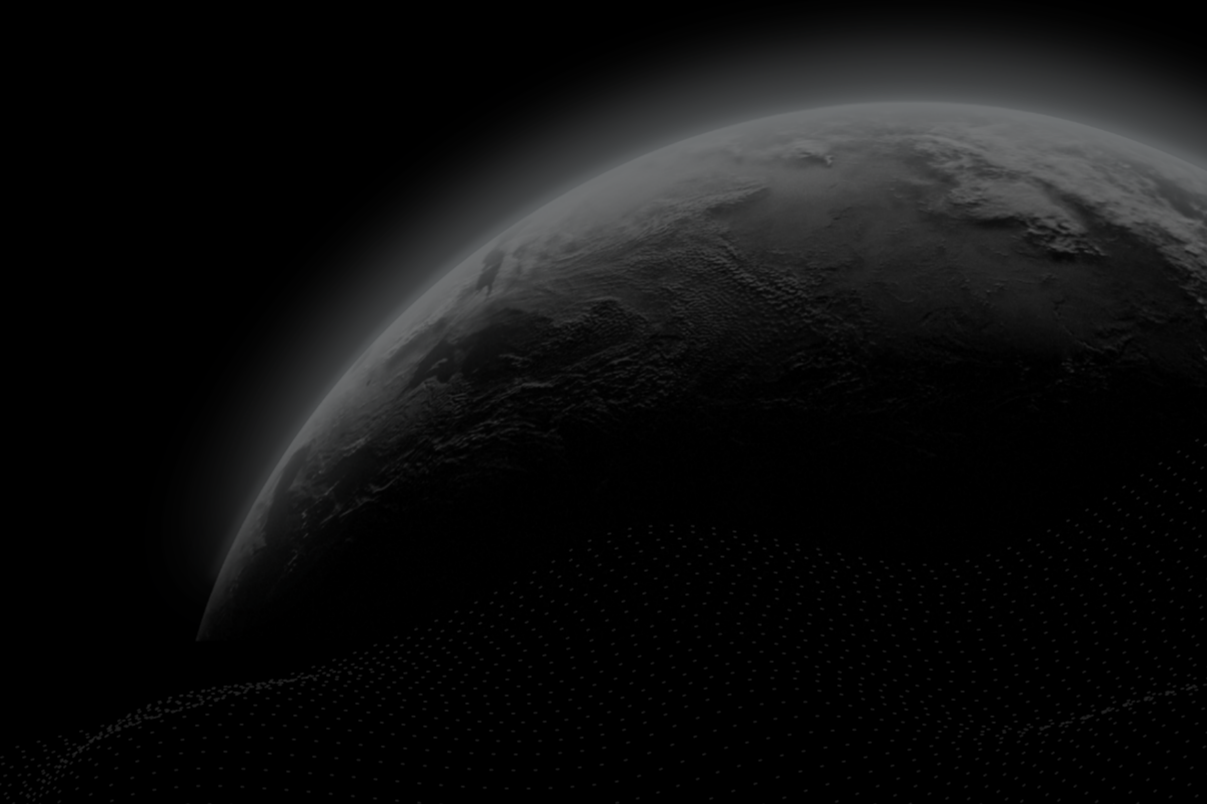
CERTIK

Security Assessment

XP Network - Dfinity Integration

CertiK Verified on May 8th, 2023



CertiK Verified on May 8th, 2023

XP Network - Dfinity Integration

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Platform	Other	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Rust	Delivered on 05/08/2023	N/A

CODEBASE
https://github.com/XP-NETWORK/dfinity-integration
...View All

COMMITS
cbe0df3b82025f3246f282c32d5f77167b73ae62
...View All

Vulnerability Summary

8	6	0	0	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	0	Critical		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.
🟧	3	Major	1 Resolved, 2 Acknowledged	Major 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.
🟨	3	Medium	3 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	0	Minor		Minor 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.
🟦	2	Informational	2 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | XP NETWORK - DFINITY INTEGRATION

