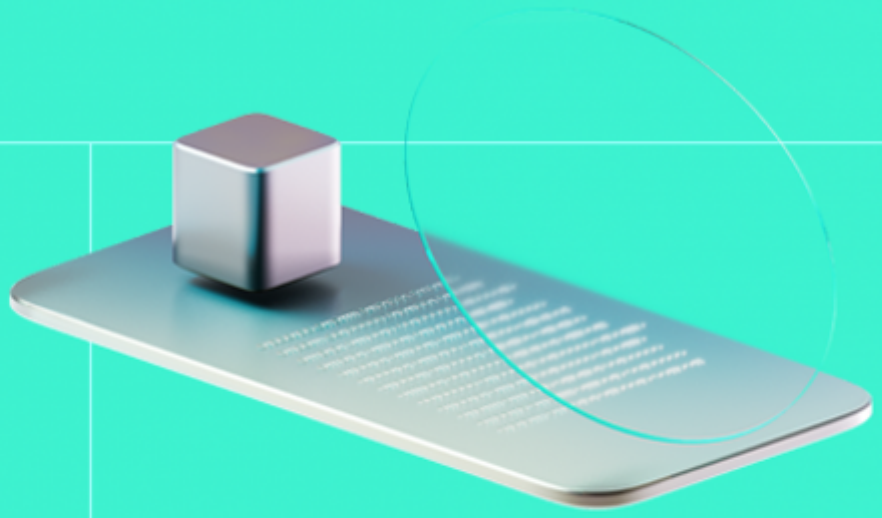
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Hello

Date: 23/05/2024

We express our gratitude to the Hello team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Hello Bridge is a Solana - EVM Bridge for the HELLO Token.

Platform: Solana

Language: Rust

Tags: Bridge

Timeline: 06/05/2024 - 09/05/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/sevenlabs-hq/hellolabs-token-bridge-program
Commit	2d7ba697864f8770e8ea431312883d1c9d7cf200

Audit Summary

7/10	6/10	N/A	7/10
Security score	Code quality score	Test coverage	Documentation quality score

Total 7/10

The system users should acknowledge all the risks summed up in the risks section of the report

4	1	1	2
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	1
High	0
Medium	0
Low	3

Vulnerability	Status
F-2024-2402 - Missing two-step ownership transfer	Mitigated
F-2024-2404 - Insufficient validation allows anyone to perform fake deposits	Mitigated
F-2024-2400 - Centralization concerns	Accepted
F-2024-2403 - The chain list might contain duplicates, which might break the protocol logic	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Hello
Audited By	Lukasz Mikula
Approved By	Ataberk Yavuzer, Przemyslaw Swiatowiec
Website	TBC
Changelog	10/05/2024 - Preliminary Report
	14/05/2024 - Final Report

Table of Contents

System Overview

Hello Bridge is a Solana - EVM Bridge with the following utilities:

- pausing the bridge
- changing the authority / signers
- deposit by users
- withdraw by users
- adding/removing supported EVM chains

Privileged roles

- Bridge authority is the most privileged role that can change bridge authority,
- Signers can confirm withdrawal transactions via an off-chain logic.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation quality score is **7** out of **10**.

- The documentation is missing
- Technical description is not provided.

Code quality

The total Code quality score is **4** out of **10**.

- The code does not utilize best practices in common patterns for example related to secure ownership transfers or decentralization
- The code is missing threshold safety checks to avoid human errors - for example, edge or incorrect values of key parameters are not checked and the program relies on the protocol administrator to set proper parameter values
- The code lacks proper event emissions which increases transparency and trackability of the protocol events
- Error messages are missing on unchecked arithmetic
- Finally, potential unbounded loops exist in the code that may lead to DoS issues

Test coverage

Code coverage of the project is N/A.

- Some unit tests were provided
- It was required to amend the tests in order to get them working due to a hardcoded, nonexistent private key
- The Score is N/A since there is no reliable tool to calculate Solana test coverage

Security score

Upon auditing, the code was found to contain **1** critical, **0** high, **0** medium, and **3** low severity issues, leading to a security score of **7** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **7**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

1. Out-of-Scope Components & 3rd Party Dependencies.

- Scope Definition and Security Guarantees: The audit does not cover all code in the repository. Contracts outside the audit scope may introduce vulnerabilities, potentially impacting the overall security due to the interconnected nature of smart contracts. A significant part of the protocol relies on backend logic which processes the bridge operations. Any risk related to that part of the protocol is unknown since backend was not subject to the audit.

