Solidity Scan | Powered by CRED SHIELDS

Security Assessment

Test67

24 May 2023

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Table of Contents.

- FUNCTION SHOULD BE EXTERNAL
- USE OF TX.GASPRICE
- VARIABLES SHOULD BE IMMUTABLE
- WEAK PRNG

Scan History

Disclaimer

Project Summary

This report has been prepared for Test67 using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (130+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after Test67 introduces new features or refactors the code.

Audit Summary

Project Name

Test67

Contract Type

Smart Contract

Language

Solidity

Codebase

https://github.com/satyamkjha/qualification

Date Published

27 Jun 2023

Publishers/Owners Name

klashjklasdasd

Audit Methodology

Static Scanning

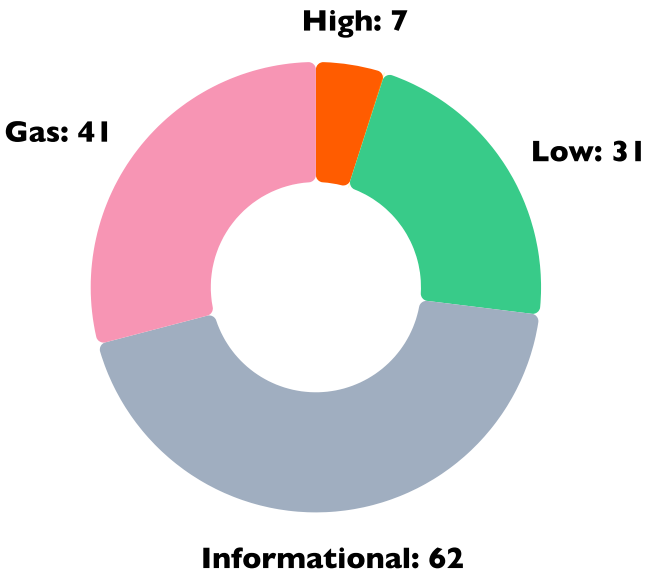
Findings Summary

Test67

https://github.com/satyamkjha/qualification

Lines of Code **515**

70.87
Score



Critical	0	Low	31
High	7	Informational	62
Medium	0	Gas	41

ACTION TAKEN

Fixed	False Positive
24	17
Won't Fix	**Pending Fix**
5	108

Bug ID	Severity	Bug Type	Status
SSP_1324_91	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_91	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_122	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_122	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_114	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_114	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix

SSP_1324_123	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_123	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSP_1324_88	Gas	ARRAY LENGTH CACHING	Pending Fix
SSP_1324_89	Gas	ARRAY LENGTH CACHING	Pending Fix
SSP_1324_98	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_99	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_104	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_105	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_102	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_103	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_100	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_101	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix

SSP_1324_107	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_108	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSP_1324_106	Informational	BLOCK VALUES AS A PROXY FOR TIME	Fixed
SSP_1324_10	Informational	BOOLEAN EQUALITY	Pending Fix
SSP_1324_11	Informational	BOOLEAN EQUALITY	Pending Fix
SSP_1324_90	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_90	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_112	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_118	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_119	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_116	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_116	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix

SSP_1324_117	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_113	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_113	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_115	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_120	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_120	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_121	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSP_1324_12	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSP_1324_45	Gas	CUSTOM ERRORS TO SAVE GAS	Pending Fix
SSP_1324_46	Gas	CUSTOM ERRORS TO SAVE GAS	Pending Fix
SSP_1324_47	Gas	CUSTOM ERRORS TO SAVE GAS	Pending Fix
SSP_1324_124	Low	EVENT BASED REENTRANCY	Pending Fix

SSP_1324_127	Low	EVENT BASED REENTRANCY	Pending Fix
SSP_1324_126	Low	EVENT BASED REENTRANCY	Pending Fix
SSP_1324_125	Low	EVENT BASED REENTRANCY	Pending Fix
SSP_1324_128	Low	EVENT BASED REENTRANCY	Pending Fix
SSP_1324_1	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSP_1324_4	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSP_1324_3	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSP_1324_2	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSP_1324_5	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSP_1324_40	Low	USE OF FLOATING PRAGMA	Won't Fix
SSP_1324_43	Low	USE OF FLOATING PRAGMA	Won't Fix
SSP_1324_42	Low	USE OF FLOATING PRAGMA	Won't Fix

