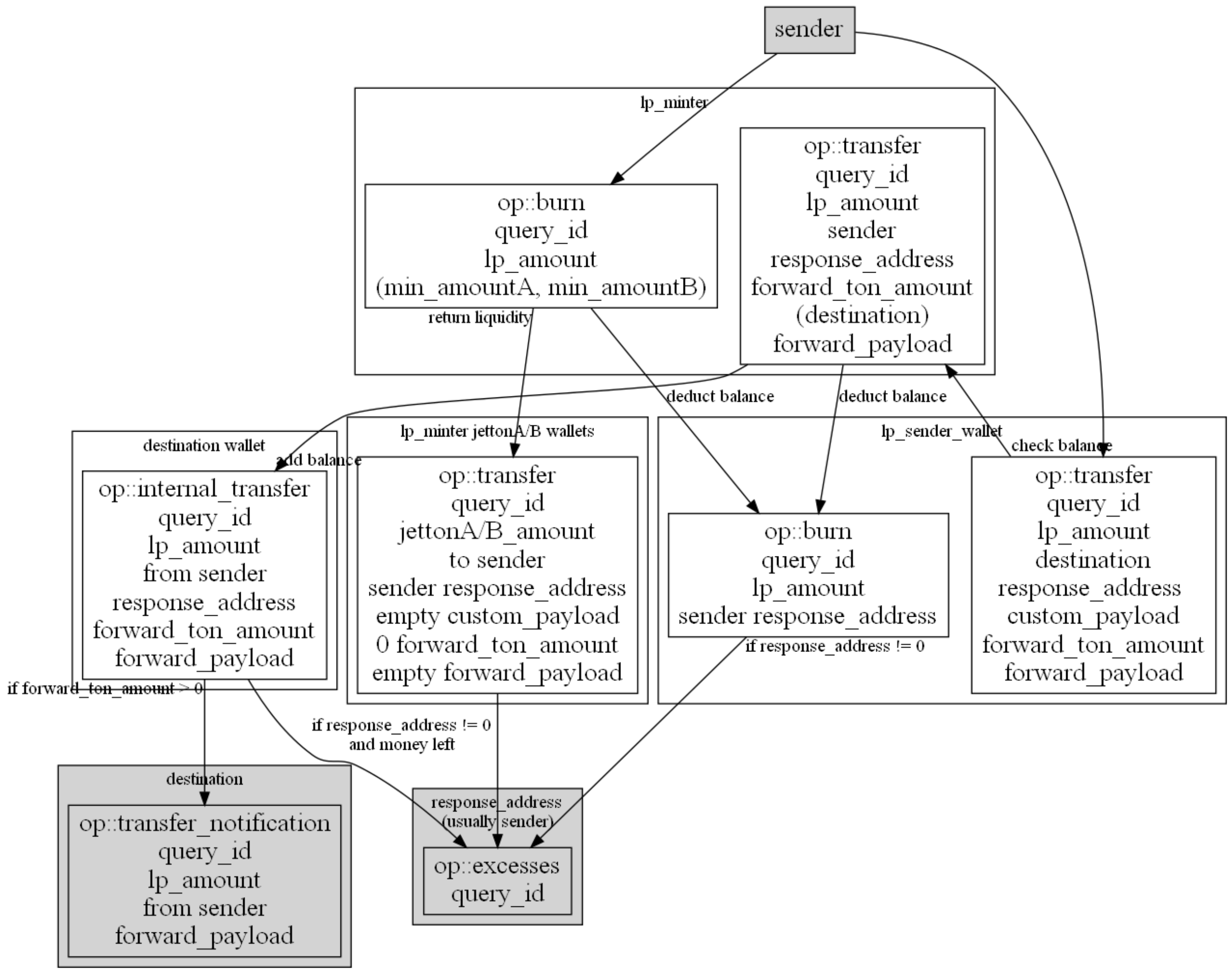
TEP-74 Standard wallet interaction diagram:



According to TEP-74:

- `op::transfer` uses the `destination` address argument after `amount`. But `lp-wallet::send_tokens()` uses `owner_address` ("from") instead. At the same time, `transfer#0f8a7ea5` tag is preserved.
- `op::burn` is rejected if received not from the owner. But `lp-wallet::burn_tokens()` accepts the message only from `lp_minter_address`.

`lp-wallet` interaction diagram:



`lp-wallet` doesn't allow direct transfers between wallets, all the state changes are controlled by `lp-minter`. The infinite sharding paradigm (when the transactions are processed independently on different accounts) can't be used in this case. The `balance` is mirrored between `lp-wallet` and `lp-minter`.

The possible bounced messages between `lp-wallet` and `lp-minter` are not handled by both contracts.

Recommendation

We recommend dropping the `lp-wallet` and using `lp-minter` only. We recommend changing the `op::transfer` tag or the arguments layout.

Alleviation

The `op::transfer` message layout was updated to follow the standard.

ROT-01 | `router::handle_change_lp_content()` IS NEVER EXECUTED

Category	Severity	Location	Status
Inconsistency	● Medium	contracts/amm/router.fc (update6): 508~509	● Resolved

Description

`router::handle_change_lp_content()` is supposed to change jetton content for the specific `lp_address`. However, the function is inaccessible, `recv_internal()` doesn't handle the corresponding message.

