Guvenkaya® The Bedrock of Security

Spin
NEAR Rust Smart Contract Security Assessment

Lead Security Engineer: Timur Guvenkaya
Date of Engagement: 15th April 2024 - 28th June 2024
Visit: www.guvenkaya.co

Contents

About Us 01
About Spin 01
Audit Results 02
.1 Project Scope 02
.2 Out of Scope 16
.3 Timeline 16
Methodology 17
Severity Breakdown 18
.1 Likelihood Ratings 18
.2 Impact 18
.3 Severity Ratings 18
.4 Likelihood Matrix 19
.5 Likelihood/Impact Matrix 19
Findings Summary 20
Findings Details 23
.1 GUV-1 Market-wide Denial of Service via Log Limit Exceeding - Critical 23
.2 GUV-2 Any account can unregister another account - Critical 26
.3 GUV-3 Order Placement with Negative/Zero Margin Ratio Is Possible - Critical 27
.4 GUV-4 Wrapping casting overflow leads to the invalid storage deposit - High 29
.5 GUV-5 Storage Withdraw Does Not Refund User Tokens - High 30
.6 GUV-6 Storage Bloating - High 32
.7 GUV-7 Intention of the function does not match with implementation - Medium 33
.8 GUV-8 Usage of Non-Maintained Oracle - Medium 35
.9 GUV-9 Orders are unprocessable due to log limit during effects processing - Medium 36
.10 GUV-10 Vault can be set to incorrect values via update_settings - Medium 39

Findings Details 23

.11 GUV-11 Vault Min Capacity Is Not Asserted - Medium 40

.12 GUV-12 Management Fee Is Not Asserted - Low 41

.13 GUV-13 Token id Collision Leads To Two Different Vaults Using The Same OFT - Low 42

.14 GUV-14 Metadata Is Not Validated During Vault Creation - Low 47

.15 GUV-15 Storage Withdraw Is Not Protected With 2fa - Low 49

.16 GUV-16 Better Contract Upgrade Mechanism Can Be Implemented - Informational 50

.17 GUV-17 Casting overflow via as_decimals() when the account cannot be liquidated - Informational 56

.18 GUV-18 .view() method leads to casting overflow in get_positions due to as_decimals usage in Fraction Display implementation - Informational 58

.19 GUV-19 Suboptimal Assertion - Informational 61

.20 GUV-20 Redundant Prepaid Gas Check - Informational 62

.21 GUV-21 Contract Storage Can Be Optimized - Informational 63

.22 GUV-22 Not Resolved TODOs In The Codebase - Informational 68

About Us

Guvenkaya is a security research firm specializing in Rust security, Web3 security of Rust-based protocols, and Web2 security. With our expertise, we provide both security auditing services and custom security solutions

About Spin

Founded in June 2021, Spin was the first DEX Orderbook in the NEAR ecosystem to release Perpetual Futures and DeFi Option Vaults. Spin's mission is extending DeFi services offered on NEAR Protocol also attracting & onboarding the CeFi audience to on-chain trading and investments.

Audit Results

Guvenkaya conducted a security assessment of the Spin spot, perp, and options-vault contracts from 15th of April 2024 to 28th June 2024. During this engagement, a total of 22 findings were reported. 3 of the findings were critical, 3 high, 5 medium, and the remaining were low or informational severity. All major issues are yet to be fixed by the Spin team.

Project Scope

Lines of Code Reviewed: 21770

Spot Contract

File name	Link
marketplace.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/marketplace.rs
market.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/market.rs
lib.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/spot/contract/src/lib.rs
event.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/event.rs
list.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/order_book/list.rs
book.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/order_book/book.rs

File name	Link
accounts.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/accounts.rs
buffer.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/buffer.rs
currency.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/currency.rs
map.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/map.rs
limits.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/limits.rs
order.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/order_book/order.rs
collector.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/order_book/collector.rs
reverse.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/reverse.rs
mod.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/mod.rs

File name	Link
heap.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/heap.rs
platform.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/spot/contract/src/platform.rs
set.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/types/set.rs
keys.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/keys.rs
ft.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/spot/contract/src/ft.rs
order_book/mod.rs	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/order_book/mod.rs
lib.rs (marketplace)	https://github.com/spin-fi/near-dapps/tree/4596971fefe1646512bd66116b725cee2a14aeac/marketplace/src/lib.rs

Perp Contract

File name	Link
contract.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/contract.rs
market.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/markets/market.rs
liquidation.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/liquidation.rs
position.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/perp/position.rs
log.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/events/log.rs
position.rs (risk)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/position.rs
buffer.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/buffer.rs
event.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/events/event.rs
lib.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/contract_api/src/lib.rs
mod.rs (perp)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/perp/mod.rs

File name	Link
map.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/map.rs
account.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/accounts/account.rs
funding.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/funding.rs
orders.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/perp/orders.rs
markets.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/perp/markets.rs
api.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/api.rs
storage.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/markets/storage.rs
margin.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/margin.rs
epoch.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/epoch.rs
set.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/set.rs

File name	Link
mod.rs (accounts)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/accounts/mod.rs
times.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/times.rs
currency.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/currency.rs
currency.rs (perp)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/perp/currency.rs
o_chain.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/platform/src/o_chain.rs
positions.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/positions.rs
cache.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/cache.rs
reverse.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/reverse.rs
holder.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/accounts/holder.rs
heap.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/heap.rs

File name	Link
near.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/platform/src/near.rs
flush.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/flush.rs
lib.rs (platform)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/platform/src/lib.rs
lib.rs (ft)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/ft/src/lib.rs
state.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/state.rs
info.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/markets/info.rs
lib.rs (types)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/lib.rs
keys.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/keys.rs
zlib.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/platform/src/zlib.rs

File name	Link
mod.rs (markets)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/markets/mod.rs
lib.rs (perp/common)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/common/src/lib.rs
lib.rs (perp/contract)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/lib.rs
mod.rs (risk)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/risk/mod.rs
number/mod.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/number/mod.rs
number/zero.rs	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/core/types/src/number/zero.rs
mod.rs (events)	https://github.com/spin-fi/near-dapps/tree/e558c0ed54b376ab07dd6a7f677b0ea36950e4f4/perp/contract/src/events/mod.rs