SSP_1324_39	Low	USE OF FLOATING PRAGMA	False Positive
SSP_1324_38	Low	USE OF FLOATING PRAGMA	Won't Fix
SSP_1324_41	Low	USE OF FLOATING PRAGMA	False Positive
SSP_1324_44	Low	USE OF FLOATING PRAGMA	Won't Fix
SSP_1324_63	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_66	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_67	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_68	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_65	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_64	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_69	Informational	FUNCTION CALLDATA OPTIMIZATION	Pending Fix
SSP_1324_6	High	UNCHECKED ARRAY LENGTH	False Positive

SSP_1324_7	High	UNCHECKED ARRAY LENGTH	False Positive
SSP_1324_8	Gas	GAS OPTIMIZATION IN INCREMENTS	Pending Fix
SSP_1324_9	Gas	GAS OPTIMIZATION IN INCREMENTS	Pending Fix
SSP_1324_70	Low	MISSING EVENTS	False Positive
SSP_1324_73	Low	MISSING EVENTS	False Positive
SSP_1324_74	Low	MISSING EVENTS	False Positive
SSP_1324_75	Low	MISSING EVENTS	False Positive
SSP_1324_76	Low	MISSING EVENTS	False Positive
SSP_1324_77	Low	MISSING EVENTS	False Positive
SSP_1324_78	Low	MISSING EVENTS	False Positive
SSP_1324_79	Low	MISSING EVENTS	False Positive
SSP_1324_72	Low	MISSING EVENTS	False Positive

SSP_1324_71	Low	MISSING EVENTS	False Positive
SSP_1324_80	Low	MISSING EVENTS	False Positive
SSP_1324_109	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSP_1324_136	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Fixed
SSP_1324_110	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Fixed
SSP_1324_111	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Fixed
SSP_1324_22	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_23	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_24	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_31	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_32	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_33	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix

SSP_1324_34	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_28	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_29	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_30	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_25	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_26	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_27	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_35	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_36	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_37	Informational	MISSING STATE VARIABLE VISIBILITY	Pending Fix
SSP_1324_83	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_86	Low	OUTDATED COMPILER VERSION	Pending Fix

SSP_1324_85	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_82	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_81	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_84	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_87	Low	OUTDATED COMPILER VERSION	Pending Fix
SSP_1324_129	High	REENTRANCY	False Positive
SSP_1324_132	High	REENTRANCY	Pending Fix
SSP_1324_131	High	REENTRANCY	Pending Fix
SSP_1324_130	High	REENTRANCY	False Positive
SSP_1324_133	High	REENTRANCY	Pending Fix
SSP_1324_92	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix
SSP_1324_95	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix

SSP_1324_96	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix
SSP_1324_94	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix
SSP_1324_93	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix
SSP_1324_97	Gas	SUPERFLUOUS EVENT FIELDS	Pending Fix
SSP_1324_15	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_14	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_21	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_20	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_19	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_17	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_16	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSP_1324_18	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix

SSP_1324_135	Gas	FUNCTION SHOULD BE EXTERNAL	Fixed
SSP_1324_134	Gas	FUNCTION SHOULD BE EXTERNAL	Fixed
SSP_1324_137	Gas	FUNCTION SHOULD BE EXTERNAL	Fixed
SSP_1324_138	Gas	FUNCTION SHOULD BE EXTERNAL	Fixed
SSP_1324_139	High	USE OF TX.GASPRICE	Pending Fix
SSP_1324_140	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_141	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_142	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_143	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_144	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_145	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_146	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix

SSP_1324_147	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_148	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_149	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_150	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_151	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_152	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_153	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_154	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSP_1324_48	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_49	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_50	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_60	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed

SSP_1324_61	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_62	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_54	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_55	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_56	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_51	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_52	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_53	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_57	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_58	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_59	Informational	VARIABLES SHOULD BE IMMUTABLE	Fixed
SSP_1324_13	Low	WEAK PRNG	Fixed

Vulnerability Details

Bug ID

SSP_1324_91

Severity

Informational

Confidence

Tentative

Line nos

53-53

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x387C1417597eFd39fb61003E1e798b218eA5Be3B`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.

Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_91

Severity

Informational

Confidence

Tentative

Line nos

60-60

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x387C1417597eFd39fb61003E1e798b218eA5Be3B`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_122

Severity

Informational

Confidence

Tentative

Line nos

53-53

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x42aca25Fd7Be774225abfbE4275beb9BF59c832f`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_122

Severity

Informational

Confidence

Tentative

Line nos

60-60

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x42aca25Fd7Be774225abfbE4275beb9BF59c832f`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_114

Severity

Informational

Confidence

Tentative

Line nos

53-53

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x387C1417597eFd39fb61003E1e798b218eA5Be3B`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_114

Severity

Informational

Confidence

Tentative

Line nos

60-60

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x387C1417597eFd39fb61003E1e798b218eA5Be3B`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_123

Severity

Informational

Confidence

Tentative

Line nos

53-53

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x42aca25Fd7Be774225abfbE4275beb9BF59c832f`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_123

Severity

Informational

Confidence

Tentative

Line nos

60-60

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x42aca25Fd7Be774225abfbE4275beb9BF59c832f`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSP_1324_88

Severity

Gas

Confidence

Certain

Line nos

89-91

Action Taken

Pending Fix

Bug Type

ARRAY LENGTH CACHING

File Location

/contracts/qualification_luckydraw.sol

Issue Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.

Issue Remediation

Consider storing the array length of the variable before the loop and use the stored length instead of fetching it in each iteration.

Bug ID

SSP_1324_89

Severity

Gas

Confidence

Certain

Line nos

95-97

Action Taken

Pending Fix

Bug Type

ARRAY LENGTH CACHING

File Location

/contracts/qualification_luckydraw.sol

Issue Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.

Issue Remediation

Consider storing the array length of the variable before the loop and use the stored length instead of fetching it in each iteration.

Bug ID

SSP_1324_98

Severity

Informational

Confidence

Firm

Line nos

31-31

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_99

Severity

Informational

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_104

Severity

Informational

Confidence

Firm

Line nos

47-47

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_105

Severity

Informational

Confidence

Firm

Line nos

112-112

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_102

Severity

Informational

Confidence

Firm

Line nos

31-31

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_103

Severity

Informational

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_100

Severity

Informational

Confidence

Firm

Line nos

31-31

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_101

Severity

Informational

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_107

Severity

Informational

Confidence

Firm

Line nos

31-31

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_108

Severity

Informational

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_106

Severity

Informational

Confidence

Firm

Line nos

138-138

Action Taken

Fixed

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_1324_10

Severity

Informational

Confidence

Certain

Line nos

116-116

Action Taken

Pending Fix

Bug Type

BOOLEAN EQUALITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Issue Remediation

It is recommended to directly use boolean constants. It is not required to equate them to `true` or `false.`

Bug ID

SSP_1324_11

Severity

Informational

Confidence

Certain

Line nos

117-117

Action Taken

Pending Fix

Bug Type

BOOLEAN EQUALITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Issue Remediation

It is recommended to directly use boolean constants. It is not required to equate them to `true` or `false.`

Bug ID

SSP_1324_90

Severity

Gas

Confidence

Firm

Line nos

53-53

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_90

Severity

Gas

Confidence

Firm

Line nos

60-60

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_112

Severity

Gas

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_118

Severity

Gas

Confidence

Firm

Line nos

105-105

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_119

Severity

Gas

Confidence

Firm

Line nos

112-112

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_116

Severity

Gas

Confidence

Firm

Line nos

53-53

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_116

Severity

Gas

Confidence

Firm

Line nos

60-60

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_117

Severity

Gas

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_113

Severity

Gas

Confidence

Firm

Line nos

53-53

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_113

Severity

Gas

Confidence

Firm

Line nos

60-60

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_115

Severity

Gas

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_120

Severity

Gas

Confidence

Firm

Line nos

53-53

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_120

Severity

Gas

Confidence

Firm

Line nos

60-60

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_121

Severity

Gas

Confidence

Firm

Line nos

63-63

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_12

Severity

- **Gas**

Confidence

Firm

Line nos

110-110

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be performing comparisons using inequalities inside the `require` statement. When inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_1324_45

Severity

Gas

Confidence

Certain

Line nos

107-107

Action Taken

Pending Fix

Bug Type

CUSTOM ERRORS TO SAVE GAS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be using `revert()` statements. Since Solidity `v0.8.4`, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using `revert`.