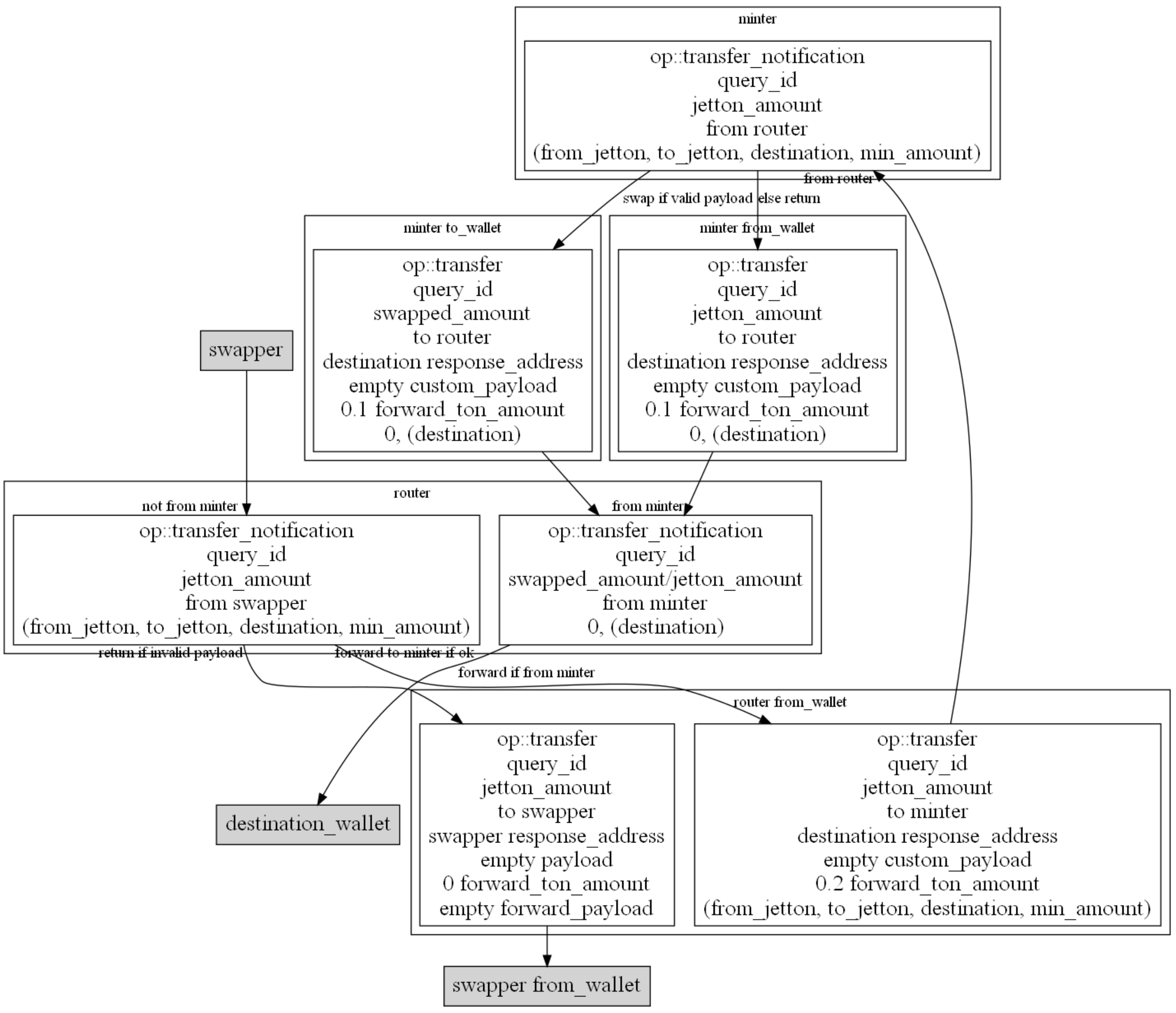
Recommendation

We recommend updating the `router::recv_internal()`.

ROU-01 | WRONG DESTINATION ADDRESS USED IN CASE OF REJECTED SWAP REQUEST

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/amm/router.fc (base): 301~302	● Resolved

Description



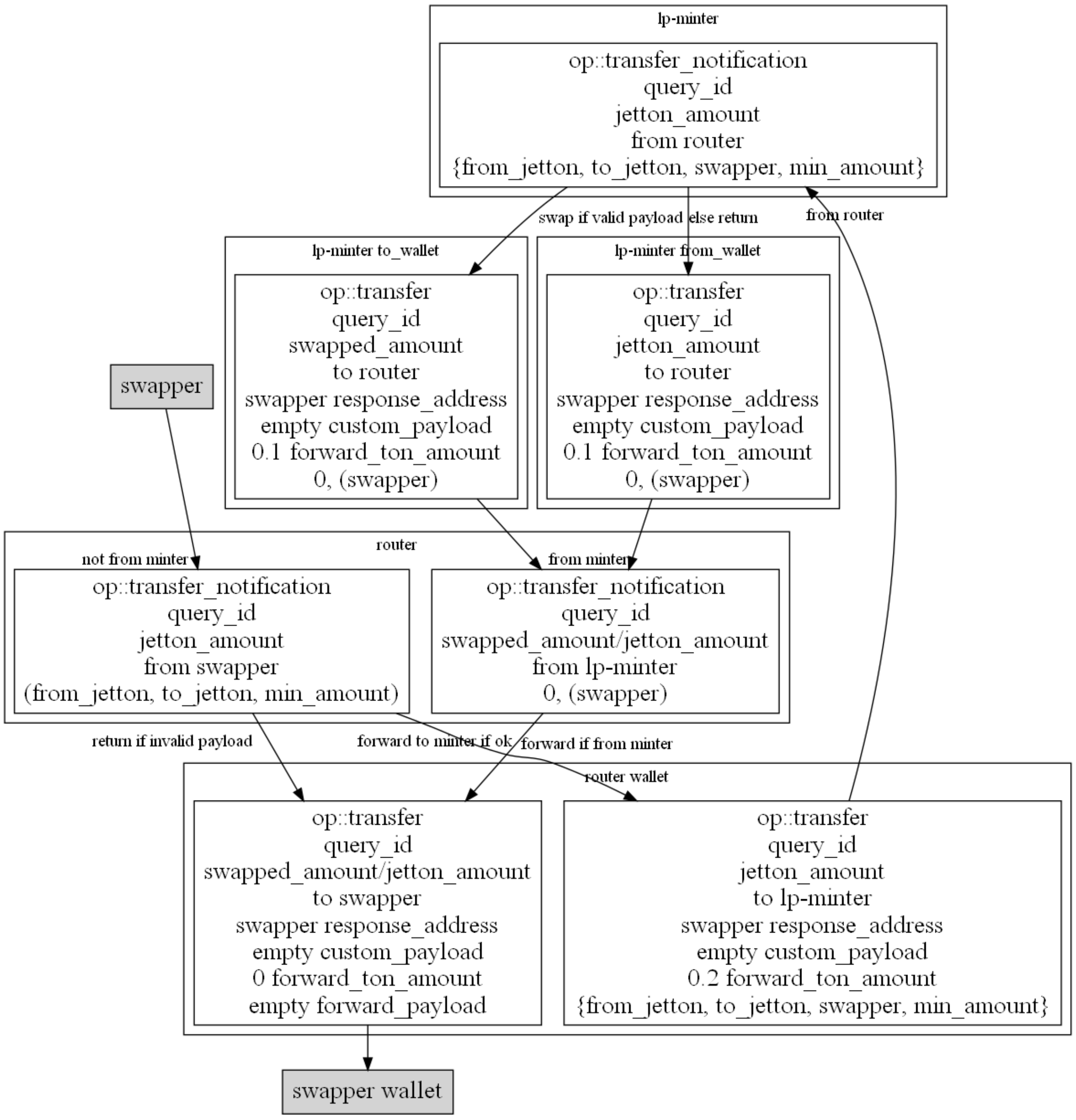
The `swapper` deposits `from_jettons` and provides the payload `(from_jetton, to_jetton, destination, min_amount)`. In case the payload is invalid (too short), the jettons are returned to `swapper from_wallet` address. However, in case `minter` finds `min_amount` criteria is not satisfied, the jettons are "returned" to `destination_wallet` address. It is unexpected by the `swapper`.

