Certified
Certified Blockchain Security Professional™

Certified
Certified Solidity Developer™

OVERVIEW

PROJECT SUMMARY

Project	**PokerSharkSociety**
Platform	N/a
Language	Solidity

AUDIT SUMMARY

Date	02-03-2022	
Audit Type	Static Analysis, Manual Review	
Audit Result	**Pending**	
Auditor	**Jarmo van de Seijp**	https://tinyurl.com/Jvdseijp

RISK SUMMARY

Risk Level	Total	Found	Pending	Solved	Acknowledgde	Objected
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	3	3	3	0	0	0
Minor	6	6	6	0	0	0
Informative	55	55	55	0	0	0
Discussion	0	0	0	0	0	0

FINDINGS

Function Default Visibility

SWC-ID: SWC-100

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:
Functions that do not have a function visibility type specified are public by default.
This can lead to a vulnerability if a developer forgot to set the visibility and a
malicious user is able to make unauthorized or unintended state changes.

Category	Risk Level	Number of Findings	Status
SWC-100	Informative	**16**	pending

Constable State

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

Constant state variables should be declared constant to save gas.

Category	Risk Level	Number of Findings	Status
SWC-108	Informative	**1**	pending

Multiple Pragma Directives used

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

In Truffle or Hardhat projects, importing files with their original pragma directive is common practice. However, since this smart contract is a single-file contract, it is recommended to use 1 pragma directive at the top of the file.

Category	Risk Level	Number of Findings	Status
Pragma Directives	Informative	**17**	pending

Missing Events

Description:

The critical variables presaleCost and publicSaleCost play an important role in the smart contract, since they are key players in its initial and subsequent ecosystem. The change of these variables is not emitted as an event. This may cause 3rd party applications as well as users to miss the change in price, causing unwanted outcome for users or aggregators

Category	Risk Level	Number of Findings	Status
Missing-Events	Medium	**2**	pending

Shadowing State Variables

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables in particular with relation to privileged roles (**_owner**) can cause confusion or unexpected results.

Category	Risk Level	Number of Findings	Status
Variable Shadowing	Minor	**2**	pending

Push-Over-Pull

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

The transfer of the contract's ownership through the function **transferOwnership()** only has 1 check, which is to ensure that the new owner is not the 0 address. It does not, however, check whether or not the ownership can be accepted by the recipient **newOwner**. In the case of a transfer of ownership to an incorrect address, or a smart contract that is not able to use the privileged functions, ownership of the contract is lost permanently with no way of getting it back. It is therefore advisable to use a pull method as opposed to push, in which case the **newOwner** would have to pro-actively accept ownership upon receiving it.

Category	Risk Level	Number of Findings	Status
Push over Pull	Minor	**1**	pending

Shadowing State Variables

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables in particular with relation to privileged roles (**_owner**) can cause confusion or unexpected results.

Category	Risk Level	Number of Findings	Status
Variable Shadowing	Minor	**2**	pending

Unused Code

Relationship:
CWE-1164: Irrelevant Code

Description:

Unused variables are allowed in Solidity and they do not pose a direct security issue. It is best practice though to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)

- indicate bugs or malformed data structures and they are generally a sign of poor code quality

- cause code noise and decrease readability of the code

Category	Risk Level	Number of Findings	Status
Dead Code	informational	**21**	pending

Comparison to Boolean constant

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

Boolean constants in code have only a few legitimate uses. Other uses (in complex expressions, as conditionals) indicate either an error or, most likely, the persistence of faulty or redunant code.

Relevance:

TokenUri compares a state variable to false. - *if (revealed == false)*

Category	Risk Level	Number of Findings	Status
Boolean Comparison	minor	**1**	pending

Risk of Centralization

Description:

There is only 1 privileged role in the smart contract, owner, who controls every privileged function. If the account or owner is compromised or if accesss is lost, the project could suffer losses. Since IMX privilege is only set and used once, this is role is not counted towards the risk of centrealization

Relevance:

14 critical privileged functions are controlled by owner

Category	Risk Level	Number of Findings	Status
Centralizaiton Risk	medium	**1**	pending

Final Note:

There is a compilation warning give by the compiler:
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon).

The smart contract has room for optimization, so there is a chance that cleaning up the code will result in a contract that falls below the 24576 bytes limit. If not, you can use the optimizer to run the contract before deploying it by a number of times, and it will check it there is expected use of functions that can be optimized in memory allocation. If this does not work, you may need to cut down on code, as this limit is enforced on mainnets to prevent DDOS attacks on smart contracts.

