CFG NINJA AUDITS

Security Assessment

Squid Girl Token

November 22, 2023

Audit Status: Pass

Audit Edition: Standard

POWERED BY BLADE POOL

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
Critical	Danger or Potential Problems.
High	Be Careful or Fail test.
Low	Pass, Not-Detected or Safe Item.
Informational	Function Detected

Manual Code Review Risk Results

Contract Priviledge	Description
Buy Tax	0%
Sale Tax	0%
Cannot Sale	Pass
Cannot Sale	Pass
Max Tax	0%
Modify Tax	No
Fee Check	Pass
Is Honeypot?	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	Not Detected

Contract Priviledge	Description
Pause Transfer?	Not Detected
Max Tx?	Fail
Is Anti Whale?	Not Detected
Is Anti Bot?	Not Detected
Is Blacklist?	Not Detected
Blacklist Check	Pass
is Whitelist?	No Detected
Can Mint?	Pass
Is Proxy?	Not Detected
Can Take Ownership?	Not Detected
Hidden Owner?	Not Detected
Owner	0xfFD3339B7a073e2816a7A2d7664CC0537A091073
Self Destruct?	Not Detected
External Call?	Not Detected
Other?	Not Detected
Holders	1
Auditor Confidence	High
KYC Completed	No

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Project Overview

Token Summary

Parameter	Result
Address	00x94280A2C7Ae819e6E1b25F21c80334159019a164
Name	Squid Girl
Token Tracker	Squid Girl (SQUIG)
Decimals	18
Supply	100,000,000
Platform	Ethereum
compiler	v0.8.7+commit.e28d00a7
Contract Name	SquidGirl
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0xa60b8883a78CA068d30BEe270EE78428CaC8D7F5#code
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
Squid Girl	00x94280A2C7Ae819e6E1b25F21c80334159019a164	Yes

TestNet Contract Assessed Contract Name

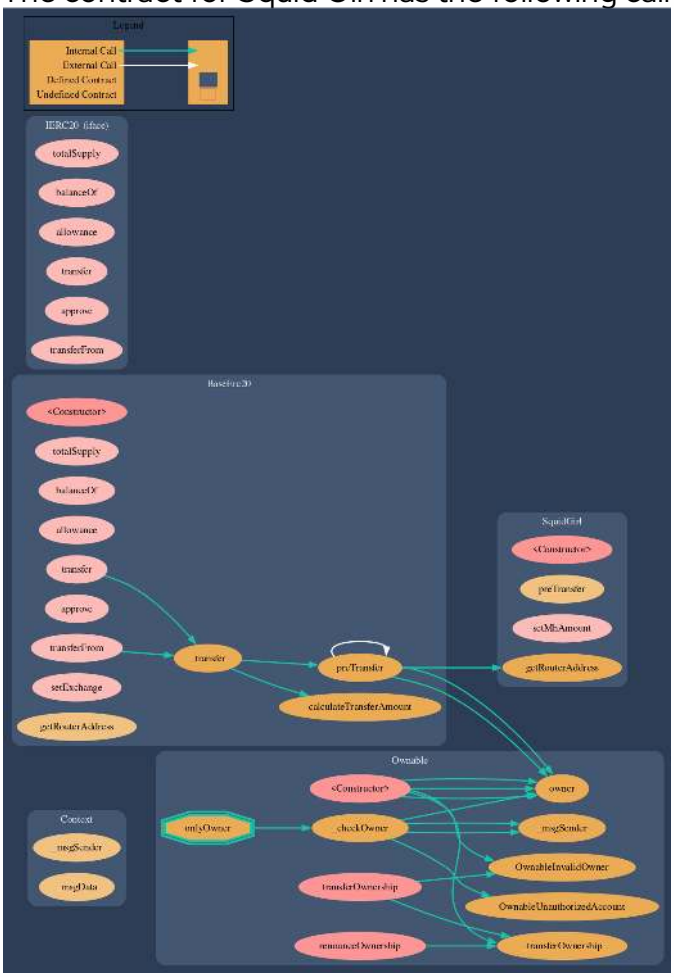
Name	Contract	Live
Squid Girl	0x8f8C3d6F67ce626C63560DCeB928f27Ba087329A	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
SQUIG	dab5d1d957864f411f0cc5a7a113466b8b5b83a8	squidgirl.sol

Call Graph

The contract for Squid Girl has the following call graph structure.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	squidgirl.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	squidgirl.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	squidgirl.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	squidgirl.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	squidgirl.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	squidgirl.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	squidgirl.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	squidgirl.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	squidgirl.sol	L: 85 C: 10
SWC-109	Pass	Uninitialized Storage Pointer.	squidgirl.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	squidgirl.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	squidgirl.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	squidgirl.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	squidgirl.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	squidgirl.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	squidgirl.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	squidgirl.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	squidgirl.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	squidgirl.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	squidgirl.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	squidgirl.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	squidgirl.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	squidgirl.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	squidgirl.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	squidgirl.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	squidgirl.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	squidgirl.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	squidgirl.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	squidgirl.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	squidgirl.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	squidgirl.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	squidgirl.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	squidgirl.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	squidgirl.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	squidgirl.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	squidgirl.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	squidgirl.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