Options Vault Contract

File name	Link
bindgen.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/bindgen.rs
auction.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/spot/auction/src/auction.rs
vault.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/vault.rs
market.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/spot/auction/src/market.rs
order.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/order.rs
v1.rs (vault events)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/vault/src/v1.rs
list.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/list.rs
log.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/log.rs
book.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/book.rs

File name	Link
methods.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/ofts/methods.rs
mod.rs (price_feed)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/feed_manager/src/price_feed/mod.rs
log.rs (auction)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/spot/auction/src/log.rs
state_mashine.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/methods/state_mashine.rs
users.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/vault/src/users.rs
lib.rs (serializers)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/utils/serializers/src/lib.rs
api.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/api.rs
lib.rs (system events)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/system/src/lib.rs
lib.rs (transfer)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/transfer/src/lib.rs
mod.rs (vaults)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/vaults/mod.rs

File name	Link
collector.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/collector.rs
zero_copy.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/utils/drop_map/src/zero_copy.rs
management.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/vaults/management.rs
accounts.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/vaults/accounts.rs
mft.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/methods/mft.rs
mod.rs (epochs)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/vault/src/epochs/mod.rs
fees.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/vault/src/fees.rs
v1.rs (spot events)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/spot/src/v1.rs
feed.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/feed_manager/src/feed.rs
mod.rs (ofts)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/ofts/mod.rs

File name	Link
storage.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/storage_manager/src/storage.rs
api.rs (auction)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/spot/auction/src/api.rs
management.rs (vault)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/vault/src/management.rs
lib.rs (events primitives)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/primitives/src/lib.rs
epoch.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/vault/src/epochs/epoch.rs
events.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/data/src/events.rs
views.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/spot/auction/src/views.rs
lib.rs (drop_map)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/utils/drop_map/src/lib.rs
id_parser.rs	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/id_parser.rs
lib.rs (vault events)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/vault/src/lib.rs

File name	Link
lib.rs (counter)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/utils/counter/src/lib.rs
lib.rs (spot events)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/spot/src/lib.rs
lib.rs (order_book)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/lib.rs
zlib.rs (events data)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/data/src/zlib.rs
keys.rs (storage_manager)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/storage_manager/src/keys.rs
keys.rs (order_book)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/order_book/src/keys.rs
lib.rs (pause)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/utils/pause/src/lib.rs
lib.rs (oft)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/oft/src/lib.rs
log.rs (events data)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/data/src/log.rs
lib.rs (storage_manager)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/components/storage_manager/src/lib.rs

File name	Link
lib.rs (dependencies)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/dependencies/src/lib.rs
lib.rs (events data)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/events/data/src/lib.rs
mod.rs (methods)	https://github.com/spin-fi/near-dapps/tree/85c985f95656437f0bef0273f86b6d55e10b194c/option-vault/src/methods/mod.rs

Out of Scope

The audit will include, but is not limited to, reviewing the code for security vulnerabilities, coding practices, and architecture. The audit does not include a review of the dependencies.

Timeline

Start of the audit	Draft Report	Final Report
15th April 2024	4th July 2024	3rd September 2024

Methodology

- RESEARCH INTO PROJECT ARCHITECTURE
- PREPARING ATTACK VECTORS
- SETTING UP AN ENVIRONMENT
- MANUAL CODE REVIEW OF THE CODE
- ASSESSMENT OF RUST SECURITY ISSUES
- ASSESSMENT OF NEAR SECURITY ISSUES
- ASSESSMENT OF ARITHMETIC ISSUES
- BUSINESS LOGIC VULNERABILITY ASSESSMENT
- ONCHAIN TESTING USING NEAR WORKSPACES
- BEST PRACTICES AND CODE QUALITY
- CHECKING FOR CODE REFACTORING/SIMPLIFICATION POSSIBILITIES
- ARCHITECTURE IMPROVEMENT SUGGESTIONS
- PREPARING POCS AND/OR TESTS FOR EACH CRITICAL/HIGH/MEDIUM ISSUES

Severity Breakdown

01. Likelihood Ratings

Likely: The vulnerability is easily discoverable and not overly complex to exploit.
Possible: The vulnerability presents some challenges either in discovery or in the complexity of the attack.
Rare: The vulnerability is either very difcult to discover or complex to exploit, or both.
This matrix provides a nuanced view, taking into account both the ease of discovering a vulnerability and the complexity involved in exploiting it.

02. Impact

Severe: The vulnerability is easily discoverable and not overly complex to exploit.
Moderate: The vulnerability presents some challenges either in discovery or in the complexity of the attack.
Negligible: The vulnerability is either very difcult to discover or complex to exploit, or both.

03. Severity Ratings

Critical: Assigned to vulnerabilities with severe impact and a likely likelihood of exploitation.
High: For vulnerabilities with either severe impact but only a possible likelihood, or moderate impact with a likely likelihood.
Medium: Used for vulnerabilities with severe impact but a rare likelihood, moderate impact with a possible likelihood, or negligible impact with a likely likelihood.
Low: For vulnerabilities with moderate impact and rare likelihood, or negligible impact with a possible likelihood.
Informational: The lowest severity rating, typically for vulnerabilities with negligible impact and a rare likelihood of exploitation.

CRITICAL **HIGH** **MEDIUM** Low Informational

Likelihood Matrix:

Attack Complexity \ Discovery Ease	Obvious	Concealed	Hidden
Complex	Possible	Rare	Rare
Moderate	Likely	Possible	Rare
Straightforward	Likely	Possible	Possible

Likelihood/Impact Matrix:

Likelihood \ Impact	Severe	Moderate	Negligible
Likely	**CRITICAL**	**HIGH**	**MEDIUM**
Possible	**HIGH**	**MEDIUM**	Low
Rare	**MEDIUM**	Low	Informational

Findings Summary

01. Remediation Complexity: This measures how difcult it is to fx the vulnerability once it has been identifed.

Simple: Patches or fixes are readily available and easily implemented.
Moderate: Requires some time and resources to remediate, but well within the capabilities of most organizations.
Difficult: Remediation requires significant resources, specialized skills, or substantial changes to systems or architecture.

02. Status: This measures how difcult it is to fx the vulnerability once it has been identifed.

Not Fixed: Indicates that the vulnerability has been identifed but no remedial action has been taken yet. This status is crucial for newly discovered vulnerabilities or those awaiting prioritization.
Fixed: This status is applied when the vulnerability has been successfully remediated. It implies that appropriate measures (like patching, confguration changes, or architectural modifcations) have been implemented to resolve the issue.
Acknowledged: This status is used for vulnerabilities that have been recognized, but for various reasons (such as risk acceptance, cost, or other business decisions), have not been fxed. It indicates that the risk posed by the vulnerability is known and has been consciously accepted.