Issue Remediation

It is recommended to replace all the instances of `revert()` statements with `error()` to save gas.

Bug ID

SSP_1324_46

Severity

Gas

Confidence

Certain

Line nos

114-114

Action Taken

Pending Fix

Bug Type

CUSTOM ERRORS TO SAVE GAS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be using `revert()` statements. Since Solidity `v0.8.4`, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using `revert`.

Issue Remediation

It is recommended to replace all the instances of `revert()` statements with `error()` to save gas.

Bug ID

SSP_1324_47

Severity

Gas

Confidence

Certain

Line nos

120-120

Action Taken

Pending Fix

Bug Type

CUSTOM ERRORS TO SAVE GAS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The contract was found to be using `revert()` statements. Since Solidity `v0.8.4`, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using `revert`.

Issue Remediation

It is recommended to replace all the instances of `revert()` statements with `error()` to save gas.

Bug ID

SSP_1324_124

Severity

Low

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_1324_127

Severity

Low

Confidence

Firm

Line nos

104-124

Action Taken

Pending Fix

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_1324_126

Severity

Low

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_1324_125

Severity

Low

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_1324_128

Severity

Low

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_1324_1

Severity

Gas

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['account'] inside event Qualification.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events.
If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSP_1324_4

Severity

Gas

Confidence

Firm

Line nos

104-124

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['account'] inside event Qualification.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events.
If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSP_1324_3

Severity

Gas

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['account'] inside event Qualification.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events.
If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSP_1324_2

Severity

Gas

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['account'] inside event Qualification.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events.
If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSP_1324_5

Severity

Gas

Confidence

Firm

Line nos

59-72

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['account'] inside event Qualification.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events.
If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSP_1324_40

Severity

• **Low**

Confidence

Tentative

Line nos

9-9

Action Taken

Won't Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/qualification_mask_history_position_500_ropsten.sol'] - >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Comments

poiuojulko

Bug ID

SSP_1324_43

Severity

Low

Confidence

Tentative

Line nos

9-9

Action Taken

Won't Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/qualification_luckydraw.sol'] - >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Comments

poiuojulko

Bug ID

SSP_1324_42

Severity

Low

Confidence

Tentative

Line nos

9-9

Action Taken

Won't Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/qualification_mask_history_position_1000_mainnet.sol']
- >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Comments

poiuojulko

Bug ID

SSP_1324_39

Severity

Low

Confidence

Tentative

Line nos

9-9

Action Taken

False Positive

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/IQLF.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/IQLF.sol'] - >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Bug ID

SSP_1324_38

Severity

Low

Confidence

Tentative

Line nos

9-9

Action Taken

Won't Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/IMTS.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/IMTS.sol'] - >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Comments

poiuojulko

Bug ID

SSP_1324_41

Severity

• **Low**

Confidence

Tentative

Line nos

9-9

Action Taken

False Positive

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/qualification_mask_history_position_1000_ropsten.sol']` – `>=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Bug ID

SSP_1324_44

Severity

• **Low**

Confidence

Tentative

Line nos

9-9

Action Taken

Won't Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/qualification_mask_history_position_500_mainnet.sol'] - >=0.8.0`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `>=0.8.0` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Comments

poiuojulko

Bug ID

SSP_1324_63

Severity

Informational

Confidence

Certain

Line nos

29-34

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_66

Severity

Informational

Confidence

Certain

Line nos

40-54

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_luckydraw.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_67

Severity

Informational

Confidence

Certain

Line nos

88-92

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_luckydraw.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_68

Severity

Informational

Confidence

Certain

Line nos

94-98

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_luckydraw.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_65

Severity

Informational

Confidence

Certain

Line nos

29-34

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

---

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_64

Severity

Informational

Confidence

Certain

Line nos

29-34

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_69

Severity

Informational

Confidence

Certain

Line nos

29-34

Action Taken

Pending Fix

Bug Type

FUNCTION CALLDATA OPTIMIZATION

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

NOT DEFINED YET.