CODEBASE | XP NETWORK - DFINITY INTEGRATION

Repository

https://github.com/XP-NETWORK/dfinity-integration

Commit

cbe0df3b82025f3246f282c32d5f77167b73ae62

AUDIT SCOPE | XP NETWORK - DFINITY INTEGRATION

4 files audited ● 1 file with Acknowledged findings ● 3 files without findings

ID	File	SHA256 Checksum
● XPT	src/minter/src/lib.rs	db4b6ff71655fcc65184a37725ce07cff433002d66f19a43a1abfed377196d83
● XPN	src/minter/src/actions.rs	826f4c3623f153078cdfa57fec22a30495abf55bce9a1201534a726af250659a
● XPE	src/minter/src/events.rs	8f5e4e61ecd3ef776e0d183c43e8f18fb2ce918b9a9ec49a1c42968b29399d61
● XPW	src/minter/src/types.rs	3a97f84cd732286af51834001d860098eb286413482719665cae85001f361e5d

APPROACH & METHODS | XP NETWORK - DFINITY INTEGRATION

This report has been prepared for XP Network to discover issues and vulnerabilities in the source code of the XP Network - Dfinity Integration project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | XP NETWORK - DFINITY INTEGRATION

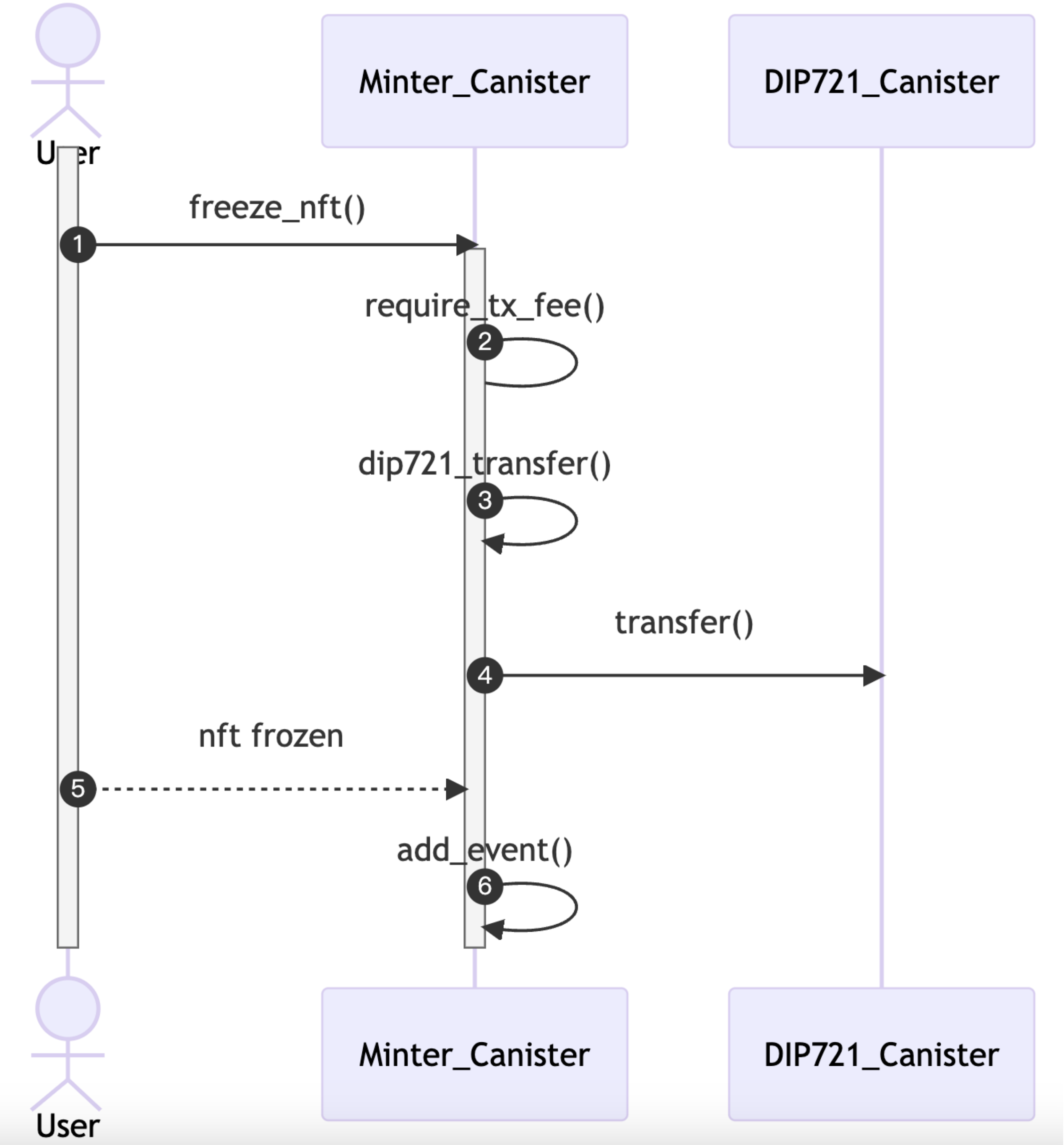
The XP Network Protocol is an ecosystem of cross-chain bridges that provides software tools to support developers in quickly constructing their own cross-chain bridge smart contract programs. XP Network supports both EVM and non-EVM protocols.