Finding	Impact	Likelihood	Severity	Remediation Complexity	Remediation Status
GUV-1: Market-wide Denial of Service via Log Limit Exceeding	Severe	Likely	CRITICAL	Simple	Fixed
GUV-2: Any account can unregister another account	Severe	Likely	CRITICAL	Simple	Fixed
GUV-3: Order Placement with Negative/Zero Margin Ratio Is Possible	Severe	Likely	CRITICAL	Simple	Fixed
GUV-4: Wrapping casting overflow leads to the invalid storage deposit	Moderate	Likely	HIGH	Simple	Fixed
GUV-5: Storage Withdraw Does Not Refund User Tokens	Moderate	Likely	HIGH	Simple	Fixed
GUV-6: Storage Bloating	Severe	Possible	HIGH	Moderate	Acknowledged
GUV-7: Intention of the function does not match with implementation	Moderate	Possible	MEDIUM	Simple	Acknowledged
GUV-8: Usage of Non-Maintained Oracle	Moderate	Possible	MEDIUM	Moderate	Acknowledged
GUV-9: Orders are unprocessable due to log limit during effects processing	Severe	Rare	MEDIUM	Moderate	Fixed
GUV-10: Vault can be set to incorrect values via update_settings	Severe	Rare	MEDIUM	Moderate	Acknowledged
GUV-11:Vault Min Capacity Is Not Asserted	Moderate	Possible	MEDIUM	Simple	Fixed
GUV-12: Management Fee Is Not Asserted	Negligible	Possible	Low	Simple	Acknowledged

Finding	Impact	Likelihood	Severity	Remediation Complexity	Remediation Status
GUV-13: Token id Collision Leads To Two Different Vaults Using The Same OFT	Negligible	Possible	Low	Simple	Acknowledged
GUV-14: Metadata Is Not Validated During Vault Creation	Negligible	Possible	Low	Simple	Acknowledged
GUV-15: Storage Withdraw Is Not Protected With 2fa	Negligible	Possible	Low	Simple	Fixed
GUV-16: Better Contract Upgrade Mechanism Can Be Implemented	Negligible	Rare	Informational	Moderate	Acknowledged
GUV-17: Casting overflow via as_decimals() when the account cannot be liquidated	Negligible	Rare	Informational	Moderate	Acknowledged
GUV-18: .view() method leads to casting overflow in get_positions due to as_decimals usage in Fraction Display implementation	Negligible	Rare	Informational	Moderate	Acknowledged
GUV-19: Suboptimal Assertion	Negligible	Rare	Informational	Simple	Acknowledged
GUV-20: Redundant Prepaid Gas Check	Negligible	Rare	Informational	Simple	Fixed
GUV-21: Contract Storage Can Be Optimized	Negligible	Rare	Informational	Simple	Acknowledged
GUV-22: Not Resolved TODOs In The Codebase	Negligible	Rare	Informational	Simple	Acknowledged

Findings Details

GUV-1 Market-wide Denial of Service via Log Limit Exceeding - Critical

We observed that it is possible to cause DoS in a market by placing a certain number of limit orders with a short expiry time. When expired orders are handled, we push several events to the event buffer, such as balance_update & order_update. After a certain number of expired orders, we will exceed the limit of 16kb per event. This will cause panic, and no one will be able to place an opposite order in a market (ask or bid) since expired orders cannot be processed Cost of an attack varies depending on the market settings. Also, it depends on market activity since any order can trigger processing of expired orders before an attack has reached a critical number.

We refund the users and send order event

spot:marketplace:handle_expired_orders:marketplace/src/market.rs

```
fn handle_expired_orders(
    &mut self,
    balances: &mut Balances<P>,
    orders: &[Order],
  ) {
    for order in orders {
      self.refund(balances, order);
      Accounts::<P>::remove_order(&order.acc, self.id, order.id);
     self.send_order_event(order, OrderStatus::Expired);
    }
  }
```

In **self.refund** we do balances.deposit, where we push balance_update event

spot:marketplace:deposit:marketplace/src/accounts.rs

```
    pub fn deposit(
        &mut self,
        account_id: &AccountId,
        token: u16,
        amount: Balance,
        event: BalanceChangeInfo,
    ) -> Balance {
        assert!(
            amount > U128(0),
            "Expected positive amount. User: {}, token: {}, amount: {}",
            account_id,
            token,
            amount.0
        );

        // Loading balance and adding amount.
        let balance = self.load_balance_mut(account_id, token);
        *balance += amount;

        // Emitting event.
       event::balance_update(
         account_id.to_string(),
         token,
         amount,
         true,
         event,
       );
        *balance
    }
```

POC: **guv_1.rs**

PROPOSED SOLUTION

Instead of accumulating all events into one big event, you could split them into several. NEAR has a limit of 100 events per transaction, so you can determine which events should go into a big one and which should be immediately emitted.

REMEDIATION - FIXED

The Spin team has fixed the issue in the next implementation of the spot contract.

GUV-2 Any account can unregister another account - Critical

It was observed that the **storage_withdraw** has exposed**account_id** parameter, which allows any account to withdraw the storage of another account.

vault:storage_withdraw:option-vault/src/bindgen.rs

```
fn storage_withdraw(
   &mut self,
  account_id: &AccountId,
   amount: Option<u64>,
   unregister: Option<bool>,
) -> u64 {
   self.call(|vault| vault.storage_withdraw(account_id, amount, unregister))
}
```

POC: **guv_2.rs**

PROPOSED SOLUTION

We propose removing the **account_id** parameter and use **env::predecessor_account_id** instead

REMEDIATION - FIXED

The Spin team has fixed the issue by removing the account_id parameter

GUV-3 Order Placement with Negative/Zero Margin Ratio Is Possible - Critical

It was observed that it is possible to continue placing orders even if the margin ratio is negative. This issue occurs because the**can_place_order** function only checks if the current margin ratio is greater than or equal to the previous margin ratio, or if it is at least 1.0. This logic fails to prevent order placement when both the current and previous ratios are negative, allowing users with negative margins to continue trading, which can lead to further losses and increased risk for the system

perp:can_place_order:perp/contract/src/risk/margin.rs

```
    pub fn can_place_order(&self, previous: MarginInfo) -> bool {
     (!previous.ratio.is_nan() && self.ratio >= previous.ratio) || self.ratio >=
Fraction::one()
    }
```

previous_ratio is set inside of **before_place_order** by via margin_ratio.