Recommendation

We recommend always returning `from_wallet` jettons to `swapper from_wallet`.

Alleviation

New workflow:



ROU-02 `router` DOESN'T VALIDATE THE `sender_address` ON `op::transfer_notification`

Category	Severity	Location	Status
Control Flow	● Medium	contracts/amm/router.fc (base): 287~289, 320~321, 343~345, 349~350, 378~379	● Resolved

Description

`router::handle_transfer_notification()` gets the `from_address` from the payload and treats it as trustworthy.

The attacker can send to the `router` the message `{ op::transfer_notification, query_id: any, jetton_amount: any, from_address: real lp-minter address, 0, (destination: self address) }`. The `router` checks the `lp-minter` address is known and sends the `op::transfer` message back with 0.05 TON in non-bounceable mode. This can drain the `router` balance.

The same problem is reproduced if `from_address` is not `lp-minter` or the payload is incorrect. The `router` sends 0.3 TONs back to attacker if the payload is valid.

Recommendation

We recommend sending `op::transfer` in `CARRY_REMAINING_GAS` mode with 0 TONs attached and bounceable flag set.

ROU-03 | THE SWAP PAYLOAD FROM EOA IS NOT PROPERLY VALIDATED IN `router::handle_transfer_notification()`

Category	Severity	Location	Status
Volatile Code	● Medium	contracts/amm/router.fc (base): 313~314, 337~341	● Acknowledged

Description

When EOA sends the swap request to `router` via jettons depositing, the expected payload format is `{ from_jetton_address, to_jetton_address, destination, min_amount }`. However, if the payload can't be parsed, the execution terminates, and the jettons and TONs are not returned.

The function checks if `slice_bits(swap_slice) <= 267 * 3`, but that doesn't guarantee the success of parsing. `min_amount` doesn't fit `267 * 3` bits payload.

Recommendation

We recommend using of `TRY` primitive and returning jettons/TONs in case of failure.

ALL-01 | BOUNCED `op::transfer` MESSAGE FROM `governance_jetton_wallet_address` IS IGNORED IN `allocator::handle_claim()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/allocator.fc (base): 84~86	● Acknowledged

Description

`allocator::handle_claim()` sends an internal `op::transfer` message to `governance_jetton_wallet_address` in bounceable mode. In case this message can't be processed, for example, if transferring is currently paused, it will be bounced back and ignored by `allocator` . `last_mined` state field will not be decreased back.

Recommendation

We recommend catching the bounced messages and reverting the corresponding changes.

Alleviation

Sending the message in non-bounceable mode doesn't address the finding.

AMM-01 | `end_parse()` IS MISSING

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/allocator.fc (base): 17~18; contracts/amm/lp-wallet.fc (base): 29~30, 33~34	● Resolved

Description

`end_parse(slice s)` ensures that no more data is available in `s`. This allows checking of message format correctness.

Recommendation

We recommend using `end_parse()` wherever possible to ensure the correct message format.

CON-01 | PULL-OVER-PUSH PATTERN IS NOT USED IN ADMIN CHANGING

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/allocator.fc (base): 105~106, 114~115; contracts/amm/lp-minter.fc (base): 1084~1085; contracts/amm/router.fc (base): 409~410; contracts/jetton-minter.fc (base): 137~138, 144~145	● Resolved

Description

The functions `handle_change_claim_admin()` / `handle_change_admin()` override the previously set `claim_admin_address` / `admin_address` with the new value without guaranteeing they are able to actuate transactions on-chain.

Recommendation

We recommend using of the pull-over-push pattern whereby a new `admin` is first proposed and consequently needs to accept the `admin` status ensuring that the account can actuate transactions on-chain.

CON-02 | TOKEN DATA IS NOT FOLLOWING TEP-64 STANDARD

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 1104~1105; contracts/amm/router.fc (base): 445~446; contracts/jetton-minter.fc (base): 151~152	● Acknowledged

Description

TEP-64 standard describes the Token Data Standard. However, `jetton-minter` , `lp-minter` contracts don't validate the data in `op::change_content` . `router` doesn't validate the data in `handle_change_lp_default_content()` .

Changing the Token Data (decimals, name, symbol) is not recommended.

Recommendation

We recommend verifying that new token data follows the standard.

JEO-01 | `msg_value` IS NOT CONTROLLED AT `jetton-minter` ON `op::mint`

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/jetton-minter.fc (update1): 55~56	● Resolved

Description

`jetton-minter::mint_tokens()` doesn't check, that `msg_value` is enough. As a result, `op::internal_transfer` can be successfully sent but not properly processed by `jetton-minter` due to out-of-gas exception. The bounced messaged will not be created in this case, leaving `jetton_minter::total_supply` in inconsistent state.

Recommendation

We recommend explicitly checking that enough gas provided by the caller.

LPM-11 | `parse_std_addr()` CAN BE USED TO PARSE ADDRESS

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 406~409	● Resolved

Description

```
slice tmp_addr = to_address;
tmp_addr~skip_bits(11);
int addr_hash = tmp_addr~load_uint(256);
```

The way the address is parsed heavily relies on internal address representations. This makes the code volatile. Not all locations are mentioned.

