



# SECURITY ASSESSMENT **Evil Kermit TOKEN**

April 5, 2024

Audit Status: Pass



# RISK ANALYSIS | Evil Kermit.

## ■ Classifications of Manual Risk Results

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

## ■ Manual Code Review Risk Results

Contract Security	Description
 Buy Tax	0%
 Sale Tax	0%
 Cannot Buy	Pass
 Cannot Sale	Pass
 Max Tax	0%
 Modify Tax	No
 Fee Check	Pass
 Is Honeypot?	Not Detected
 Trading Cooldown	Not Detected
 Enable Trade?	True
 Pause Transfer?	Not Detected

Contract Security	Description
● Max Tx?	Pass
● Is Anti Whale?	Not Detected
● Is Anti Bot?	Not Detected
● Is Blacklist?	Not Detected
● Blacklist Check	Pass
● is Whitelist?	Pass
● Can Mint?	Pass
● Is Proxy?	Not Detected
● Can Take Ownership?	Not Detected
● Hidden Owner?	Not Detected
i Owner	TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA
● Self Destruct?	Not Detected
● External Call?	Not Detected
● Other?	Not Detected
● Holders	3
● Audit Confidence	Low
● Authority Check	Pass
● Freeze Check	Pass

The summary section reveals the strengths and weaknesses identified during the assessment, including any vulnerabilities or potential risks that may exist. It serves as a valuable snapshot of the overall security status of the audited project. However, it is highly recommended to read the entire security assessment report for a comprehensive understanding of the findings. The full report provides detailed insights into the assessment process, methodology, and specific recommendations for addressing the identified issues.

CFG Ninja Verified on April 5, 2024



## Evil Kermit

### Executive Summary

TYPES

DeFi

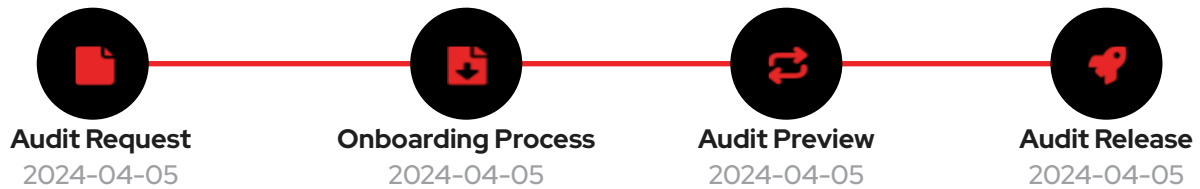
ECOSYSTEM

SOLANA

LANGUAGE

RUST

### Timeline



### Vulnerability Summary



2

Total Findings

0

Resolved

2

Pending

2

Unresolved

#### 0 Critical

Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.

#### 1 High

0 Resolved, 1 Pending

High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.

#### 0 Medium

Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.

#### 0 Low

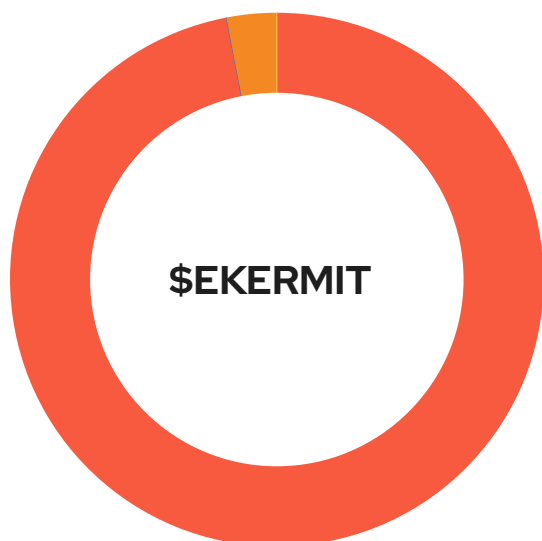
Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.

#### 1 Informational

0 Resolved, 1 Pending

Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

## Token Distribution



### Burn

Burned amount send to the deadWallet.

0%

### Liquidity

Liquidity tokens are split from sale into the pool.

32%

### Fairlaunch

Tokens allocated for the sale.