perp:before_place_order:perp/contract/src/markets/market.rs

```
    fn before_place_order(
        &mut self,
        context: &mut (MarginInfo, State<P>),
        acc_data: &mut Accounts<P>,
        order: &Order,
    ) {
        let prelaunch_markets = Set::<MarketID,
P>::new(StorageKeyEnum::PrelaunchMarkets);
        if !prelaunch_markets.contains(&self.id) {
            self.limits.validate_order_price(order, self.index_price());
        }
     context.0 = self.margin_ratio(acc_data, &order.acc, &context.1);}
```

Scenario

1. Alice and Bob each deposit 55k tokens.
2. The index price is set to 45k.
3. Alice places a bid order at 45k for 10 units.
4. Bob places an ask order at 45k for 10 units.
5. The price moves up to 55k, causing Bob to face liquidation.
6. Despite Bob having a negative margin, he can still place new orders and match them with Alice's orders, further increasing his losses.

POC: **guv_3.rs**

PROPOSED SOLUTION

We propose verifying whether the current margin ratio is greater than Fraction::zero() before allowing the order to be placed.

REMEDIATION - FIXED

The Spin team has fixed the issue by verifying whether the ratio is larger than Fraction::zero() in the **can_place_order** function

GUV-4 Wrapping casting overflow leads to the invalid storage deposit - High

It was observed that in the **storage_deposit env::attached_deposit()** is casted to the u64. Since attached_deposit is u128 casting it to u64 via "as" keyword will lead to the wrapping casting overflow if the attached deposit is larger than the u64::MAX. It is easily possible since u64::MAX in NEAR terms is **000018446744073709554N** (u64::MAX / 10**24). It leads to the invalid value put as a storage deposit

vault:storage_deposit:option-vault/src/bindgen.rs

```
#[payable]
fn storage_deposit(&mut self, to: Option<AccountId>) {
    let caller = env::predecessor_account_id();
    let deposit = env::attached_deposit();

    self.call(|vault| vault.storage_deposit(&caller, deposit as u64, to.as_ref()))
}
```

POC: **guv_4.rs**

PROPOSED SOLUTION

Make sure to check whether the attached deposit is not larger than u64::MAX before casting it to u64 or utilize panicking casting methods.

REMEDIATION - FIXED

The Spin team has fixed the by casting to the u64 via **try_into()**.

GUV-5 Storage Withdraw Does Not Refund User Tokens - High

It was observed that the storage_withdraw only changes the internal state but does not refund NEAR back to the user

vault:storage_withdraw:components/storage_manager/src/storage.rs

```
    fn storage_withdraw(&mut self, storage_amount: Option<u64>, unregister:
Option<bool>) -> u64 {
        let storage = self.get().expect("Account is not registered");

        let available_amount = storage.available_amount();
        let should_unregister = unregister.unwrap_or(false);

        if should_unregister {
            self.new_value = None;
            available_amount
        } else {
            let withdraw_amount = storage_amount.unwrap_or(available_amount);
            let account_size = self.map.borrow().measured_storage_size;
            assert!(
                withdraw_amount <= available_amount + account_size,
                "Not enough prepayed storage"
            );
            storage.sub(withdraw_amount);
            self.new_value = Some(storage);
            withdraw_amount
        }
    }
```

PROPOSED SOLUTION

It is advised to refund user's NEAR when the storage was withdrawn.

REMEDIATION - FIXED

The Spin team has fixed the issue by disabling storage withdraw via **unimplemented!()**

GUV-6 Storage Bloating - High

It was observed that the perp contract does not delete the user data, when the user balance hits 0. When user does not hold any position, user can simply deposit certain amount and immediately withdraw, which will still leave a user entry in the contract. By constantly repeating this action, a malicious actor can bloat the storage, which eventually will lead to the lack of funds for a state error in the contract. Every execution storage increases by**0.00329N** if user account length is 64

POC: **guv_6.rs**

PROPOSED SOLUTION

It is advised to delete user accounts from storage once their balance hits zero.

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue saying that there are much functionality depends on this behaivor and also that It makes sense to review this again, look at the limitations of Near on storage and calculate the potential throughput of users of our contract and also check whether any functions depend on the number of users on storage

GUV-7 Intention of the function does not match with implementation - Medium

It was observed that round method in Decimals, according to the comment, supposed to **round up** the result. However, due to the nature of division it actually rounds it down.

vault:round_up_quantity:marketplace/src/market/limits.rs

```
/// Round up the passed quantity.
pub fn round_up_quantity(&self, quantity: Decimal) -> Decimal {
  quantity.round(self.step_size)
}
```

vault:round:core/types/src/number/decimal.rs

```
/// Round up self value by using the given base.
/// self / base * base
pub fn round(&self, base: Decimal) -> Decimal {
    if base.is_zero() {
        return *self;
    }

    Decimal {
      val: self.val / base.val * base.val,
    }
}
```

PROPOSED SOLUTION

It is recommended to use **div_ceil** to round the decimals up. Alternatively, the comment has to be changed

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-8 Usage of Non-Maintained Oracle - Medium

It was observed that the perp contract uses First-Party Flux oracle, which is no longer maintained.

PROPOSED SOLUTION

It is advised to migrate over **priceoracle.near**.