Code relevance and their respective lines:

Mintable.constructor(address,address)._owner (PokerShark.sol#595) shadows:
- Ownable._owner (PokerShark.sol#262) (state variable)
PokerSharkSociety.walletOfOwner(address)._owner (PokerShark.sol#1460) shadows:
- Ownable._owner (PokerShark.sol#262) (state variable)

PokerSharkSociety.setpresaleCost(uint256) (PokerShark.sol#1515-1517) should emit an event for:
- presaleCost = _cost (PokerShark.sol#1516)
PokerSharkSociety.setPublicSaleCost(uint256) (PokerShark.sol#1519-1521) should emit an event for:
- publicSaleCost = _cost (PokerShark.sol#1520)

Different versions of Solidity is used:
- Version used: ['>=0.7.0<0.9.0', '^0.8.0', '^0.8.4']
- ^0.8.4 (PokerShark.sol#5)
- ^0.8.0 (PokerShark.sol#103)
- ^0.8.0 (PokerShark.sol#149)
- ^0.8.0 (PokerShark.sol#219)
- ^0.8.0 (PokerShark.sol#246)
- ^0.8.0 (PokerShark.sol#324)
- ^0.8.4 (PokerShark.sol#543)
- ^0.8.4 (PokerShark.sol#571)
- ^0.8.4 (PokerShark.sol#584)
- ^0.8.0 (PokerShark.sol#630)
- ^0.8.0 (PokerShark.sol#660)
- ^0.8.0 (PokerShark.sol#688)
- ^0.8.0 (PokerShark.sol#719)
- ^0.8.0 (PokerShark.sol#864)
- ^0.8.0 (PokerShark.sol#893)
- ^0.8.0 (PokerShark.sol#1319)
- >=0.7.0<0.9.0 (PokerShark.sol#1373)

Address.functionCall(address,bytes) (PokerShark.sol#400-402) is never used and should be removed
Address.functionCall(address,bytes,string) (PokerShark.sol#410-416) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PokerShark.sol#429-435) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (PokerShark.sol#443-454) is never used and should be removed
Address.functionDelegateCall(address,bytes) (PokerShark.sol#489-491) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (PokerShark.sol#499-508) is never used and should be removed
Address.functionStaticCall(address,bytes) (PokerShark.sol#462-464) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PokerShark.sol#472-481) is never used and should be removed
Address.sendValue(address,uint256) (PokerShark.sol#375-380) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (PokerShark.sol#516-536) is never used and should be removed
Bytes.fromUint(uint256) (PokerShark.sol#13-31) is never used and should be removed
Bytes.substring(bytes,uint256,uint256) (PokerShark.sol#69-79) is never used and should be removed
Context._msgData() (PokerShark.sol#236-238) is never used and should be removed
Counters.decrement(Counters.Counter) (PokerShark.sol#131-137) is never used and should be removed
Counters.reset(Counters.Counter) (PokerShark.sol#139-141) is never used and should be removed
ERC721._baseURI() (PokerShark.sol#993-995) is never used and should be removed
ERC721._burn(uint256) (PokerShark.sol#1190-1202) is never used and should be removed
Mintable._mintFor(address,uint256,bytes) (PokerShark.sol#618-622) is never used and should be removed
Strings.toHexString(uint256) (PokerShark.sol#185-196) is never used and should be removed
Strings.toHexString(uint256,uint256) (PokerShark.sol#201-211) is never used and should be removed

```
Bytes.alphabet (PokerShark.sol#33) is never used in Bytes (PokerShark.sol#7-95)

PokerSharkSociety.maxSupply (PokerShark.sol#1387) should be constant

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (PokerShark.sol#295-297)
name() should be declared external:
- ERC721.name() (PokerShark.sol#967-969)
symbol() should be declared external:
- ERC721.symbol() (PokerShark.sol#974-976)
tokenURI(uint256) should be declared external:
- ERC721.tokenURI(uint256) (PokerShark.sol#981-986)
- PokerSharkSociety.tokenURI(uint256) (PokerShark.sol#1485-1505)
approve(address,uint256) should be declared external:
- ERC721.approve(address,uint256) (PokerShark.sol#1000-1010)
setApprovalForAll(address,bool) should be declared external:
- ERC721.setApprovalForAll(address,bool) (PokerShark.sol#1024-1026)
transferFrom(address,address,uint256) should be declared external:
- ERC721.transferFrom(address,address,uint256) (PokerShark.sol#1038-1047)
safeTransferFrom(address,address,uint256) should be declared external:
- ERC721.safeTransferFrom(address,address,uint256) (PokerShark.sol#1052-1058)
totalSupply() should be declared external:
- PokerSharkSociety.totalSupply() (PokerShark.sol#1417-1419)
mint(uint256,uint256[]) should be declared external:
- PokerSharkSociety.mint(uint256,uint256[]) (PokerShark.sol#1429-1434)
mintForAddress(uint256,address,uint256[]) should be declared external:
- PokerSharkSociety.mintForAddress(uint256,address,uint256[]) (PokerShark.sol#1436-1438)
mintForSelf(uint256,uint256[]) should be declared external:
- PokerSharkSociety.mintForSelf(uint256,uint256[]) (PokerShark.sol#1440-1442)
walletOfOwner(address) should be declared external:
- PokerSharkSociety.walletOfOwner(address) (PokerShark.sol#1460-1483)
setRevealed(bool) should be declared external:
- PokerSharkSociety.setRevealed(bool) (PokerShark.sol#1507-1509)
setMerkleRoot(bytes32) should be declared external:
- PokerSharkSociety.setMerkleRoot(bytes32) (PokerShark.sol#1511-1513)
setpresaleCost(uint256) should be declared external:
- PokerSharkSociety.setpresaleCost(uint256) (PokerShark.sol#1515-1517)
setPublicSaleCost(uint256) should be declared external:
- PokerSharkSociety.setPublicSaleCost(uint256) (PokerShark.sol#1519-1521)
setPublicSale(bool) should be declared external:
- PokerSharkSociety.setPublicSale(bool) (PokerShark.sol#1523-1525)
setPreSale(bool) should be declared external:
- PokerSharkSociety.setPreSale(bool) (PokerShark.sol#1527-1529)
setMaxMintAmountPerTx(uint256) should be declared external:
- PokerSharkSociety.setMaxMintAmountPerTx(uint256) (PokerShark.sol#1531-1533)
setMaxWalletLimit(uint256) should be declared external:
- PokerSharkSociety.setMaxWalletLimit(uint256) (PokerShark.sol#1535-1537)
setUriSuffix(string) should be declared external:
- PokerSharkSociety.setUriSuffix(string) (PokerShark.sol#1547-1549)
setPaused(bool) should be declared external:
- PokerSharkSociety.setPaused(bool) (PokerShark.sol#1551-1553)
withdraw() should be declared external:
- PokerSharkSociety.withdraw() (PokerShark.sol#1555-1558)
checkIfMinted(uint256) should be declared external:
- PokerSharkSociety.checkIfMinted(uint256) (PokerShark.sol#1560-1562)
```