Issue Remediation

NOT DEFINED YET.

Bug ID

SSP_1324_6

Severity

High

Confidence

Tentative

Line nos

89-89

Action Taken

False Positive

Bug Type

UNCHECKED ARRAY LENGTH

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If `array.length` is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.

```
for (uint256 i = 0; i < array.length ; i++) { cosltyFunc(); }
```

This becomes a security issue if an external actor influences `array.length`.
E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Issue Remediation

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Bug ID

SSP_1324_7

Severity

High

Confidence

Tentative

Line nos

95-95

Action Taken

False Positive

Bug Type

UNCHECKED ARRAY LENGTH

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If `array.length` is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.

```
for (uint256 i = 0; i < array.length ; i++) { cosltyFunc(); }
```

This becomes a security issue if an external actor influences `array.length`.
E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Issue Remediation

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Bug ID

SSP_1324_8

Severity

- **Gas**

Confidence

Tentative

Line nos

89-89

Action Taken

Pending Fix

Bug Type

GAS OPTIMIZATION IN INCREMENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

`++i` costs less gas compared to `i++` or `i += 1` for unsigned integers. In `i++`, the compiler has to create a temporary variable to store the initial value. This is not the case with `++i` in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Issue Remediation

Consider changing the post-increments `(i++)` to pre-increments `(++i)` as long as the value is not used in any calculations or inside returns. Make sure that the logic of the code is not changed.

Bug ID

SSP_1324_9

Severity

Gas

Confidence

Tentative

Line nos

95-95

Action Taken

Pending Fix

Bug Type

GAS OPTIMIZATION IN INCREMENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

`++i` costs less gas compared to `i++` or `i += 1` for unsigned integers. In `i++`, the compiler has to create a temporary variable to store the initial value. This is not the case with `++i` in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Issue Remediation

Consider changing the post-increments `(i++)` to pre-increments `(++i)` as long as the value is not used in any calculations or inside returns. Make sure that the logic of the code is not changed.

Bug ID

SSP_1324_70

Severity

Low

Confidence

Firm

Line nos

48-50

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_HISTORY_POSITION_500_MASK_ROPSTEN was found to be missing these events on the function set_start_time which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_73

Severity

Low

Confidence

Firm

Line nos

68-70

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function set_start_time which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_74

Severity

Low

Confidence

Firm

Line nos

72-74

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function set_max_gas_price which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_75

Severity

Low

Confidence

Firm

Line nos

76-78

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function set_min_token_amount which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_76

Severity

Low

Confidence

Firm

Line nos

80-82

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function set_lucky_factor which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_77

Severity

Low

Confidence

Firm

Line nos

84-86

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function set_token_addr which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_78

Severity

Low

Confidence

Firm

Line nos

88-92

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function add_whitelist which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_79

Severity

Low

Confidence

Firm

Line nos

94-98

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_LUCKYDRAW was found to be missing these events on the function remove_whitelist which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_72

Severity

Low

Confidence

Firm

Line nos

48-50

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_HISTORY_POSITION_1000_MASK_MAIN was found to be missing these events on the function set_start_time which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_71

Severity

Low

Confidence

Firm

Line nos

48-50

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_HISTORY_POSITION_1000_MASK_ROPSTEN was found to be missing these events on the function set_start_time which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_80

Severity

Low

Confidence

Firm

Line nos

48-50

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract QLF_HISTORY_POSITION_500_MASK_MAIN was found to be missing these events on the function set_start_time which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_1324_109

Severity

Informational

Confidence

Certain

Line nos

37-37

Action Taken

Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/IQLF.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_1324_136

Severity

Informational

Confidence

Certain

Line nos

33-33

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_1324_110

Severity

Informational

Confidence

Certain

Line nos

33-33

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_1324_111

Severity

Informational

Confidence

Certain

Line nos

34-34

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_1324_22

Severity

Informational

Confidence

Tentative

Line nos

20-20

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable start_time that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_23

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable creator that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_24

Severity

Informational

Confidence

Tentative

Line nos

22-22

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable black_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_31