More details: https://docs.near.org/build/primitives/oracles

Price feeds: https://github.com/NearDeFi/near-price-oracle-bot/tree/main/feeds

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-9 Orders are unprocessable due to log limit during effects processing - Medium

We observed that due to the log limit, orders which cause the effect to apply to > 10 other orders will fail. It is important to note that it does not include processing expired orders. If the processing was also handling expired orders, the apply_to vector could be even smaller to cause failure

On each apply_to element, we handle_execution effects and then send the order event to be added to the emitted log.

spot:place_limit_order:marketplace/src/market.rs

```
if !res.effects.apply_to.is_empty() {
    self.handle_execution_effects(balances, &res.effects, false);

  self.send_order_event(
    &res.effects.applied,
    if res.effects.is_filled {
      OrderStatus::Filled
    } else {
      OrderStatus::PartiallyFilled
    },
  );
} else {
    self.send_order_event(&res.effects.applied, OrderStatus::New);
}
```

Also inside of handle_execution_effects we push the trade_update event

spot:handle_execution_effects:marketplace/src/market.rs

```
let (taker_fee, taker_fee_token_id) = match fees_info.taker_fee_currency {
    FeeCurrency::Base => (
        fees_info.taker_fee.scale(self.base.decimals).value(),
        self.base.id,
    ),
    FeeCurrency::Quote => (
        fees_info.taker_fee.scale(self.quote.decimals).value(),
        self.quote.id,
    )
};

event::trade_update(
    self.id,
    order.id,
    effects.applied.id,
    maker_fee,
    maker_fee_token_id,
    taker_fee,
    taker_fee_token_id,
    fees_info.is_rebate,
    price.value(),
    quantity.value(),
    effects.applied.o_type,
);

self.send_release_funds_event(order, quantity, price);
```

POC: **guv_9.rs**

Possible Exploitation

1. This case could be exploited if we put many limit orders with a small price & quantity. This could cause some opposite orders to fail since there is a high chance that they will reach critical apply_to count (not counting if there are any expired orders)

2. This case could be exploited by front running certain transactions to influence the orderbook and prevent certain users from executing certain orders.

  a. Alice puts an ask order with quantity of 100 and price as 1N

  b. Bob sees this transaction in transaction pool and sends required number of orders, which will cause the log limit.

  c. Alice order fails

PROPOSED SOLUTION

It is advised to set a limit on the number of orders that can be matched.

REMEDIATION - FIXED

The Spin team has fixed the issue and mentioned that:

On the spot there is a problem with the size of the logs, yes. However there shouldn’t be any problems on the perp, since we use some optimization mechanisms (archiving, using an separated account map, ...). In theory, orders can still get stuck in the order book, but in practice the following mechanism is used to mitigate this:

1. Parameter **max_match_count** - due to which a match of orders will occur for a certain number of orders, and the rest of the order will be canceled
2. **min/max_base/quote_quantity** parameters ensure minimum costs for creating a huge number of orders in the orderbook
3. There is also an admin method for force cancellation of ALL ORDERS one by one - this is just a last resort

GUV-10 Vault can be set to incorrect values via update_settings - Medium

It was observed that the *update_settings* function lacks provided parameter checks, which are present in the create function. Thus, it is possible to update the vault with incorrect values such as **min_deposit > max_deposit** and so on.

PROPOSED SOLUTION

It is recommended to incorporate parameter validation into update_settings from the create function to ensure that the vault is always operating with sane parameters

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-11 Vault Min Capacity Is Not Asserted - Medium

During vault creation, we can specify the minimum capacity of the vault. However, it is not asserted anywhere in the code. Currently, it is possible to forward an epoch even if the minimum capacity is not met.

PROPOSED SOLUTION

Consider asserting whether the vault reached its minimum capacity before letting anyone forward epoch

REMEDIATION - FIXED

The Spin team has fixed the issue in the next version of the vault.

GUV-12 Management Fee Is Not Asserted - Low

It was observed that the **assert_valid** method of the VaultFee contains duplicate validation logic. Based on the logic, it is possible the management fee is supposed to be asserted there

vault:assert_valid:components/vault/src/fees.rs

```
fn assert_valid(&self) {
if let Some(performance_fee_numerator) = self.performance_fee_numerator {

        assert!(self.fee_denominator > performance_fee_numerator);

    }

    if let Some(performance_fee_numerator) = self.performance_fee_numerator {

        assert!(self.fee_denominator > performance_fee_numerator);

    }

}
```

PROPOSED SOLUTION

It was advised to assert the management_fee in one of the conditional blocks.

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-13 Token id Collision Leads To Two Different Vaults Using The Same OFT - Low

It was observed that it is possible for two different vaults to use the same OFT due to the token_id collision.

In the forward_epoch function, self.ofts.create is called to create a new oft per vault. Internally, it uses construct_token_id method, which calculates the token_id as follows:

vault:construct_token_id:components/oft/src/oft/mod.rs

```
    pub fn construct_token_id(
        metadata: &FungibleTokenMetadata,
        option_data: &OptionData,
        authority: &AccountId,
    ) -> String {
        format!(
            "o{}-{:?}-{}-{}-{}",
            metadata.symbol,
            option_data.option_type,
            cast_integer_to_float(option_data.strike_price, PRICE_FEED_DECIMALS as
usize),
            option_data.execution_ts,
            authority,
        )
    }
```

Token id collision can happen if two vaults have the same:

- underlying asset
- strike price
- option type
- option expiration time
- auction start

Then, a malicious actor can initiate a batch transaction to synchronize env::block_timestamp to cause option_expiration_ts, which in return gets assigned as option_data.execution_ts, to be the same for both vaults. This will lead to the token_id collision for both vaults.

option_data.execution_ts is calculated in calculate_params through option_expiration_ts and depends on the start_of_epoch, which depends on the current block timestamp.

vault:calculate_params:option-vault/src/strategy.rs (Part 1)

```
impl VaultInvariant {
    pub fn calculate_params(
        &self,
        pricing: &PricingParams,
        time: &TimeIntervalParams,
        current_ts: u64,
    ) -> BootstrappingParams {
        let mut start_of_epoch = current_ts / time.period * time.period +
time.period_offset;
        if start_of_epoch + time.auction_start < current_ts {
            start_of_epoch += time.period;
        }
        let auction_start_ts = start_of_epoch + time.auction_start;
        let auction_end_ts = auction_start_ts + time.auction_duration;
```

vault:calculate_params:option-vault/src/strategy.rs

```
            BootstrappingParams {
                epoch_start_ts: start_of_epoch,
                epoch_end_ts: start_of_epoch + time.period,
                auction_start_ts,
                auction_end_ts,
              option_expiration_ts: start_of_epoch + time.option_expiration,
                auction_min_price: pricing.auction_min_price,
                strike_price: pricing.strike_price,
                option_type: self.option_type.clone(),
                execution_asset: self.execution_asset.clone(),
                underlying_asset: self.underlying_asset.clone(),
                price_feed_id: pricing.price_feed_id,
            }
        }
    }
```