2. Token Supply/ Minting

- Fixed Total Supply Post-Deployment: The token's total supply is determined at deployment and cannot be verified later, potentially limiting the project's adaptability and economic model flexibility.

3. Loop Efficiency

- Dynamic Array Iteration Gas Limit Risks: The project iterates over vectors but they seem to be limited to several items related to number of supported EVM chain, which as long as there are only few ones is safe, but otherwise leads to excessive gas costs, risking denial of service due to out-of-gas errors, directly impacting contract usability and reliability.

4. Permissions, Authorization & Access

- Absence of Time-lock Mechanisms for Critical Operations: Without time-locks on critical operations, there is no buffer to review or revert potentially harmful actions, increasing the risk of rapid exploitation and irreversible changes.
- Insufficient Multi-signature Controls for Critical Functions: The lack of multi-signature requirements for key operations centralizes decision-making power, increasing vulnerability to single points of failure or malicious insider actions, potentially leading to unauthorized transactions or configuration changes.

5. Centralization

- Single Points of Failure and Control: The project is partially centralized, introducing single points of failure and control, when it comes to the bridge authority able to pause the contract or disable/enable EVM chains. This centralization can lead to vulnerabilities in decision-making and operational processes, making the system more susceptible to targeted attacks or manipulation.
- Administrative Key Control Risks: The digital contract architecture relies on administrative keys for critical operations. Centralized control over these keys presents a significant security risk, as compromise or misuse can lead to unauthorized actions or loss of funds.
- Single Entity Upgrade Authority: The token ecosystem grants a single entity the authority to implement upgrades or changes. This centralization of power risks unilateral decisions that may not align with the community or stakeholders' interests, undermining trust and security.

Findings

Vulnerability Details

F-2024-2404 - Insufficient validation allows anyone to perform fake deposits - Critical

Description:

The deposit to EVM is triggered by operation `deposit_to_chain`, defined in **deposit_to_chain.rs**.

The operation can be performed by any user (who wants to bridge) and is executed in lines 64-72:

```
token::transfer_checked(CpiContext::new(
token_program.to_account_info(),
TransferChecked {
from: source.to_account_info(),
mint: mint.to_account_info(),
to: vault.to_account_info(),
authority: payer.to_account_info()
}
), amount, mint.decimals)?;
```

However, the token mint that is being bridged is not validated properly. The `HelloToken` mint is not saved in the program account, and instead there are some complex but ineffective requirements on the `vault` - it should have that specific token account and the `token::authority` should be set to be `bridge_info`. Additionally, the vault address is not validated as well.

Code snippet from **deposit_to_chain.rs** with the `DepositToChain` struct:

```
pub struct DepositToChain<'info> {
#[account(mut, seeds = [&"bridge_info".as_bytes()], bump = bridge_in
fo.bump)]
pub bridge_info: Account<'info, BridgeInfo>,

#[account(
init_if_needed,
payer = payer,
space = 8 + ChainInfo::INIT_SPACE,
seeds = [
&"chain_info".as_bytes(),
&payer.key().as_ref(),
&evm_chain_id.to_le_bytes().as_ref(),
&evm_address.as_ref()
],
bump
)]
pub chain_info: Account<'info, ChainInfo>,

pub mint: Account<'info, Mint>,

#[account(mut, token::mint = mint, token::authority = payer)]
pub source: Account<'info, TokenAccount>,

#[account(mut, token::mint = mint, token::authority = bridge_info)]
pub vault: Account<'info, TokenAccount>,

#[account(mut)]
pub payer: Signer<'info>,
```

```
pub token_program: Program<'info, Token>,
pub system_program: Program<'info, System>,
}
```

Due to this, an attacker can setup an arbitrary mint that bypasses these requirements, and as a result perform the deposit with an incorrect token, which would be accounted by the bridge as the correct deposit transaction (even that `HelloToken` was not transferred from the user).

Assets:

- instructions/deposit_to_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Mitigated

Classification

Impact: 5/5

Likelihood: 5/5

Exploitability: Independent

Complexity: Medium

Severity: Critical

Recommendations

Remediation: Since the `mint` is known on initialization as well as the `vault`, the best and safest way will be to save it and compare against as a validation of deposit - if the mint and the vault address do not match the original one, then the transaction should revert.

Resolution: As of commit **c50984**, the program now correctly validates token mint on deposit.

However there is no validation of vault.

That still poses a risk, that a malicious user will deposit legitimate token mint to an arbitrary crafted vault, which might break bridge accounting.

Such attack is much less likely to happen, because essentially someone would have to sacrifice their own tokens, to lock them in an inaccessible vault, just to have tokens bridged. But at this point a balance mismatch between EVM and solana side might occur.