Severity

Informational

Confidence

Tentative

Line nos

18-18

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable start_time that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_32

Severity

Informational

Confidence

Tentative

Line nos

29-29

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable creator that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_33

Severity

Informational

Confidence

Tentative

Line nos

30-30

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable black_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_34

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable whilte_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_28

Severity

Informational

Confidence

Tentative

Line nos

20-20

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable start_time that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_29

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable creator that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_30

Severity

Informational

Confidence

Tentative

Line nos

22-22

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable black_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_25

Severity

Informational

Confidence

Tentative

Line nos

20-20

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable start_time that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_26

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable creator that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_27

Severity

Informational

Confidence

Tentative

Line nos

22-22

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable black_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_35

Severity

Informational

Confidence

Tentative

Line nos

20-20

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable start_time that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_36

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable creator that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_37

Severity

Informational

Confidence

Tentative

Line nos

22-22

Action Taken

Pending Fix

Bug Type

MISSING STATE VARIABLE VISIBILITY

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Visibility modifiers determine the level of access to the variables in your smart contract. This defines the level of access for contracts and other external users. It makes it easier to understand who can access the variable.
The contract defined a state variable black_list that was missing a visibility modifier.

Issue Remediation

Explicitly define visibility for all state variables. These variables can be specified as public, internal or private.

Bug ID

SSP_1324_83

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/qualification_mask_history_position_500_ropsten.sol'] -
>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_86

Severity

• **Low**

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/qualification_luckydraw.sol']` - `>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_85

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/qualification_mask_history_position_1000_mainnet.sol']`
- `>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_82

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/IQLF.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/IQLF.sol']` - `>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_81

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/IMTS.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/IMTS.sol']` - `>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_84

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/qualification_mask_history_position_1000_ropsten.sol']`
- `>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_87

Severity

Low

Confidence

Certain

Line nos

9-9

Action Taken

Pending Fix

Bug Type

OUTDATED COMPILER VERSION

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['/contracts/qualification_mask_history_position_500_mainnet.sol'] -
>=0.8.0`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.18`, which patches most solidity vulnerabilities.

Bug ID

SSP_1324_129

Severity

High

Confidence

Certain

Line nos

59-72

Action Taken

False Positive

Bug Type

REENTRANCY

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_1324_132

Severity

High

Confidence

Certain

Line nos

104-124

Action Taken

Pending Fix

Bug Type

REENTRANCY

File Location

/contracts/qualification_luckydraw.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_1324_131

Severity

High

Confidence

Certain

Line nos

59-72

Action Taken

Pending Fix

Bug Type

REENTRANCY

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_1324_130

Severity

High

Confidence

Certain

Line nos

59-72

Action Taken

False Positive

Bug Type

REENTRANCY

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_1324_133

Severity

High

Confidence

Certain

Line nos

59-72

Action Taken

Pending Fix

Bug Type

REENTRANCY

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_1324_92

Severity

Gas

Confidence

Certain

Line nos

70-70

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_95

Severity

Gas

Confidence

Certain

Line nos

119-119

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_96

Severity

Gas

Confidence

Certain

Line nos

122-122

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_luckydraw.sol

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_94

Severity

Gas

Confidence

Certain

Line nos

70-70

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_93

Severity

Gas

Confidence

Certain

Line nos

70-70

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

---

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_97

Severity

Gas

Confidence

Certain

Line nos

70-70

Action Taken

Pending Fix

Bug Type

SUPERFLUOUS EVENT FIELDS

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs extra gas.

Issue Remediation

`block.timestamp` and `block.number` do not need to be added manually. Consider removing them from the emitted events.