Then it gets assigned to OptionData:

vault:forward_epoch:option-vault/src/methods/state_mashine.rs

```
    VaultStateMashine::NotStarted => {
        let quantity = vault.current_epoch().start_amount;
        let params = settings.invariant.calculate_params(
            &settings.pricing,
            &settings.time,
            current_ts,
        );
```

vault:forward_epoch:option-vault/src/methods/state_mashine.rs

```
let option_data = OptionData {
    option_type: params.option_type.clone(),
    execution_asset: params.execution_asset.clone(),
    underlying_asset: params.underlying_asset.clone(),
    strike_price: params.strike_price,
  execution_ts: params.option_expiration_ts,
};
```

vault:forward_epoch:option-vault/src/methods/state_mashine.rs

```
let oft_token_id = self.ofts.create(
    vault_address,
    storage_counter,
    underlying_asset_metadata.clone(),
  option_data,
    params.price_feed_id,
    execution_decimals,
);
```

Scenario

1. ALICE created two vaults (1, 2) in the beginning with the same underlying asset but different execution asset
2. Same option type and option expiration time
3. When auction time has approached BOB creates a batch transaction to call forward_epoch with vault_id as 1 and vault_id as 2
4. Calling via batch transaction will make env::block_timestamp to be the same for 2 calls
5. Epoch is forwarded
6. First call creates oft
7. Second call returns previous oft token id since there is a check to prevent creation of new oft if it already exists
8. Both vaults will get the same oft token_id even though the execution assets and other parameters are different

POC: **guv_13.rs**

PROPOSED SOLUTION

If we truly want to have a different oft per vault, we should also include vault_id into the token_id generation

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue by saying that even if provided scenario is used and identical OFTs were created, it won't cause any problems.

GUV-14 Metadata Is Not Validated During Vault Creation - Low

It was observed that neither vault_metadata nor asset metadata is validated despite FungibleTokenMetadata having the **assert_valid** method.

vault:assert_valid:near_contract_standards

```
impl FungibleTokenMetadata {
    pub fn assert_valid(&self) {
        require!(self.spec == FT_METADATA_SPEC);
        require!(self.reference.is_some() == self.reference_hash.is_some());
        if let Some(reference_hash) = &self.reference_hash {
            require!(reference_hash.0.len() == 32, "Hash has to be 32 bytes");
        }
    }
}
```

vault:create_ft_metadata_callback:option-vault/src/bindgen.rs

```
    #[private]
    pub fn create_ft_metadata_callback(
        &mut self,
        #[callback_result] execution_asset_metadata:
Result<FungibleTokenMetadata, PromiseError>,
        #[callback_result] underlying_asset_metadata:
Result<FungibleTokenMetadata, PromiseError>,
        caller: &AccountId,
        settings: VaultSettings,
        capacity: VaultCapacity,
        fee: VaultFeeParams,
        vault_metadata: FungibleTokenMetadata,
    ...
```

PROPOSED SOLUTION

Consider utilizing the assert_valid method to validate passed metadata before processing

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue by saying that they had experienced issues with validation of one of the production tokens, so they decided not to revalidate it and there should be no issues.

GUV-15 Storage Withdraw Is Not Protected With 2fa - Low

It was observed that the **storage_withdraw** does not have 2fa protection through asserting one yocto. This is mentioned in the near-contract-standards: https://github.com/near/near-sdk-rs/blob/master/near-contract-standards/src/storage_management/mod.rs#L97.

PROPOSED SOLUTION

In general, It is advised for all privileged functions, which deal with user balances or crucial contract states, to have assert_one_yocto to prevent malicious calls if a function call key (commonly used on frontends) is compromised.

Alternatively, all the function call keys generated should specify exactly, which methods are allowed to be called so that the most crucial ones should be called only via full key

REMEDIATION - FIXED

The Spin team has fixed the issue by disabling storage withdraw via **unimplemented!()**.

GUV-16 Better Contract Upgrade Mechanism Can Be Implemented - Informational

Currently state is read via two functions with a callback

Current Implementation

```
/// Used to change the state of the contract when updating the contract.
pub fn set_root_state(&mut self, state: String) {
    let signer = env::signer_account_id();
    self.marketplace.ensure_root(signer.as_str());
    self.marketplace.ensure_markets_are_stopped();
    Promise::new(current_account_id()).function_call(
        "set_root_state_callback".to_string(),
        json!({ "state": state }).to_string().into_bytes(),
        0,
        SINGLE_CALL_GAS,
    );
}

#[private]
pub fn set_root_state_callback(state: String) {
    env::storage_write(b"STATE", &base64::decode(state).expect("Invalid state"));
}
```

This pattern is error-prone since we directly write the state to the contract. There are no guarantees that the state you written is correct. The better approach to avoid the "Cannot deserialize the contract state." error on data structure changes is explicitly having an update and migration mechanisms inside the contract. So we know exactly what was an old and new data structure and whether we need to migrate user data

PROPOSED SOLUTION

It is better to include:

- Update Contract Function (for self upgrade)
- Migrate function

Examples (taken from public sources)

Update Contract Function Example

```
pub fn update_contract(&self) -> Promise {
    // Check the caller is authorized to update the code
    assert!(
        env::predecessor_account_id() == self.manager,
        "Only the manager can update the code"
    );
    // Receive the code directly from the input to avoid the
    // GAS overhead of deserializing parameters
    let code = env::input().expect("Error: No input").to_vec();

    // Deploy the contract on self
    Promise::new(env::current_account_id())
        .deploy_contract(code)
        .function_call(
            "migrate".to_string(),
            NO_ARGS,
            NearToken::from_near(0),
            CALL_GAS,
        ).as_return()}
```

Migrate Function Example 1

```
    #[near_bindgen]
    #[derive(BorshDeserialize, BorshSerialize, PanicOnDefault)]
    pub struct OldState {
        token_account_id: AccountId,
        oracles: UnorderedSet<AccountId>,
        claim_period: Duration,
        burn_period: Duration,
        accruals: UnorderedMap<UnixTimestamp, (Vector<TokensAmount>,
TokensAmount)>,
        accounts: LookupMap<AccountId, AccountRecordLegacy>,
        is_service_call_running: bool,
    }

    #[near_bindgen]
    impl Contract {
        #[private]
        #[init(ignore_state)]
        pub fn migrate() -> Self {
            let old_state: OldState = env::state_read().expect("Failed to read old state");

            Self {
                token_account_id: old_state.token_account_id,
                oracles: old_state.oracles,
                claim_period: old_state.claim_period,
                burn_period: old_state.burn_period,
                accruals: old_state.accruals,
                accounts_legacy: old_state.accounts,
                accounts: LookupMap::new(StorageKey::Accounts),
                is_service_call_running: old_state.is_service_call_running,
                balance_to_burn: 0,
            }
        }
```