The likelihood is low and attack is costly for the attacker, but since such possibility exists, it should be highlighted.

The remediation is to validate the Vault Pubkey and save it in state.rs, as other key program data.

Evidences

Proof of concept

Reproduce:

following unit test was added:

```
it("attack", async () => {

const attacker = anchor.web3.Keypair.generate();
signature = await provider.connection.requestAirdrop(
attacker.publicKey,
LAMPORTS_PER_SOL
);
await provider.connection.confirmTransaction(signature);
const attackerMint = anchor.web3.Keypair.generate();
const attackerTokenAccount = anchor.web3.Keypair.generate();
const attackerVault = anchor.web3.Keypair.generate();

const attackChainInfo = anchor.web3.PublicKey.findProgramAddressSync
(
[Buffer.from("chain_info"), attacker.publicKey.toBuffer(), evmChainI
dBuffer, evmAddressBytes],
program.programId
);


tx = new Transaction().add(
// Create malicious mint
SystemProgram.createAccount({
newAccountPubkey: attackerMint.publicKey,
fromPubkey: attacker.publicKey,
space: MintLayout.span,
lamports: await provider.connection.getMinimumBalanceForRentExemptio
n(MintLayout.span),
programId: TOKEN_PROGRAM_ID,
}),
createInitializeMint2Instruction(attackerMint.publicKey, 9, attacker
.publicKey, null),
// create token account for attacker
SystemProgram.createAccount({
newAccountPubkey: attackerTokenAccount.publicKey,
fromPubkey: attacker.publicKey,
space: AccountLayout.span,
lamports: await provider.connection.getMinimumBalanceForRentExemptio
n(AccountLayout.span),
programId: TOKEN_PROGRAM_ID,
}),
createInitializeAccount3Instruction(
attackerTokenAccount.publicKey,
attackerMint.publicKey,
attacker.publicKey
),
// create malicious valt
SystemProgram.createAccount({
newAccountPubkey: attackerVault.publicKey,
fromPubkey: attacker.publicKey,
space: AccountLayout.span,
lamports:
```

See more

F-2024-2400 - Centralization concerns - Low

Description:

Following components of the protocol might singlehandedly influence the state of the protocol and are controlled by one signer:

- status (pause of the protocol)
- disabling/enabling chains

In case of a private key compromise, those may become single points of failure.

However even if that happens, that still would not mean a direct profit for the attacker, the only outcome would be rendering the protocol unusable. An attacker would have to take additional actions in order to turn such attack into a financial gain e.g. by taking a short position against $HELLO.

For example, the bridge can be paused which requires only the authority alone to sign such operation in **update_status.rs**

```
pub fn handle(ctx: Context<UpdateStatus>, new_status: bool) -> Resul
t<()> {
ctx.accounts.bridge_info.status = new_status;
Ok(())
}
```

as an additional note, renaming the variable to paused would add additional clarity to the code.

Similarly, the operation of adding/removing EVM chain support requires also only the authority to sign the operation:

```
pub fn handle(ctx: Context<DisableEvmChain>, evm_chain_id: u32) -> R
esult<()> {
let index = ctx
.accounts
.bridge_info
.enable_evm_chains
.iter()
.position(|&r| r == evm_chain_id)
.ok_or(BridgeError::EvmChainNotFound)?;

ctx.accounts.bridge_info.enable_evm_chains.remove(index);
Ok(())
}
```

Assets:

- instructions/disable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/enable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_status.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Accepted

Classification

Impact:	5/5
Likelihood:	1/5
Exploitability:	Dependent
Complexity:	Medium
Severity:	Low

Recommendations

Remediation:	Consider using a multisignature wallet or require multiple signers to sign sensitive operations. In case of a private key leak, it will be still not possible to damage the protocol since there will be other signers requried.
Resolution:	A patch to this finding was not provided in the commit **c50984**.

F-2024-2402 - Missing two-step ownership transfer - Low

Description:

The ownership transfer, similarly to signers change, relies on an one-way operation which just requires specifying new address and if the signer check is correct, new address is saved to the protocol configuration account. However if by chance the new address is incorrect, there will be no way to recover the role anymore. Since roles are unique, this may lead to lost of control over the protocol as a result of a simple error.