The XP Network protocol encompasses the XP.Network Bridge and the XP.Network token. The XP.Network relay validators, a part of the Bridge component, adopt a BFT consensus algorithm to ensure the reliability of cross-chain transactions.

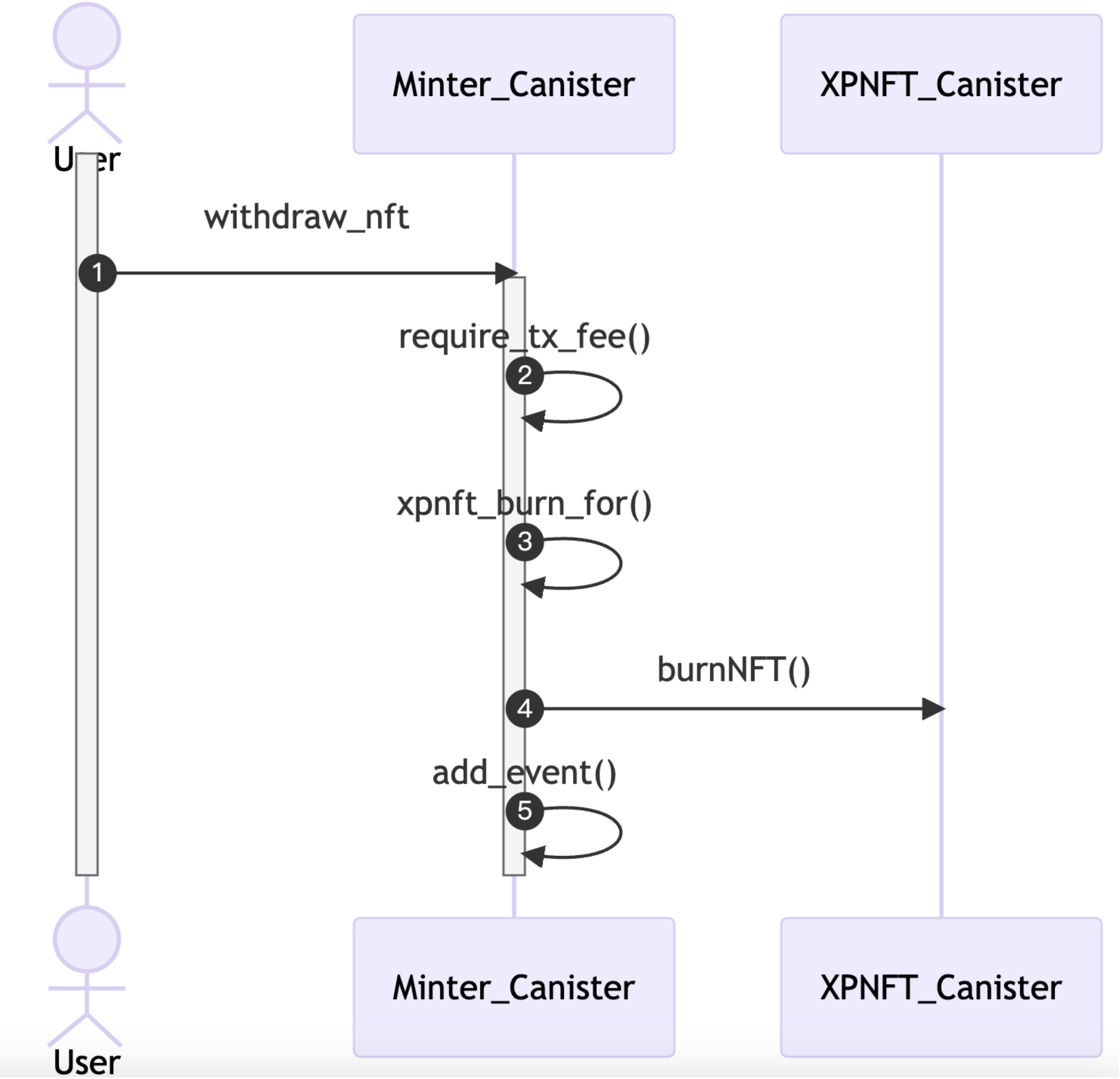
The XP Network Bridge uses an event-driven communication mechanism between two blockchains. When the relay validators detect a NFT transfer event on the source chain, one of them sends the signed message to the target chain to execute the specified operation.