Migrate Function Example 2 (Part 1: Old State Structures)

Old State Structures

```
#[near(serializers=[borsh])]
pub struct OldPostedMessage {
   pub premium: bool,
   pub sender: AccountId,
   pub text: String,
}

#[near(serializers=[borsh])]
pub struct OldState {
   messages: Vector<OldPostedMessage>,
   payments: Vector<NearToken>,
}
```

Migrate Function Example 2 (Part 2: Migrate Function)

Migrate Function

```
#[near]
impl GuestBook {
    #[private]
    #[init(ignore_state)]
    pub fn migrate() -> Self {
        // retrieve the current state from the contract
        let old_state: OldState = env::state_read().expect("failed");

        // iterate through the state migrating it to the new version
        let mut new_messages: Vector<PostedMessage> = Vector::new(b"p");

        for (idx, posted) in old_state.messages.iter().enumerate() {
            let payment = old_state
                .payments
                .get(idx as u64)
                .unwrap_or(NearToken::from_near(0));

            new_messages.push(&PostedMessage {
                payment,
                premium: posted.premium,
                sender: posted.sender,
                text: posted.text,
            })
        }

        // return the new state
        Self {
            messages: new_messages,
        }
    }
}
```

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue by saying: "For the perp, the set_root_state method no longer exists; for the spot it remains as a legacy method, but can be removed. We can add a method migrate in case of emergency and necessity."

GUV-17 Casting overflow via as_decimals() when the account cannot be liquidated - Informational

It was observed that the check_liquidation as_decimals() is used to format the margin_ratio from Fractional to the Decimal. Usage of as_decimals causes casting overflow in some scenarios since when performing division if numerator to denominator we first multiply the numerator by 10**24

perp:check_liquidation:perp/contract/src/risk/liquidation.rs

```
    fn check_liquidation(&self, positions_collateral: Decimal) -> Result<bool,
LiquidationError> {
        let margin_ratio = MarginInfo::from_parts(self.total_margin, positions_collateral);

        if !margin_ratio.can_be_liquidated() {
            Err(LiquidationError::CanNotBeLiquidated(
                margin_ratio.ratio().as_decimal(),
            ))
        } else {
            Ok(margin_ratio.can_completely_liquidated())
        }
    }
```

This can lead to situations when the transaction failure cannot be correctly interpreted, especially if some offchain components rely on the correct error message returned.

PROPOSED SOLUTION

It is advised to add additional checks to ensure that the value cannot be larger than u128::MAX

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue by saying: "As we understand, the problem should not arise on real values, taking into account the restrictions min/max_base/quote_quantity but we will check separately and probably make some threshold, starting from which the margin ratio will be considered infinity / NaN"

GUV-18 .view() method leads to casting overflow in get_positions due to as_decimals usage in Fraction Display implementation - Informational

It was observed that when **get_positions()** is called, **.view()** method is invoked on margin_ration, which internally uses Display implementation for **Fraction**. Since Display implementation for Fraction uses **as_decimals()**, it sometimes leads to casting overflow on division

perp:get_positions:perp/contract/src/perp/position.rs

```
pub fn get_positions(&self, account_id: AccountId) -> PositionsView {
    let accounts = self.accounts_holder.instance();
    let acc = accounts.get(&account_id);
   let positions = acc.get_positions().view(&self.markets);
    let margin_ratio = self.margin_ratio(&acc).view();
    PositionsView {
        margin_ratio,
        positions,
    }
}
```

perp:fmt:core/types/src/number/fraction.rs

```
impl Display for Fraction {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if self.is_nan() {
            write!(f, "NaN")
        } else {
         write!(f, "{}", self.as_decimal())
        }
    }
}
```

perp:div:core/types/src/number/decimal.rs

```
impl Div<Self> for Decimal {
    type Output = Decimal;

    fn div(self, rhs: Self) -> Self::Output {
        let self_scaled_with_overflow = U256::from(self.val) * U256::from(DECIMAL);
        let division = self_scaled_with_overflow / U256::from(rhs.val);
        Self {
         val: division.as_u128(),
        }
    }
}
```

POC: **guv_19.rs**

PROPOSED SOLUTION

It is advised to add additional checks to ensure that the value cannot be larger than u128::MAX

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue by saying: "As we understand, the problem should not arise on real values, taking into account the restrictions min/max_base/quote_quantity but we will check separately and probably make some threshold, starting from which the margin ratio will be considered infinity / NaN"

GUV-19 Suboptimal Assertion - Informational

When you have > 0 and operate within unsigned integer space, a more logical check is != 0. By default, unsigned integer space checks that the value is >= 0. So the only thing you need to check is val != 0 since the value cannot be smaller than 0

deposit

```
pub fn deposit(
    &mut self,
    account_id: &AccountId,
    token: u16,
    amount: Balance,
    event: BalanceChangeInfo,
) -> Balance {
    assert!(
        amount > U128(0),
        "Expected positive amount. User: {}, token: {}, amount: {}",
        account_id,
        token,
        amount.0
    );
```

PROPOSED SOLUTION

It is advised to change > operator to != 0

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-20 Redundant Prepaid Gas Check - Informational

It was observed that there is a redundant prepaid gas check in the function

withdraw function

```
#[payable]
pub fn withdraw(&mut self, token: AccountId, amount: Balance) -> Promise {
   let _session = self.make_session();
   let signer = env::signer_account_id();
   assert!(
      env::prepaid_gas() >= SINGLE_CALL_GAS * 5,
      "Expected prepaid gas {}.",
      SINGLE_CALL_GAS.0 * 5
   );
   self.marketplace.withdraw(
      signer.to_string(),
      token.to_string(),
      amount,
   );
   assert!(
      env::prepaid_gas() >= SINGLE_CALL_GAS * 5,
      "Expected prepaid gas {}.",
      SINGLE_CALL_GAS.0 * 5
   );
```

PROPOSED SOLUTION

It is advised to only keep one

REMEDIATION - FIXED

The Spin team has fixed the issue in the next version of the contract"

GUV-21 Contract Storage Can Be Optimized - Informational

We can decrease the total size of the spot and other contracts by following these steps:

Steps Performed

Get initial Size

We got the total size of the binary by using twiggy.