0%

### Development and Ecosystem

Ecosystem

0%

### Team and Advisors

Teams

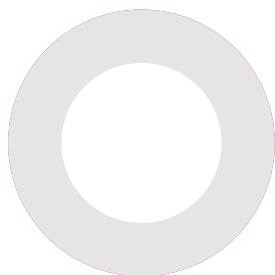
1%

### Community and Partnerships

Community

0%

## Total Unlock Progress



<span style="color: green;">■</span> Unlocked	0	0%
<span style="color: magenta;">■</span> Total Locked	1000000000000	150000.00015%
<span style="color: grey;">■</span> Untracked	-999333333334	-149900.00015%

# PROJECT OVERVIEW | Evil Kermit.

## Token Summary

Parameter	Result
Address	2Uyadf6xEvN8mAgvtrhURpSwkHHXvqd43HHvz73iH72P
Name	Evil Kermit
Token Tracker	Evil Kermit (\$EKERMIT)
Decimals	9
Supply	666,666,666
Platform	SOLANA
Compiler	v0.8.20+commit.a1b79de6
Contract Name	\$EKERMIT
Optimization	Yes with 200 runs
LicenseType	Unlicensed
Language	RUST
Codebase	<a href="https://solscan.io/token/2Uyadf6xEvN8mAgvtrhURpSwkHHXvqd43HHvz73iH72P#metadata">https://solscan.io/ token/2Uyadf6xEvN8mAgvtrhURpSwkHHXvqd43HHvz73iH72P#metadata</a>

# PROJECT OVERVIEW | Evil Kermit.

## Token Summary - Solana

Parameter	Result
Address	2Uyadf6xEvN8mAgvtrhURpSwkHHXvqd43HHvz73iH72P
Name	Evil Kermit
Token Tracker	Evil Kermit (\$EKERMIT)
Decimals	9
Supply	666,666,666
Platform	SOLANA
Program	TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA
Creator Name	Bladepool
Creation Site	<a href="https://cfg.ninja">https://cfg.ninja</a>
Language	RUST
Image	<a href="https://arweave.net/UyeRGxDt4tvCVE0ualzZ1tb7XI-WHm9n64SPOSOGiYk">https://arweave.net/UyeRGxDt4tvCVE0ualzZ1tb7XI-WHm9n64SPOSOGiYk</a>
Metadata File Type	JSON
Solana Source	<a href="https://solscan.io/token/Cb85u66JqUThurhWnm5pkhmw3Y58zobu5Mf3CbAnq8RV#metadata">https://solscan.io/token/Cb85u66JqUThurhWnm5pkhmw3Y58zobu5Mf3CbAnq8RV#metadata</a>

# PROJECT OVERVIEW | Evil Kermit.

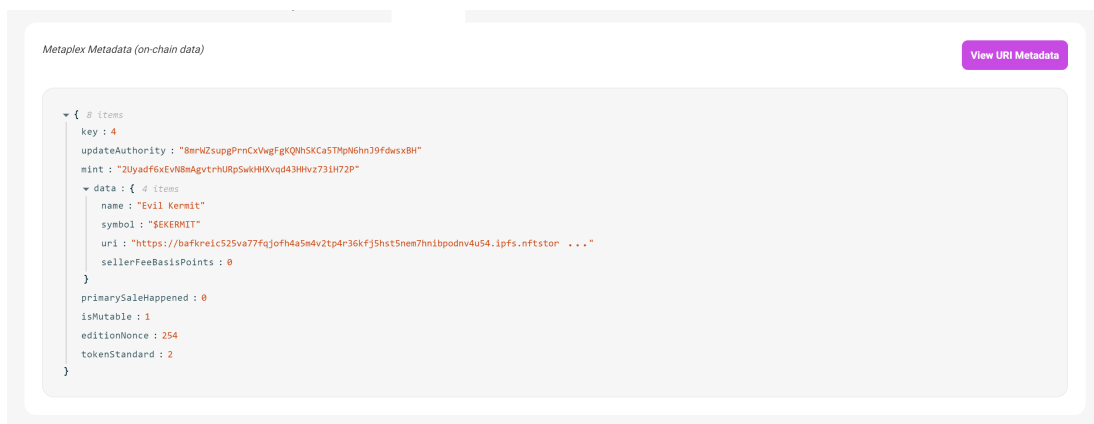
## Metaplex Metadata (on-chain data)

Solana metadata refers to the additional information associated with a digital asset or NFT (Non-Fungible Token) on the Solana blockchain. It includes details such as the name, description, image, attributes, and other relevant data about the asset.

In the context of Solana, metadata is typically stored in a JSON format and is linked to the asset's unique identifier or token ID. This metadata provides important information about the asset, allowing users and applications to understand and interact with it.

Solana metadata can be used for various purposes, including displaying asset information in marketplaces, creating rich visual representations of NFTs, and enabling advanced functionalities like royalties, provenance tracking, and interoperability across different platforms.

It's worth noting that the specific structure and content of Solana metadata can vary depending on the project or application that utilizes the Solana blockchain.