Recommendation

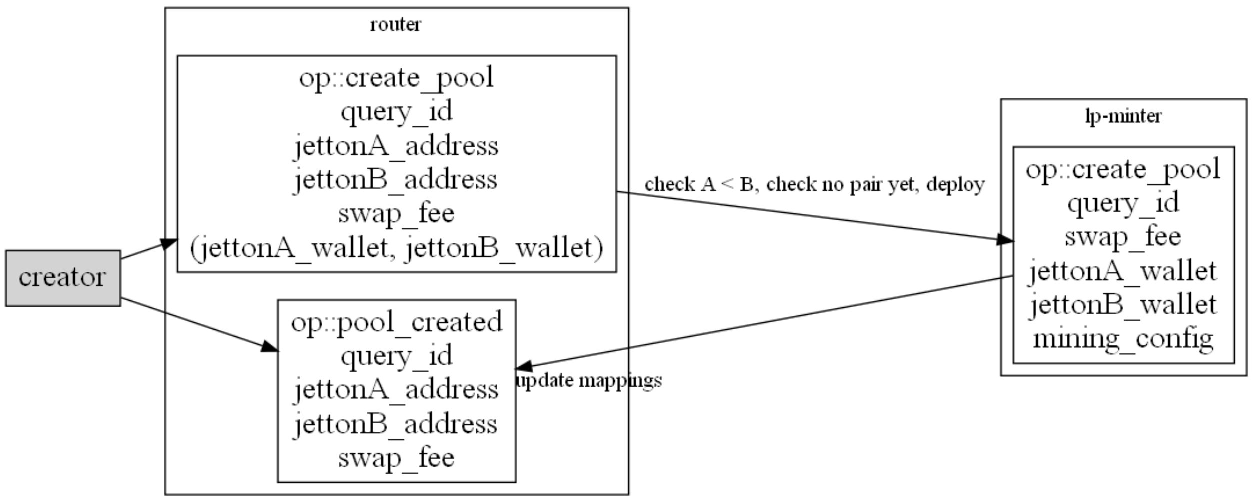
We recommend using `(int wc, int hash) = parse_std_addr(addr)` .

LPM-12 `msg_value` IS NOT CONTROLLED AT `router` ON `op::create_pool`

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-minter.fc (base): 111~112	● Resolved

Description

`router::handle_create_pool()` doesn't check that `msg_value` is enough.



Pool creation works this way:

1. `pool_creator_address` sends `op::create_pool` to `router`. `msg_value` is not checked.
2. `router` sends `op::create_pool` to `lp-minter`, forwards 0.1 TON, and pays for processing, forwarding, and deploying.
3. `lp-minter` sends `op::pool_created` to `router`, keeps 0.03 TON for storage, pays for processing and forwarding, and sends all the rest.
4. `router` pays for processing and keeps the change.

As a result, it is unclear to the caller, what is the expected `msg_value`.

Recommendation

We recommend explicitly checking in `handle_create_pool()` that the contract balance is bigger than `const::min_tons_for_storage + const::gas_consumption + fwd_fee + 0.1`, or checking the `msg_value` and `CARRY_REMAINING_GAS` in `router::create_pool()`.

LPM-13 `mined` AND `current_index` CALCULATION CAN BE SIMPLIFIED

Category	Severity	Location	Status
Coding Style	● Minor	contracts/amm/lp-minter.fc (base): 144~150, 183~189	● Resolved

Description

```
183         if ((current_mining_rate != 0) & (const::total_mining_rate != 0)) {
184           this_mined = current_mining_rate * (current_mined - last_mined) /
const::total_mining_rate;
185         }
186         if ((this_mined != 0) & (total_supply != 0)) {
187           current_index = current_index + (this_mined * 1000000000000000000) /
total_supply; ;; 10^18
188         }
```

The check `(current_mining_rate != 0)` is redundant, since in this case `this_mined` will still be zero.

The check `(const::total_mining_rate != 0)` is redundant, since the constant is not zero. If the constant can be zero, we recommend adding this check to lines 144, 154, or leaving the function immediately.

The check `(this_mined != 0)` is redundant, since `current_index` is not changed in this case.

`muldiv()` can be used to prevent potential overflows

Recommendation

We recommend removing of redundant checks to simplify the code.

LPM-14 | `lp-minter::handle_burn()` DOESN'T CALL `force_chain()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 452~453	● Resolved

Description

`lp-minter::handle_burn()` doesn't enforce the `sender_address` chain to be `basechain`. But `user_info_dict` is indexed by `addr_hash` only. Calling the function from another chain can lead to unexpected results.

`calculate_contract_address()` enforces the address to be in `workchain()`. But `calculate_*_state_init()` functions do not.

Recommendation

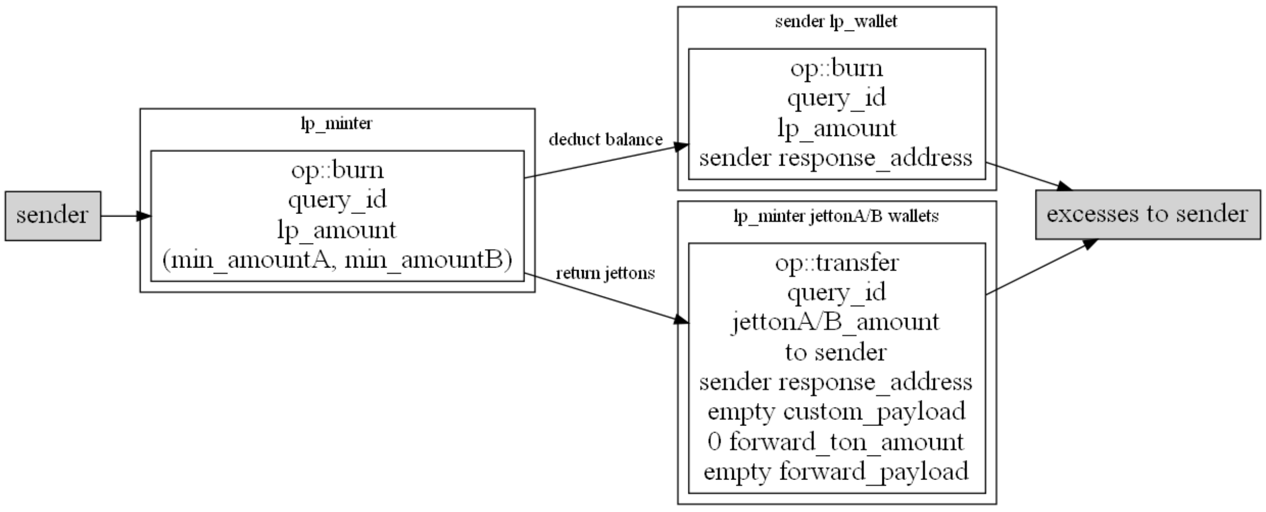
We recommend enforcing the chain in `recv_internal()`, `get_wallet_address()`, `handle_change_router()`, `handle_change_admin()`, and in other functions accepting addresses.