Bug ID

SSP_1324_15

Severity

Gas

Confidence

Certain

Line nos

48-50

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_14

Severity

Gas

Confidence

Certain

Line nos

59-72

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_21

Severity

Gas

Confidence

Certain

Line nos

88-92

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_20

Severity

Gas

Confidence

Certain

Line nos

94-98

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_19

Severity

Gas

Confidence

Certain

Line nos

84-86

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_17

Severity

Gas

Confidence

Certain

Line nos

80-82

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_16

Severity

Gas

Confidence

Certain

Line nos

72-74

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_18

Severity

Gas

Confidence

Certain

Line nos

76-78

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_135

Severity

Gas

Confidence

Certain

Line nos

68-70

Action Taken

Fixed

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_134

Severity

Gas

Confidence

Certain

Line nos

104-124

Action Taken

Fixed

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_luckydraw.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_137

Severity

Gas

Confidence

Certain

Line nos

59-72

Action Taken

Fixed

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_138

Severity

Gas

Confidence

Certain

Line nos

48-50

Action Taken

Fixed

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSP_1324_139

Severity

High

Confidence

Certain

Line nos

105-105

Action Taken

Pending Fix

Bug Type

USE OF TX.GASPRICE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

The `tx.gasprice` should always be set by users of the contract and not by the developers.
The contract QLF_LUCKYDRAW was found to be using `tx.gasprice` on line 105.

Issue Remediation

Suggesting gas using `tx.gasprice` could lead to exploits depending on the business logic of the code. (Eg: The Ethereum Alarm Clock Gas Refund Exploit)

Bug ID

SSP_1324_140

Severity

Informational

Confidence

Tentative

Line nos

18-18

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_141

Severity

Informational

Confidence

Tentative

Line nos

19-19

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_142

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_143

Severity

Informational

Confidence

Tentative

Line nos

16-16

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_144

Severity

Informational

Confidence

Tentative

Line nos

17-17

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_145

Severity

Informational

Confidence

Tentative

Line nos

29-29

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_146

Severity

Informational

Confidence

Tentative

Line nos

18-18

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_147

Severity

Informational

Confidence

Tentative

Line nos

19-19

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_148

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_149

Severity

Informational

Confidence

Tentative

Line nos

18-18

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_150

Severity

Informational

Confidence

Tentative

Line nos

19-19

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_151

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_152

Severity

Informational

Confidence

Tentative

Line nos

18-18

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_153

Severity

Informational

Confidence

Tentative

Line nos

19-19

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_154

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_48

Severity

Informational

Confidence

Tentative

Line nos

30-30

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_49

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_50

Severity

Informational

Confidence

Tentative

Line nos

33-33

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_60

Severity

Informational

Confidence

Tentative

Line nos

30-30

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_61

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_62

Severity

Informational

Confidence

Tentative

Line nos

33-33

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_500_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_54

Severity

Informational

Confidence

Tentative

Line nos

30-30

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_55

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_56

Severity

Informational

Confidence

Tentative

Line nos

33-33

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_mainnet.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_51

Severity

Informational

Confidence

Tentative

Line nos

30-30

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_52

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_53

Severity

Informational

Confidence

Tentative

Line nos

33-33

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_mask_history_position_1000_ropsten.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_57

Severity

Informational

Confidence

Tentative

Line nos

46-46

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_58

Severity

Informational

Confidence

Tentative

Line nos

47-47

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_59

Severity

Informational

Confidence

Tentative

Line nos

53-53

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_1324_13

Severity

Low

Confidence

Tentative

Line nos

140-140

Action Taken

Fixed

Bug Type

WEAK PRNG

File Location

/contracts/qualification_luckydraw.sol

Issue Description

Random numbers find their use in many features and function logics. The sole purpose of these random number generators is that it should not be possible to guess or enumerate them.
The contract was using functions like `block.timestamp`, `now` or `blockhash` which can be manipulated by miners to some extent, and therefore the integrity of the contract is compromised.

Issue Remediation

It is recommended to go through the contract and check these functions where the `block.timestamp` and the `blockhash` are used to make sure that the function logic cannot be manipulated and abused.

Scan History

Critical High Medium Low Informational Gas

No	Date	Score	Scan Overview
1.	2023-05-24	70.87	0 4 0 17 59 41
2.	2023-04-18	71.46	0 3 0 17 60 41
3.	2023-04-10	66.41	0 3 0 17 60 41
4.	2023-04-10	63.30	0 4 0 28 60 41
5.	2023-04-10	63.30	0 7 0 30 60 41
6.	2023-04-07	63.30	0 7 0 30 60 41
7.	2023-04-07	63.30	0 7 0 30 60 41
8.	2023-04-07	63.18	0 7 0 30 61 41
9.	2023-04-07	63.31	0 7 0 31 62 41

Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.