Initial Size Analysis

```
 Shallow Bytes │ Shallow % │ Item
───────────────┼───────────┼──────────────────────────────────────
        610946 ┊    29.03% ┊ custom section '.debug_str'
        379733 ┊    18.04% ┊ custom section '.debug_info'
        240083 ┊    11.41% ┊ custom section '.debug_line'
        197639 ┊     9.39% ┊ "function names" subsection
        164224 ┊     7.80% ┊ custom section '.debug_ranges'
         60562 ┊     2.88% ┊ data[0]
          6298 ┊     0.30% ┊
core::num::flt2dec::strategy::dragon::format_shortest::h5d521f4a27ed81b4
          5663 ┊     0.27% ┊ base64::decode::decode_config_buf::h6817914728a71c2f
          5318 ┊     0.25% ┊
core::num::flt2dec::strategy::dragon::format_exact::h9b3f09aaeac78ab3
          4449 ┊     0.21% ┊ custom section '.debug_abbrev'
        406891 ┊    19.33% ┊ ... and 2245 more.
       2104765 ┊   100.00% ┊ Σ [2265 Total Rows]
```

Total Size: 2104765 bytes = 2104.765kb

Cost to store: ~21.047649999999997N (100kb = ~1N) = $132.60019499999999 (at 6.30 price)

Stripping Debug Info

If we do not include these flags in .cargo/config.toml:

Cargo Config

```
[target.wasm32-unknown-unknown]
rustflags = ["-C", "link-arg=-s"]
```

Or pass them directly during cargo build, our generated WASM binary will contain debug info. We can decrease binary size by stripping them:

Debug Info Size

```
610946 ┊  29.03% ┊ custom section '.debug_str'
379733 ┊  18.04% ┊ custom section '.debug_info'
240083 ┊  11.41% ┊ custom section '.debug_line'
197639 ┊   9.39% ┊ "function names" subsection
```

Removing rlib

We noticed that the crate type is set as both cdylib and rlib. However, since NEAR smart contracts are compiled to WASM, rlib is unnecessary. Eliminating rlib can significantly reduce the size of the generated WASM binary.

Cargo.toml

```toml
[lib]
crate-type = ["cdylib"]
```

Optimizing Via Wasm-Opt

We used wasm-opt to further optimize generated WASM binary by performing several iterations of optimization:

1. wasm-opt -Oz -o swap_optimized.wasm ./target/wasm32-unknown-unknown/release/swap.wasm
2. wasm-opt -Oz -o swap_optimized.wasm swap_optimized.wasm

Diff To See Result

We ran twiggy-diff between old binary and optimized one:

Optimization Diff

```
 Delta Bytes │ Item
─────────────┼──────────────────────────────────────────────────────
 -610946 ┊ custom section '.debug_str'
 -379733 ┊ custom section '.debug_info'
 -240083 ┊ custom section '.debug_line'
 -164224 ┊ custom section '.debug_ranges'
  -59852 ┊ data[0]
  +19939 ┊ data[4]
   +9668 ┊ code[199]
   +7026 ┊ data[1]
   +6664 ┊ code[662]
   +6379 ┊ data[28]
   -6298 ┊ core::num::flt2dec::strategy::dragon::format_shortest::h5d521f4a27ed81b4
   -5663 ┊ base64::decode::decode_config_buf::hfb20bcc6885b3c7c
   +5577 ┊ code[452]
   +5466 ┊ data[5]
   -5318 ┊ core::num::flt2dec::strategy::dragon::format_exact::h9b3f09aaeac78ab3
   +4986 ┊ code[576]
   +4825 ┊ code[121]
   +4742 ┊ code[644]
   +4473 ┊ data[2]
   -4449 ┊ custom section '.debug_abbrev'
 -151797 ┊ ... and 3032 more.
-1754694 ┊ Σ [3052 Total Rows]
```

Result

Here are details of a new binary size after various optimizations without changing the actual code:

- Total Saved: 1754694 bytes = 1754.694 kb
- Saved in percentage: 83.36769188009113%
- Total Cost Saved: ~17.54694N = ~$110.545722 (6.30 price)
- Current Size: 350071 bytes = 350.071kb
- Current Cost: 3.50071N = ~$22.054473 (6.30 price)

PROPOSED SOLUTION

It is advised to follow the stepts above to decrease the binary size.

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.

GUV-22 Not Resolved TODOs In The Codebase - Informational

It was observed that the codebase contains several TODOs that are not resolved. Some of them describe certain vulnerabilities and suboptimal mathematical operations.

vault:mul_div:types/num/src/uint256.rs

```
    //TODO: remove this muldiv and change everywhere to be muldiv256 with
chaning it's name to this
    pub fn mul_div<A: Into<U256>, B: Into<U256>, C: Into<U256>>(a: A, b: B, c: C) ->
Option<u128> {
        let a = a.into();
        let b = b.into();
        let c = c.into();

        if c == U256::from(0) {
            Some(0)
        } else {
            a.checked_mul(b)
                .and_then(|v| v.checked_div(c))
                .map(TryInto::try_into)
                .transpose()
                .ok()
                .flatten()
        }
    }
```

vault:mft_resolve_transfer:core/mft/src/lib.rs

```
    #[private]
    fn mft_resolve_transfer(
        &mut self,
        token_id: String,
        sender_id: AccountId,
        receiver_id: &AccountId,
        amount: U128,
    ) -> U128 {
        let unused_amount = match env::promise_result(0) {
            PromiseResult::NotReady => unreachable!(),
            PromiseResult::Successful(value) => {
                if let Ok(unused_amount) = near_sdk::serde_json::from_slice::<U128>(&value) {
                    std::cmp::min(amount.0, unused_amount.0)
                } else {
                    amount.0
                }
            }
            PromiseResult::Failed => amount.0,
        };

        if unused_amount > 0 {
          // TODO: vulnerability with account deleting would be available,
          // there should be a check if sender account was not deleted before resolve
            let receiver_balance = self.internal_balance_of(&token_id, receiver_id);
            if receiver_balance > 0 {
                let refund_amount = std::cmp::min(receiver_balance, unused_amount);
                self.internal_mft_transfer(token_id, &sender_id, receiver_id, refund_amount,
None);
            }
        }

        unused_amount.into()
    }
```

PROPOSED SOLUTION

It is advised to resolve these TODOs.

REMEDIATION - ACKNOWLEDGED

The Spin team has acknowledged the issue.