LPM-15 `msg_value` IS NOT CONTROLLED AT `lp-minter` ON `op::burn`

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-minter.fc (base): 559~560	● Resolved

Description

`lp-minter::handle_burn()` doesn't check that `msg_value` is enough.



Burning works this way:

1. `sender` sends `op::burn` to `lp-minter`. `msg_value` is not checked, `handle_burn()` argument is unused.
2. `lp-minter` sends `op::claim` to `router` with 0.04 TON.
3. `lp-minter` sends `op::burn` to `sender lp-wallet` with 0.03 TON.
4. `lp-minter` sends 2 `op::transfer` to jetton wallets with 0.04 TON.
5. All messages send excesses to the `sender`.

As a result, the `sender` can steal up to 0.15 TON from `lp-minter` per each `op::burn` message.

Recommendation

We recommend explicitly checking that enough gas provided by the caller. We recommend using `CARRY_REMAINING_GAS` mode in the last `send_raw_message()`.

LPM-16 | `lp-minter` SENDS `op::transfer` TO `jettonA_wallet_address` IN NON-BOUNCEABLE MODE

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 330~331, 341~342, 361~362, 498~499, 517~518, 993, 1017	● Resolved

Description

According to Guidelines, almost all internal messages sent between smart contracts should be bounceable. Then, if the destination smart contract throws an unhandled exception while processing this message, the message will be "bounced" back carrying the remainder of the original value (minus all message transfer and gas fees).

`lp-minter::handle_burn()`, `handle_pending_jetton_notification()`, `handle_mintable_notification()` send non-bounceable messages to own wallets. Forwarded TONs will not be returned in case of exception.

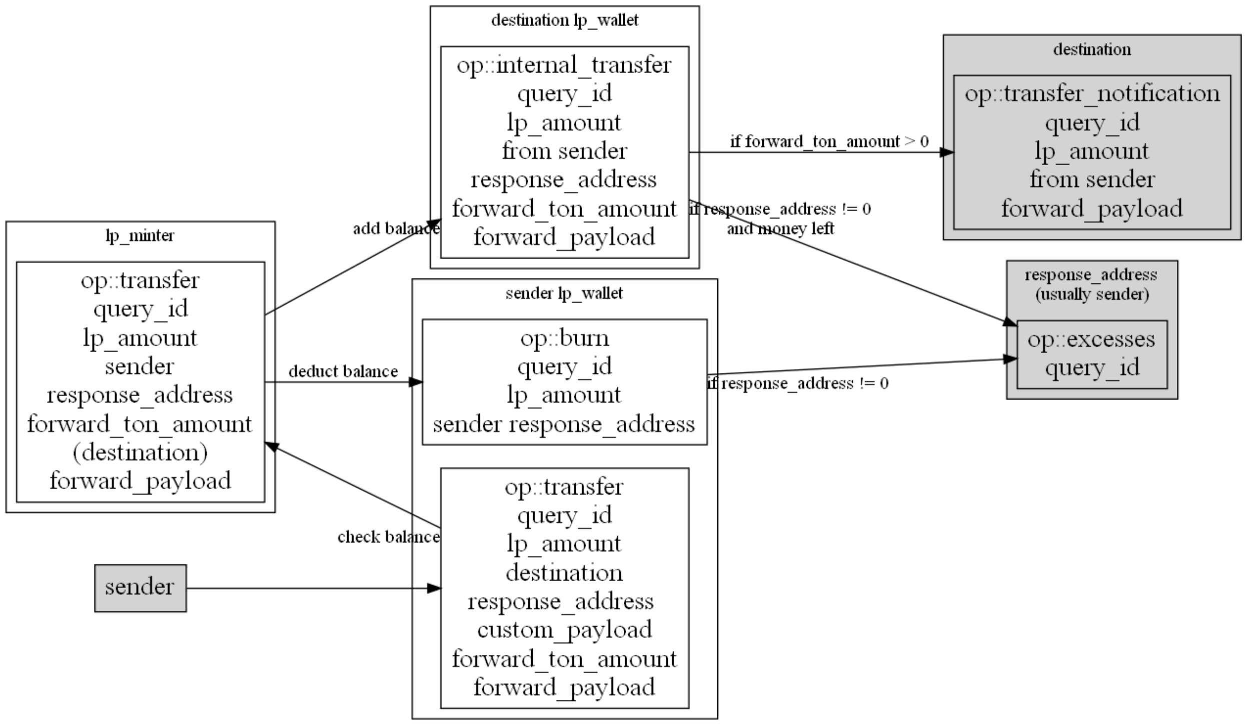
Recommendation

We recommend sending all the messages in bounceable mode unless the destination is expected to keep the TONs.

LPM-17 GAS MANAGEMENT IN `lp-minter::handle_transfer()` IS INCONSISTENT

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-minter.fc (base): 709~710	● Resolved

Description



`lp-minter` processes `op::transfer` this way:

1. `sender lp-wallet` checks that `msg_value > forward_ton_amount + fwd_count * fwd_fee + 2 * 0.01 + 0.01 + 0.2` and sends `op::transfer` to `lp-minter` carrying all the value.
2. `lp-minter::handle_transfer()` sends `op::burn` to `sender lp-wallet` with 0.03 TONs attached, and pays forwarding fees.
3. `lp-minter::handle_transfer()` sends `op::internal_transfer` to `destination lp-wallet` with `0.03 + forward_ton_amount` attached, pays forwarding fees.
4. `lp-minter::handle_transfer()` sends up to 2 `op::claim` to `router` with 0.04 TONs attached, and pays forwarding fees.

As a result, `const::lp_transfer_gas_consumption` (0.2 TON) is bigger than actually used. The excess is not returned to `response_address`. `lp-minter` will accumulate the value.

Recommendation

We recommend carrying all the remaining gas to `op::internal_transfer` .

LPM-18 | `to_jetton_address` IS NOT CHECKED IN `lp-minter::handle_transfer_notification()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 736~737, 835~836	● Acknowledged

Description

`lp-minter::handle_transfer_notification()` gets the payload `(from_jetton_address, to_jetton_address, destination, min_amount)` prepared by `router`. But `to_jetton_address` is not checked and is passed to `emit_log_cell_ref()` as is.