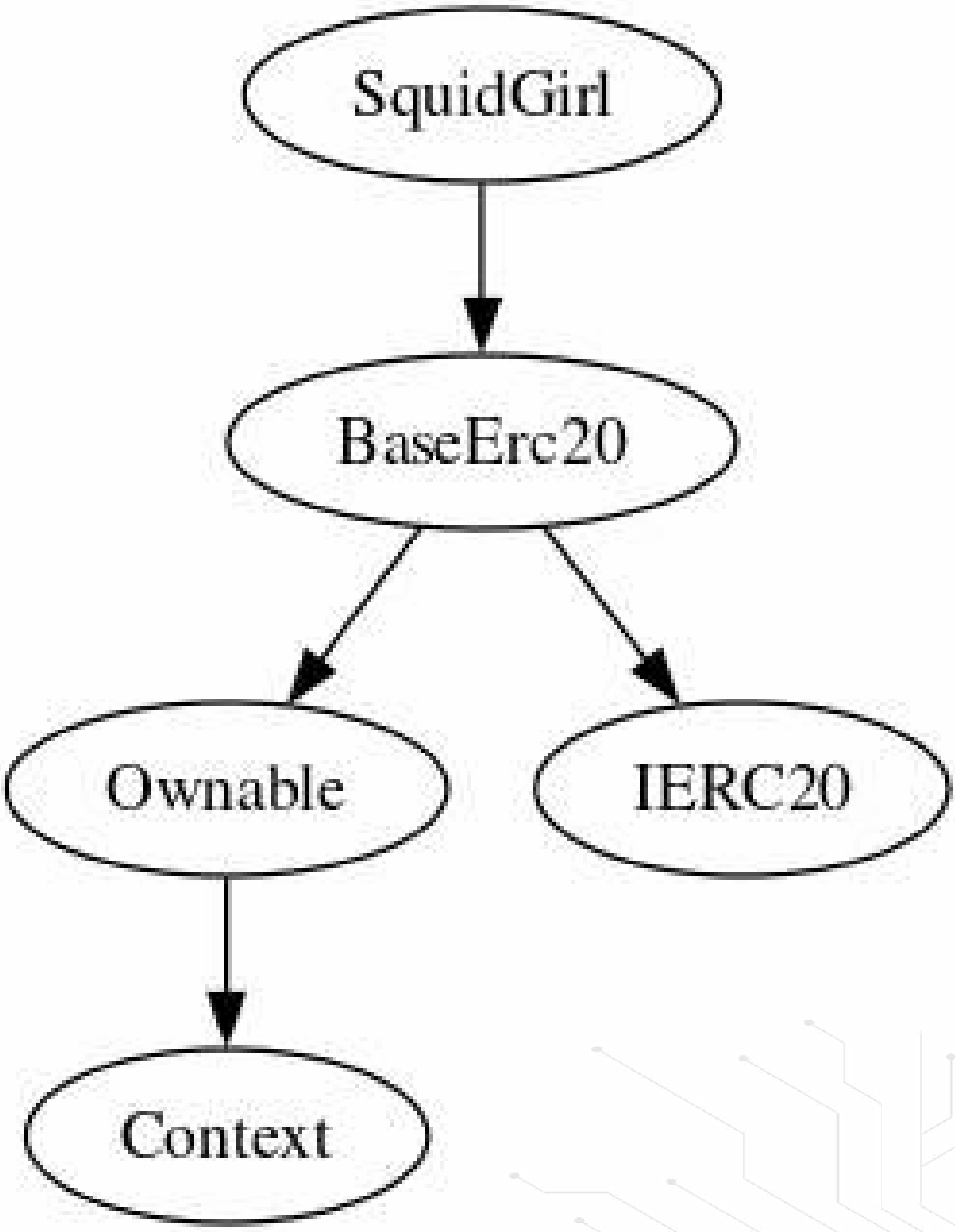
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Inheritance

The contract for Squid Girl has the following inheritance structure.



SQUIG-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	squidgirl.sol: L: 329 C: 14	Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the all onlyOwners.

Recommendation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. all onlyOwners.

Mitigation

References:

Zero Address check. The danger!!!

SQUIG-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	squidgirl.sol: L: 329 C: 14, L: 265 C: 14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Recommendation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Mitigation

References:

Understanding Events in Smart Contracts

SQUIG-19 | Centralization Privileges of SQUIG

Category	Severity	Location	Status
Coding Style	Medium	squidgirl.sol: L: 265 C: 14, L: 329 C: 14	Detected

Description

Centralized Privileges are found on the following functions.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
setExchange		External
setMhAmount		External

Recommendation

Inheriting from Ownable and calling its constructor on yours ensures that the address deploying your contract is registered as the owner. The onlyOwner modifier makes a function revert if not called by the address registered as the owner. It is important that deployr or owner secure the credentials that has owner priviledge to ensure the security of the project.

Mitigation

References:

Guide to Ownership and Access Control in Solidity

Writing Clean Code for Solidity: Best Practices for Solidity Development

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
High	0	0	0
Medium	1	0	0
Low	2	0	0
Informational	0	0	0
Total	3	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/SquidGirlETH	Pass
Other		Fail
Website	https://squidgirl.io/	Pass
Telegram	https://t.me/SquidGirlETH	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Assessment Results

Score Results

Review	Score
Overall Score	85/100
Auditor Score	90/100
Review by Section	**Score**
Manual Scan Score	16
SWC Scan Score	35
Advance Check Score	34

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Assessment Results

Important Notes:

- No issues or vulnerabilities were found.
- Please DYOR on the project.

Auditor Score =90
Audit Passed

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

CFGNINJA has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will CFGNINJA or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access 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. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.