The ownership transfer that utilizes aforementioned pattern is present in **update_authority.rs**:

```
pub fn handle(ctx: Context<UpdateAuthority>, new_authority: Pubkey)
-> Result<()> {
ctx.accounts.bridge_info.authority = new_authority;
Ok(())
}
```

Similar pattern is used to update withdraw signer, in **update_withdraw_signer_one.rs** and **update_withdraw_signer_two.rs**:

```
pub fn handle(ctx: Context<UpdateWithdrawSigner1>, new_withdraw_sign
er_1: Pubkey) -> Result<()> {
ctx.accounts.bridge_info.withdraw_signer_1 = new_withdraw_signer_1;
Ok(())
}
```

Assets:

- instructions/update_authority.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_withdraw_signer_one.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_withdraw_signer_two.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Mitigated

Classification

Impact: 3/5

Likelihood: 1/5

Exploitability: Dependent

Complexity: Simple

Severity: Low

Recommendations

Remediation: It is recommended to use two-step transfer ownership logic.

In a two-step process, in step one, the current role holder may appoint the receiver.

Next, the receiver is able to accept the role (requires signer check). If it succeeds, only then the role change is finalized.

It is important to clear the temporarily appointed roles afterwards, so the process is closed.

Resolution: The commit **c50984** implements changes for signer one and two but does not apply similar logic to the authority role.

F-2024-2403 - The chain list might contain duplicates, which might break the protocol logic - Low

Description:

The enable/disable EVM chain list uses a simple logic of push/pop new ID to the list to make it enabled. However, there is no uniqueness check. While it is still a privileged operation which might suggest that an authorized, experienced user will be operating this utility, it is possible that due to a mistake a chain will be added twice or more. In that case, it may cause the protocol to not work properly, e.g. on next removal attempt, the chain ID will maintain, since the removal relies on an iteration over list, and removed index of first occurence, and not all occurences.

For example, in **enable_evm_chain.rs**, if by a mistake chain is added twice, it will be simply pushed to the vector. For example, nothing prevents the vector to become `[1,2,3,2]`

```
pub fn handle(ctx: Context<EnableEvmChain>, evm_chain_id: u32) -> Re
sult<()> {
ctx.accounts
.bridge_info
.enable_evm_chains
.push(evm_chain_id);
Ok(())
}
```

If that happens, removing support for that chain would require doing it twice, which won't be mentioned by any warning or error. As a result, the chain to be removed may still remain in use. Below code of **disable_evm_chain.rs** shows that it just removes first iteration of requested ID, assuming that it's unique.

```
pub fn handle(ctx: Context<DisableEvmChain>, evm_chain_id: u32) -> R
esult<()> {
let index = ctx
.accounts
.bridge_info
.enable_evm_chains
.iter()
.position(|&r| r == evm_chain_id)
.ok_or(BridgeError::EvmChainNotFound)?;

ctx.accounts.bridge_info.enable_evm_chains.remove(index);
Ok(())
}
```

Assets:

- instructions/disable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/enable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Fixed

Classification

Impact: 2/5

Likelihood: 1/5

Exploitability: Dependent

Complexity: Simple

Severity: Low

Recommendations

Remediation: On adding new enabled chain, make sure to check if it is not already added, and if so, revert.

Resolution: The issue was fixed in commit **c50984** by checking if EVM chains vector already contains the chain to be added and revert if true.

Observation Details

F-2024-2399 - Vulnerable dependencies in use - Info

Description:

As a result of cargo-audit audit command, it was detected that one of project dependencies contain known vulnerabilities. The vulnerable crate is listed below along with additional informaion:

```
Crate: ed25519-dalek
Version: 1.0.1
Title: Double Public Key Signing Function Oracle Attack on `ed25519-dalek`
Date: 2022-06-11
ID: RUSTSEC-2022-0093
URL: https://rustsec.org/advisories/RUSTSEC-2022-0093
Solution: Upgrade to >=2
```

As of now it might be that this crate is part of solana which rules out possibility of patching it alone. On the other hand, there is no known, direct risk related to exploitation of that issue. Therefore, while currently there is no possibility of applying a direct patch, it is worth doing so as soon as it becomes possible.

Status: Accepted

Recommendations

Remediation:

Upgrade the vulnerable crate to version >=2 as per the aforementioned cargo-audit result, when it will become possible.

Resolution:

A patch to this finding was not provided in the commit **c50984**. Also, as per the finding description, most likely the patch should come from Solana crate, therefore mitigation is out of scope for the protocol.

F-2024-2401 - Missing events emission - Info

Description: During the review it was discovered that the protocol does not emit events at all - that means the issue concerns the protocol globally.

Logging events on key state change is an important operation that increases transparency and trackability on the blockchain, there fore it is encouraged to log a suitable event for each key state change such as: modification of key parameters, deposit or withdraw.