Recommendation

We recommend checking that `to_jetton_address == jettonB_address` (or `jettonA_address` depending on the branch).

LPM-19 | `lp-minter` SILENTLY ACCEPTS INCOMING LP TRANSFERS

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 920~921	● Resolved

Description

`lp-minter::handle_transfer_notification()` silently accepts incoming LP transfers. The funds become locked.

Recommendation

We recommend sending jettons back if they are not processed properly.

LPM-20 | `op::claim` EVENT EMITTED IN `lp-minter::handle_change_lp_mining_rate()`

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-minter.fc (base): 1051	● Resolved

Description

`lp-minter::handle_change_lp_mining_rate()` emits event with `op::claim` argument.

Recommendation

We recommend using `op::change_lp_mining_rate` argument.

LPM-21 | `min_amount` IS NOT RESPECTED BY `lp-minter::handle_mintable_notification()`

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-minter.fc (base): 320~321	● Resolved

Description

`min_amount` is supposed to disallow the user to have a too small LP balance. However, the minted amount is not checked in `lp-minter::handle_mintable_notification()`.

Recommendation

We recommend not minting LP if the resulting user LP balance is less, than `min_amount`.

LPM-22 | `lp-minter` ACCEPTS INCOMING TRANSFERS OF UNRECOGNIZED JETTONS

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-minter.fc (base): 923	● Resolved

Description

`lp-minter::handle_transfer_notification()` accepts incoming transfers of unrecognized jettons. The funds become locked.

Reverting the `op::transfer_notification` transaction will not return the funds.

The transfer is treated as unrecognized if valid `{ from_jetton, to_jetton }` payload was provided of known existing `lp-minter` , but the wrong jetton was actually sent to the `router` .

Recommendation

We recommend sending the jettons back.

LPW-04 | WRONG `fwd_count` CALCULATION

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-wallet.fc (base): 69~70	● Resolved

Description

```
69    int fwd_count = forward_ton_amount ? 3 : 1;
70    throw_unless(709, msg_value >
71                          forward_ton_amount +
72                          ;; 5 messages: wal1->minter, minter->wal1, minter->wal2,
wal2->owner, wal2->response
73                          ;; but last one is optional (it is ok if it fails)
74                          fwd_count * fwd_fee +
75                          (2 * const::gas_consumption + const::min_tons_for_storage
+ const::lp_transfer_gas_consumption));
```

As a result of `lp-wallet::send_tokens()` , 5 messages are generated: "wal1->minter, minter->wal1, minter->wal2, wal2->owner, wal2->response". The last one is optional. The message "wal2->owner" is not sent if `forward_ton_amount == 0` . The expected `fwd_count = forward_ton_amount ? 4 : 3` .

It is also expected that 4 message processing will be done. So, `const::lp_transfer_gas_consumption` is expected to be at least `2 * const::gas_consumption` .

Recommendation

We recommend updating the calculation of `fwd_count` .

LPW-05 | `jetton_address` IS NOT VALIDATED IN `lp-wallet::check_mintable()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/lp-wallet.fc (base): 201~208	● Resolved

Description

`lp-wallet::check_mintable()` expects `jetton_address` argument to be either `jettonA_address` , or `jettonB_address` . However, that is not enforced.

Recommendation

We recommend ensuring the address is one of the expected.

LPW-06 | `lp-wallet::on_bounce()` IS REDUNDANT

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/amm/lp-wallet.fc (base): 304~315	● Resolved

Description

`lp-wallet::on_bounce()` processes `op::internal_transfer` bounced message. However, it is never sent by `lp-wallet` .

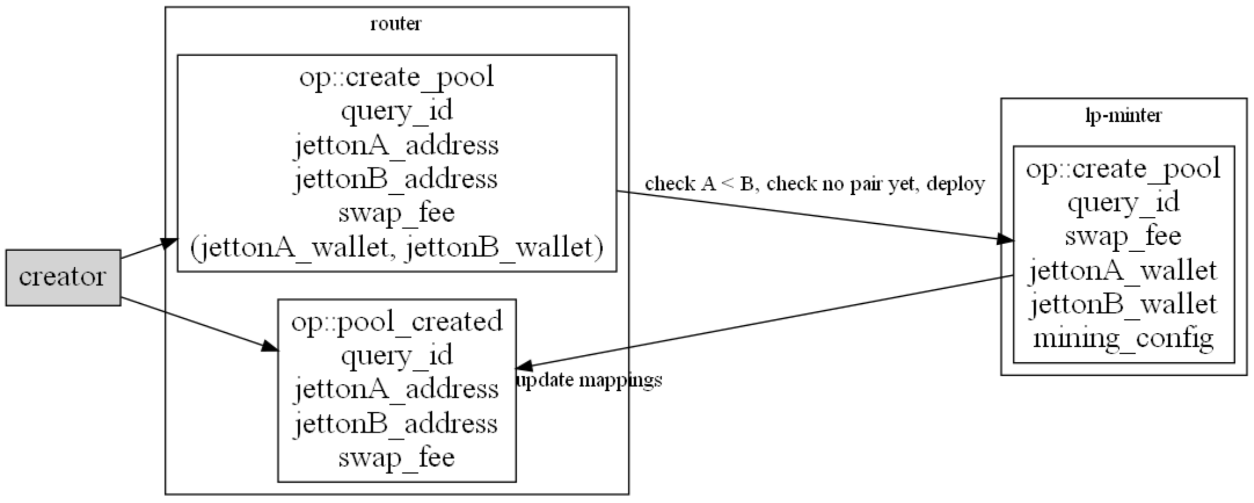
Recommendation

We recommend removing of unused code.

ROU-04 | `router` ALLOWS `op::pool_created` FROM `pool_creator_address`

Category	Severity	Location	Status
Control Flow	● Minor	contracts/amm/router.fc (base): 197~198	● Acknowledged

Description



`router` handles `op::pool_created` not only from `lp-minter` but also from `pool_creator_address`. This allows for skipping several important checks:

1. `pool_creator_address` can provide `jettonA_address` / `jettonB_address` arguments in the wrong order. `jetton_pair_to_lp` will still be updated with the wrong address.
2. `pool_creator_address` can forget to deploy the `lp-minter`.
3. `pool_count` will be incremented after each message.
4. `swap_fee` can be fake, it will still be emitted.

Recommendation

We recommend forbidding direct `op::pool_created` from `pool_creator_address`.