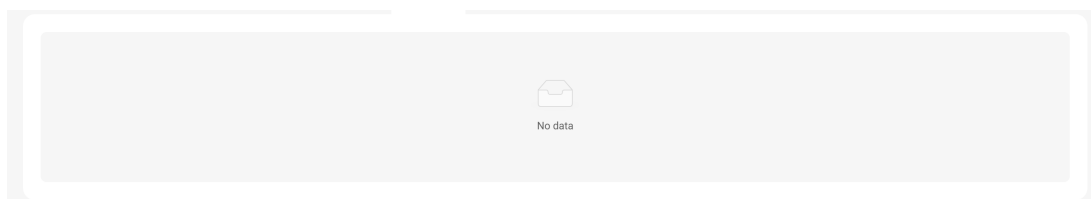
## \$EKERMIT | Metadata Results.

Parameter	Value	Description
key	4	This is an integer value (int4) that represents the key associated with the root object.
updateAuthority	EZSjvA5cy4ymQep8rMAK JXP3F6iqSmShbdL9w1NV HmLd	This is a string value that represents the update authority for the program.
mint	Cb85u66JqUThurhWnm5 pkhmw3Y58zobu5Mf3Cb Anq8RV	This is a string value that represents the mint address for the program.
name	Evil Kermit	This is a string value that represents the name of the token.
symbol	\$EKERMIT	This is a string value that represents the symbol of the token.
uri	https://bafkreic525va77fqj ofh4a5m4v2tp4r36kfj5hst 5nem7hnibpodnv4u54.ipfs .nftstorage.link	This is a string value that represents the URI (Uniform Resource Identifier) of the token.
sellerFeeBasisPoints	0	This is an integer value (int0) that represents the seller fee basis points for the token
primarySaleHappened	0	This is an integer value (int0) that indicates whether the primary sale of the token has happened.
isMutable	1	This is an integer value (int1) that indicates whether the token is mutable. The specific value is 1, which suggests that the token is mutable. and 0 suggest is not mutable.
editionNonce	255	This is an integer value (int255) that represents the edition nonce for the token.
tokenStandard	2	This is an integer value (int2) that represents the token standard for the program.

# PROJECT OVERVIEW | Evil Kermit.

## URI Metadata

URI metadata in Solana refers to the metadata associated with a token that is retrieved from its URI (Uniform Resource Identifier). In this case, the token's URI is <https://bafkreic525va77fqjofh4a5m4v2tp4r36kfj5hst5nem7hnibpodnv4u54.ipfs.nftstorage.link>.



## TECHNICAL FINDINGS | Evil Kermit.



Smart contract security audits classify risks into several categories: Critical, High, Medium, Low, and Informational. These classifications help assess the severity and potential impact of vulnerabilities found in smart contracts.

### Classification of Risk

Severity	Description
 Critical	Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.
 High	High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.
 Medium	Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.
 Low	Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.
 Informational	Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

By categorizing risks into these classifications, smart contract security audits can prioritize the resolution of critical and high-risk vulnerabilities to ensure the contract's overall security and protect user funds and data.

## \$EKERMIT-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 High	\$EKERMIT.sol:	 Detected

### Description

All of the Evil Kermit tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

### Recommendation



We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

### Mitigation

### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## I \$EKERMIT-24 | Ability to update token metadata..

Category	Severity	Location	Status
Coding Style	 Informational	\$EKERMIT.sol:	 Detected

### Description

The mint authority is responsible for creating new tokens, but it does not affect the ability to update the token metadata. Token metadata includes information such as the token name, symbol, image, and other details. This metadata can be updated by the token contract owner or any other authorized entity, regardless of the mint authority status..

### Recommendation

The updateAuthority can update the token metadata, this include name,symbol and logo. while this is highly unlikely we like to report it. .

### Mitigation

### References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

## FINDINGS

In this document, we present the findings and results of the smart contract security audit. The identified vulnerabilities, weaknesses, and potential risks are outlined, along with recommendations for mitigating these issues. It is crucial for the team to address these findings promptly to enhance the security and trustworthiness of the smart contract code.

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

In a smart contract, a technical finding summary refers to a compilation of identified issues or vulnerabilities discovered during a security audit. These findings can range from coding errors and logical flaws to potential security risks. It is crucial for the project owner to thoroughly review each identified item and take necessary actions to resolve them. By carefully examining the technical finding summary, the project owner can gain insights into the weaknesses or potential threats present in the smart contract. They should prioritize addressing these issues promptly to mitigate any risks associated with the contract's security. Neglecting to address any identified item in the security audit can expose the smart contract to significant risks. Unresolved vulnerabilities can be exploited by malicious actors, potentially leading to financial losses, data breaches, or other detrimental consequences. To ensure the integrity and security of the smart contract, the project owner should engage in a comprehensive review process. This involves understanding the nature and severity of each identified item, consulting with experts if needed, and implementing appropriate fixes or enhancements. Regularly updating and maintaining the smart contract's codebase is also essential to address any emerging security concerns. By diligently reviewing and resolving all identified items in the technical finding summary, the project owner can significantly reduce the risks associated with the smart contract and enhance its overall security posture.