Assets:

- lib.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/deposit_to_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/disable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/enable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/initialize.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_authority.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_status.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_withdraw_signer_one.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/update_withdraw_signer_two.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/withdraw_from_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Accepted

Recommendations

Remediation: Implement events and emit them on each state change and especially key operations such as deposit or withdraw.

Resolution: A patch to this finding was not provided in the commit **c50984**.

F-2024-2461 - Arithmetic operations may panic on overflow - Info

Description:

Arithmetic operations used in the code use `unwrap()` to handle potential overflows. That means, if an overflow occurs, there is no information what happened and instead, the program will just panic.

Below occurence comes from **withdraw_from_chain.rs,** line 67

```
let amount_to_withdraw = total_deposits.checked_sub(chain_info.total
_withdrawals_from_chain).unwrap();
```

Assets:

- instructions/withdraw_from_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Accepted

Recommendations

Remediation: Use a custom error to handle potential overflows and display more verbose error message, instead of handling the overflow with potential panic.

Resolution: A patch to this finding was not provided in the commit **c50984**.

F-2024-2463 - Insufficient validation of mint account during initialization - Info

Description:

During the initialization (**Initialize** instruction) of the Solana program, it was noted that the mint account undergoes inadequate validation checks. This oversight could potentially allow unauthorized actions or the exploitation of token mechanics, thereby compromising the integrity and expected functionality of the token system. Specifically, the program fails to ensure that:

1. The **mint_authority** is revoked, which should ideally prevent the minting of new tokens.
2. The **freeze_authority** is deactivated, which is necessary to prohibit freezing of tokens on user accounts.
3. Extensions that could affect token transactions (e.g., fees on transfer or transfer hooks) are disabled.
4. The **mint_decimals** setting conforms to the expected value to maintain consistency in token transactions.

Below snippet shows fragment of **initialize.rs** file:

```
#[derive(Accounts)]
pub struct Initialize<'info> {
#[account(
init,
payer = authority,
space = BRIDGE_INFO_INIT_SPACE,
seeds = [&"bridge_info".as_bytes()],
bump
)]
pub bridge_info: Account<'info, BridgeInfo>,

pub mint: Account<'info, Mint>,
```

These shortcomings could result in deviations from the anticipated protocol business flow and may expose the system to various security risks.

Assets:

- instructions/initialize.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Accepted

Recommendations

Remediation:

Implement comprehensive validation checks for the mint account during the initialization phase. Ensure that:

- The `mint_authority` is effectively revoked to prevent unauthorized token minting.

- The `freeze_authority` is nullified to disable the freezing of tokens.
- No potentially harmful extensions are active.
- The `decimals` parameter is vesrified against the expected value.

Resolution: A patch to this finding was not provided in the commit **c50984**.

F-2024-2464 - Unbounded loops may lead to denial of service - Info

Description:

The enabled EVM chains are stored in a vector, that is being iterated on when checking if a chain is supported. While there is a finite, rather low amount of possible EVM chains that can be added, it should be kept in mind that such an approach is not suitable for larger amounts of data, as iterating over a large vector is not Gas efficient and in edge cases may lead to Denial of Service.

Since the chain support is an admin-only feature, and there are only a few possible chains in existence (assuming duplicates won't be added), there is little risk related to potential Denial of Service, therefore this issue is informational only.

The below snippet from **enable_evm_chains.rs** shows the related code:

```
pub fn handle(ctx: Context<EnableEvmChain>, evm_chain_id: u32) -> Result<()> {
ctx.accounts
.bridge_info
.enable_evm_chains
.push(evm_chain_id);
Ok(())
}
```

Assets:

- instructions/disable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]
- instructions/enable_evm_chain.rs [https://github.com/sevenlabs-hq/hellolabs-token-bridge-program]

Status: Accepted

Recommendations

Remediation:

It is recommended to avoid storing whitelisted chains in vectors and generate appropriate PDA with account structure for whitelisted chains instead.

Resolution:

A patch to this finding was not provided in the commit **c50984**.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/sevenlabs-hq/hellolabs-token-bridge-program
Commit	2d7ba697864f8770e8ea431312883d1c9d7cf200
Whitepaper	N/A
Requirements	N/A
Technical Requirements	N/A

Files in Scope

errors.rs

lib.rs

state.rs

instructions/deposit_to_chain.rs

instructions/disable_evm_chain.rs

instructions/enable_evm_chain.rs

instructions/initialize.rs

instructions/mod.rs

instructions/update_authority.rs

instructions/update_status.rs

instructions/update_withdraw_signer_one.rs

instructions/update_withdraw_signer_two.rs

instructions/withdraw_from_chain.rs