Alleviation

The team is planning to implement a factory contract that supports the creation of pools in the next version. Then, this `pool_creator_address` will be changed to that factory's address.

ROU-05 `router::handle_change_lp_mining_rate()` GAS CONSUMPTION IS INCONSISTENT

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/router.fc (base): 454~455	● Resolved

Description

`router::handle_change_lp_mining_rate()` checks that `msg_value` is at least `const::change_mining_rate_router_gas_consumption + pool_count * const::change_mining_rate_lp_gas_consumption` . That means, that each pool will have at least 0.1 TON for `op::change_lp_mining_rate` processing and `router` will have at least 1 TON for it.

However, the gas consumption of the `router` significantly depends on the `pool_count` :

1. The size of `new_lp_mining_rate_dict` depends on the `pool_count` .
2. The number of messages sent by the function also depends on the `pool_count` . The transfer fees are paid by `router` .

With a big enough `pool_count` the `const::change_mining_rate_router_gas_consumption` can be not enough to pay transfer fees.

`handle_change_mining_amount()` uses constants with the same names and is also affected.

Recommendation

We recommend:

1. Checking that `msg_value > pool_count * (const::change_mining_rate_router_gas_consumption + const::change_mining_rate_lp_gas_consumption)` .
2. Setting `const::change_mining_rate_router_gas_consumption = const::gas_consumption` .
3. Sending `op::change_lp_mining_rate` to `lp-minter` without `PAY_FEES_SEPARATELY` mode flag.
4. Renaming the constants to be more generic.

ROU-06 `jettonA_address` / `jettonB_address` CAN BE ARBITRARY, IRRELEVANT TO REAL JETTONS

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/amm/router.fc (base): 153~156	● Acknowledged

Description

`jettonA_address` / `jettonB_address` are provided by `pool_creator_address` with `op::create_pool` message to `router`. These addresses can be arbitrary, valid in `basechain`, and can be considered as "tags". They don't have to be related to `jettonA_wallet_address` / `jettonB_wallet_address`. They are used to generate the `lp-minter` address and the corresponding `lp-wallet` addresses. Swap operations must specify the same "tags" to get redirected to the same `lp-minter`.

Unlike `jettonA_address` / `jettonB_address`, wallets `jettonA_wallet_address` / `jettonB_wallet_address` are significant. `lp-minter` must be their owner for some unspecified jettons.

Recommendation

We recommend:

1. Providing the wallet code with `op::create_pool`
2. Validating the wallet addresses (and their real owner)

Or taking into account, and commenting the code correspondingly, that jetton addresses can be arbitrary.

Alleviation

The team is planning to implement factory contract that supports creation of pools in next version same as ROU-02. Then, this `pool_creator_address` will be changed to that factory's address. Checking will be done at factory contract.

UTI-01 | `mined()` CAN BE SIMPLIFIED

Category	Severity	Location	Status
Coding Style	● Minor	contracts/imports/utils.fc (base): 237~238	● Resolved

Description

`i` and `level` variables are redundant in `mined()`. `i < level` condition can be replaced with `half_life > 0`.

```
245    res = res + datetime_amount * (current_time - start_time + 1);
```

It is unclear, why one more second is added. For example, if `current_time = minable_time = start_time = 0`, the result is non-zero. We recommend clarifying the intended behavior and commenting the code.

Assignment operations (`+=`, `/=`) can be used to simplify the statements.

The function can be simplified to avoid redundant cycles and save gas.

Recommendation

We recommend rewriting the function this way:

```
int mined(int mining_amount, int minable_time, int datetime_amount, int half_life, int current_time) {
  int elapsed = current_time - minable_time;
  if (elapsed <= 0) {
    return 0;
  }

  int res = 0;
  if (half_life == 0) {
    ;; constant mining speed
    res = datetime_amount * elapsed;
  } else {
    ;; mining speed for the current period
    int datetime_amount_now = datetime_amount >> (elapsed / half_life);
    ;; mined for all full periods
    res = (datetime_amount - datetime_amount_now) * 2 * half_life;
    ;; mined in current period
    res += datetime_amount_now * (elapsed % half_life);
  }

  ;; limit the result by mining_amount
  if ((mining_amount > 0) & (res > mining_amount)) {
    res = mining_amount;
  }

  return res;
}
```

CON-03 | MISLEADING COMMENTS

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/amm/lp-minter.fc (base): 85~86; contracts/amm/lp-wallet.fc (base): 180; contracts/amm/router.fc (base): 573, 621, 626; contracts/jetton-minter.fc (base): 92~93	● Resolved

Description

```
180        .store_uint(0x10, 6) ;; nobounce - int_msg_info$0 ihr_disabled:Bool
bounce:Bool bounced:Bool src:MsgAddress -> 011000
```

The comment states `011000`, however, `010000` flags are used.

```
92      ;; NOTE : bridge minter        jetton   custom_payload
```

The comments should be in English.

```
85    ;; sender_address can be admin or router
```

In fact, only messages from the router are accepted by `handle_create_pool()`.

```
573    send_raw_message(msg.end_cell(), 64); ;; pay transfer fees separately, revert
on errors
```

In fact, the mode is `CARRY_REMAINING_GAS | REVERT_ON_ERRORS`.

```
626    if (op == op::change_lp_policy_admin) { ;; NOTE : swap_fee, min_amount admin
```

In fact, it is not possible to change `swap_fee`. `NODE` is supposed to be `NOTE`.

Recommendation

We recommend updating the comments.

IMP-01 | UNUSED CODE

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/imports/op-codes.fc (base): 4~5; contracts/imports/utils.fc (base): 5~14	● Resolved

Description

These functions and variables are unused:

- `utils::send_grams()`
- `message_utils::send_receipt_message()`
- `op::change_next_admin`
- `message_utils::send_text_receipt_message()`
- `message_utils::emit_log_simple()`
- `const::claim_gas_consumption`

Recommendation

We recommend removing of unused code.

LPM-23 | `update_mining_index()` CAN BE REFACTORED

Category	Severity	Location	Status
Coding Style	● Informational	contracts/amm/lp-minter.fc (base): 124, 712	● Acknowledged

Description

`lp-minter::update_mining_index()` contains code repetitions. This decreases code readability and maintainability. Subroutine `update_mining_index_for_mining_rate()` can be created and used 3 times.

`lp-minter::handle_transfer_notification()` contains code repetitions, it can be significantly refactored to increase code readability and maintainability.