AUDIT RESULT

Basic Coding Bugs

1. Constructor Mismatch

o Description: Whether the contract name and its constructor are not identical to each other.

o Result: PASSED

o Severity: Critical

Ownership Takeover

o Description: Whether the set owner function is not protected.

o Result: PASSED

o Severity: Critical

Redundant Fallback Function

o Description: Whether the contract has a redundant fallback function.

o Result: PASSED

o Severity: Critical

Overflows & Underflows

Description: Whether the contract has general overflow or underflow Vulnerabilities

o Result: PASSED

o Severity: Critical

Reentrancy

o Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.

o Result: PASSED

o Severity: Critical

MONEY-Giving Bug

o Description: Whether the contract returns funds to an arbitrary address.

o Result: PASSED

o Severity: High

Blackhole

o Description: Whether the contract locks ETH indefinitely: merely in without out.

o Result: PASSED

o Severity: High

Unauthorized Self-Destruct

o Description: Whether the contract can be killed by any arbitrary address.

o Result: PASSED

o Severity: Medium

Revert DoS

o Description: Whether the contractis vulnerable to DoSattack because of unexpected revert.

o Result: PASSED

o Severity: Medium

Unchecked External Call

o Description: Whether the contract has any external call without checking the return value.

o Result: PASSED

o Severity: Medium

Gasless Send

o Description: Whether the contractis vulnerable to gasless send.

o Result: PASSED

o Severity: Medium

Send Instead of Transfer

o Description: Whether the contract uses send instead of transfer.

o Result: PASSED

o Severity: Medium

Costly Loop

o Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.

o Result: PASSED

o Severity: Medium

(Unsafe) Use of Untrusted Libraries

o Description: Whether the contract use any suspicious libraries.

o Result: PASSED

o Severity: Medium

(Unsafe) Use of Predictable Variables

o Description: Whether the contract contains any randomness variable, but its value can be predicated.

o Result: PASSED

o Severity: Medium

Transaction Ordering Dependence

o Description: Whether the final state of the contract depends on the order of the transactions.

o Result: PASSED

o Severity: Medium

. Deprecated Uses

o Description: Whether the contract use the deprecated tx.origin to perform the authorization.

o Result: PASSED

o Severity: Medium