## SOCIAL MEDIA CHECKS | Evil Kermit.

Social Media		URL	Result
Website	evil-kermit.fun		Pass
Telegram		<a href="https://t.me/EvilKermitSOLANA">https://t.me/EvilKermitSOLANA</a>	Pass
Twitter		<a href="https://twitter.com/EvilKermitSO">https://twitter.com/EvilKermitSO</a>	Pass
Facebook			N/A
Reddit	N/A		N/A
Instagram			N/A
CoinGecko	N/A		N/A
Github			N/A
CMC	N/A		N/A
Email	N/A		Contact
Other			N/A

From a security assessment standpoint, inspecting a project's social media presence is essential. It enables the evaluation of the project's reputation, credibility, and trustworthiness within the community. By analyzing the content shared, engagement levels, and the response to any security-related incidents, one can assess the project's commitment to security practices and its ability to handle potential threats.

### Social Media Information Notes:

**Auditor Notes:** Website needs a bit of improvement.

**Project Owner Notes:**

## ASSESSMENT RESULTS | Evil Kermit.

### Score Results

Review	Score
Overall Score	89/100
Auditor Score	80/100

Review by Section	Score
Manual Scan Score	22
SWC Scan Score	37
Advance Check Score	30

Our security assessment or audit score system for the smart contract and project follows a comprehensive evaluation process to ensure the highest level of security. The system assigns a score based on various security parameters and benchmarks, with a passing score set at 80 out of a total attainable score of 100. The assessment process includes a thorough review of the smart contracts codebase, architecture, and design principles. It examines potential vulnerabilities, such as code bugs, logical flaws, and potential attack vectors. The evaluation also considers the adherence to best practices and industry standards for secure coding. Additionally, the system assesses the projects overall security measures, including infrastructure security, data protection, and access controls. It evaluates the implementation of encryption, authentication mechanisms, and secure communication protocols. To achieve a passing score, the smart contract and project must attain a minimum of 80 points out of the total attainable score of 100. This ensures that the system has undergone a rigorous security assessment and meets the required standards for secure operation.





## Important Notes for \$EKERMIT

- No issues or vulnerabilities were found.

**Auditor Score =80**  
**Audit Passed**



## Appendix

### Finding Categories

#### Centralization / Privilege

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

#### Gas Optimization

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

#### Logical Issue

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

#### Control Flow

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

#### Volatile Code

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

#### Coding Style

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

#### Inconsistency

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

#### Coding Best Practices

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

## Disclaimer

The purpose of this disclaimer is to outline the responsibilities and limitations of the security assessment and smart contract audit conducted by Bladepool/CFG NINJA. By engaging our services, the project owner acknowledges and agrees to the following terms:

1. Limitation of Liability: Bladepool/CFG NINJA shall not be held liable for any damages, losses, or expenses incurred as a result of any contract malfunctions, vulnerabilities, or exploits discovered during the security assessment and smart contract audit. The project owner assumes full responsibility for any consequences arising from the use or implementation of the audited smart contract. 2. No Guarantee of Absolute Security: While Bladepool/CFG NINJA employs industry-standard practices and methodologies to identify potential security risks, it is important to note that no security assessment or smart contract audit can provide an absolute guarantee of security. The project owner acknowledges that there may still be unknown vulnerabilities or risks that are beyond the scope of our assessment. 3. Transfer of Responsibility: By engaging our services, the project owner agrees to assume full responsibility for addressing and mitigating any identified vulnerabilities or risks discovered during the security assessment and smart contract audit. It is the project owner's sole responsibility to ensure the proper implementation of necessary security measures and to address any identified issues promptly. 4. Compliance with Applicable Laws and Regulations: The project owner acknowledges and agrees to comply with all applicable laws, regulations, and industry standards related to the use and implementation of smart contracts. Bladepool/CFG NINJA shall not be held responsible for any non-compliance by the project owner. 5. Third-Party Services: The security assessment and smart contract audit conducted by Bladepool/CFG NINJA may involve the use of third-party tools, services, or technologies. While we exercise due diligence in selecting and utilizing these resources, we cannot be held liable for any issues or damages arising from the use of such third-party services. 6. Confidentiality: Bladepool/CFG NINJA maintains strict confidentiality regarding all information and data obtained during the security assessment and smart contract audit. However, we cannot guarantee the security of data transmitted over the internet or through any other means. 7. Not a Financial Advice: Bladepool/CFG NINJA please note that the information provided in the security assessment or audit should not be considered as financial advice. It is always recommended to consult with a financial professional or do thorough research before making any investment decisions.

By engaging our services, the project owner acknowledges and accepts these terms and releases Bladepool/CFG NINJA from any liability, claims, or damages arising from the security assessment and smart contract audit. It is recommended that the project owner consult legal counsel before entering into any agreement or contract.