Recommendation

We recommend refactoring the functions. We recommend adding helper functions that prepare and send common messages.

LPM-24 | USAGE OF MAGIC NUMBERS

Category	Severity	Location	Status
Coding Style	● Informational	contracts/amm/lp-minter.fc (base): 591~593	● Acknowledged

Description

Different magic numbers are used as-is in code.

Recommendation

We recommend declaring constants to improve code maintainability and readability.

- SWAP_FEE_SCALE_FACTOR = 10000
- MINING_INDEX_SCALE_FACTOR = 1000000000000000000
- mode::REVERT_ON_ERRORS = 0
- mode::PAY_FEES_SEPARATELY = 1
- mode::IGNORE_ERRORS = 2
- mode::CARRY_REMAINING_GAS = 64
- etc.

LPM-25 | `in_msg_body` IS UNUSED IN `lp-minter::handle_claim()`

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/amm/lp-minter.fc (base): 929, 962	● Resolved

Description

`in_msg_body` argument is unused in `lp-minter::handle_claim()` and `lp-minter::handle_pending_jetton()`.

Recommendation

We recommend removing of unused arguments.

LPM-26 `op::change_router` CAN'T BE HANDLED PROPERLY BY `lp-minter`

Category	Severity	Location	Status
Volatile Code	● Informational	contracts/amm/lp-minter.fc (base): 1194~1198	● Acknowledged

Description

`lp-minter` allows to `op::change_router`. The router address is an argument of `calculate_lp_minter_state_init()`, so, defines the `lp-minter` address. `lp-minter` address is used by `router::create_pool()` and `router::pool_created()`.

`lp-minter` with a changed router address can't be added to another `router`, because it will have an address based on the old router value. The router is allowed to change `swap_fee` and mining configuration, so, one can change it to EOA, change the configuration, change the router back, and `op::claim` more, than expected.

Recommendation

We recommend removing of `op::change_router` message handling.

Alleviation

[**Megaton**]: If we have to change the router contract in the future, the new router contract will have a new op to migrate the previous lp contract. And the address of the previous lp contract will be handled via the admin address.

[**CertiK**]: Only the `router` can now set the new router address. The severity was lowered to the Informational level.

OPC-01 | RESPONSE MESSAGES `op` DON'T HAVE HIGH-ORDER BIT SET

Category	Severity	Location	Status
Coding Style	● Informational	contracts/imports/op-codes.fc (base): 22, 32	● Resolved

Description

Section 5 of the Internal Messages Guidelines recommends the "response" messages to have an `op` with the high-order bit set, i.e., in the range `2^31 .. 2^32-1`. This allows the contracts to ignore all the unhandled response messages easily.

`op::pool_created` is the response for `op::create_pool`.

`op::check_mintable_notification` is the response for `op::check_mintable`.

These op-codes have high-order bit unset.

Recommendation

We recommend updating the op-codes in accordance with the Guidelines.

ROU-07 | ARGUMENT NAMES OF `router::get_lp_address()` ARE MISLEADING

Category	Severity	Location	Status
Coding Style	● Informational	contracts/amm/router.fc (base): 92~93	● Resolved

Description

```
(slice, int) get_lp_address(slice jettonA_address, slice jettonB_address, cell
jetton_pair_to_lp) {
```

`jettonA_address` and `jettonB_address` argument names are misleading. The addresses can be in another order.

Recommendation

We recommend renaming the arguments to `jetton1_address` , `jetton2_address` for better code maintainability.

ROU-08 | `either_forward_payload` VARIABLE IS UNUSED

Category	Severity	Location	Status
Coding Style	● Informational	contracts/amm/router.fc (base): 373~374	● Resolved

Description

`either_forward_payload` local variable is never used.

Recommendation

We recommend removing of unused variables.

UTI-02 | `calculate_jetton_wallet_address()` CAN BE REPLACED WITH `calculate_contract_address()`

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/imports/utils.fc (base): 40~48	● Resolved

Description

`calculate_jetton_wallet_address()` can be removed. Universal `calculate_contract_address()` can be used instead.

Recommendation

We recommend removing of redundant code.

UTI-03 | LONG AND COMPLICATED MESSAGE BUILDING STATEMENTS CAN BE FORMATTED

Category	Severity	Location	Status
Coding Style	● Informational	contracts/imports/utils.fc (base): 132~133	● Acknowledged

Description

```
132      .store_dict(pack_lp_minter_data(0, 0, 0, admin_address, router_address,
jettonA_address, jettonA_address, 0, 0, jettonB_address, jettonB_address, 0, 0, 0,
0, 0, 0, 0, 0, begin_cell().store_uint(0, 32 + 64).end_cell(), new_dict(),
lp_default_content, lp_wallet_code))
```

Some statements are huge and complicated. That decreases readability and maintainability.

Recommendation

We recommend formatting of long statements using new lines and indentation.

UTI-04 | `calculate_jetton_minter_address()` IS UNUSED AND DANGEROUS

Category	Severity	Location	Status
Volatile Code	● Informational	contracts/imports/utils.fc (base): 63~72, 82~85	● Resolved

Description

`calculate_jetton_minter_state_init()` and `calculate_jetton_minter_address()` are unused.

`calculate_jetton_minter_address()` should not be used to discover the `jetton-minter` address. It uses `admin_address` , `minter_address` , and `content` as arguments, which can be updated by the `jetton-minter` contract. As a result, only providing original values will give the same `jetton-minter` address.

Recommendation

We recommend removing of unused functions.

OPTIMIZATIONS | MEGATON FINANCE - AUDIT 1

ID	Title	Category	Severity	Status
CON-04	Constants Can Be Used Instead Of `PUSHINT`	Gas Optimization	Optimization	● Resolved

CON-04 | CONSTANTS CAN BE USED INSTEAD OF `PUSHINT`

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/imports/utils.fc (base): 15~16; contracts/jetton-wallet.fc (base): 21~23	● Resolved

Description

```
int min_tons_for_storage() asm "10000000 PUSHINT"; ;; 0.01 TON
int gas_consumption() asm "10000000 PUSHINT"; ;; 0.01 TON
```

According to the documentation, numeric constants are substituted during compilation, so all optimization and pre-computations performed during the compilation are successfully performed (unlike the old method of defining constants via inline asm `PUSHINT` s).

Recommendation

We recommend declaring the constants.

APPENDIX | MEGATON FINANCE - AUDIT 1

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | **Securing** the **Web3** World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.