In this audit report, the scope is the multi-chain bridge smart contract, `Minter` , on the IC blockchain. We assume the outside factors like validators are safely implemented.

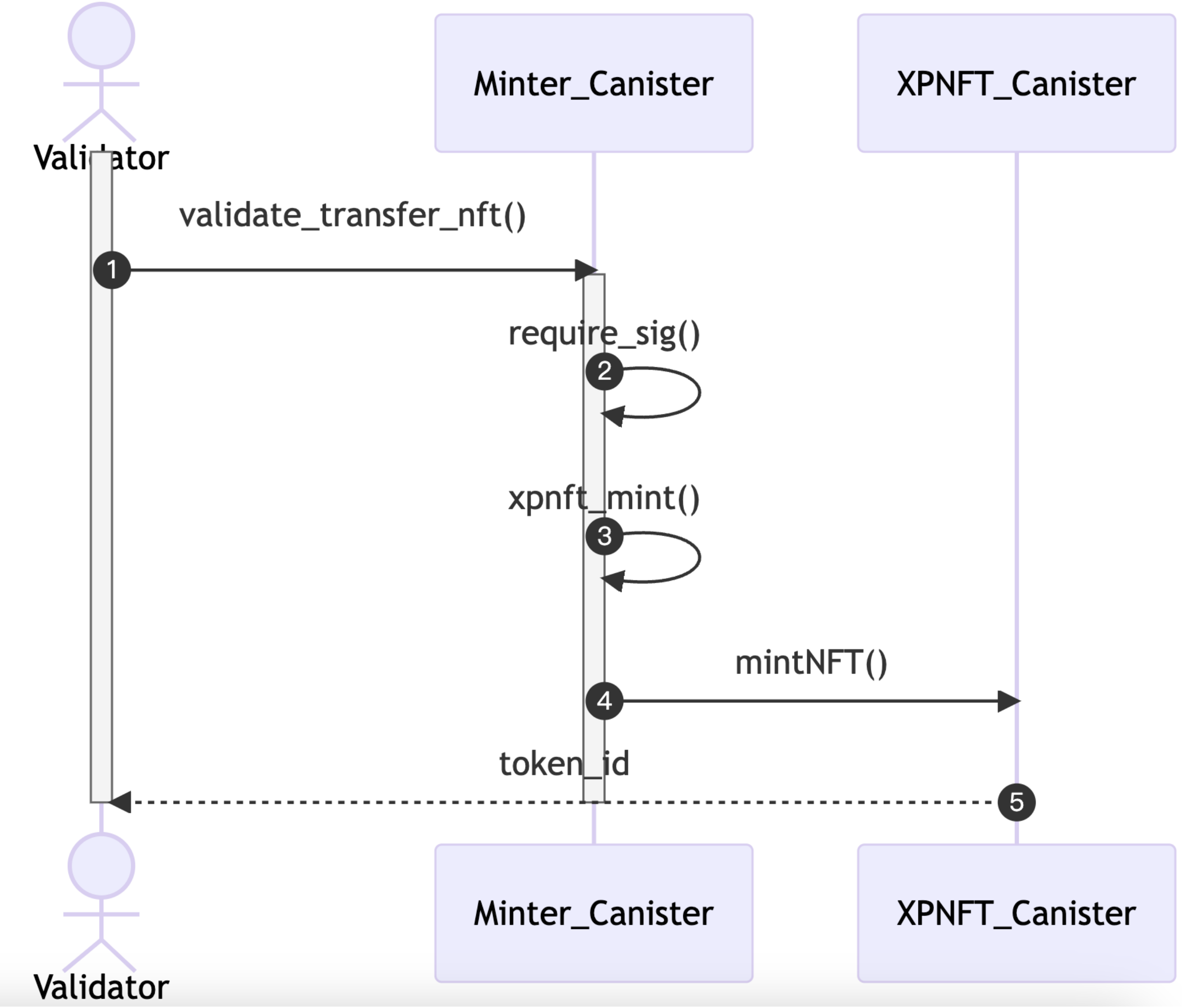
Users Freeze NFT



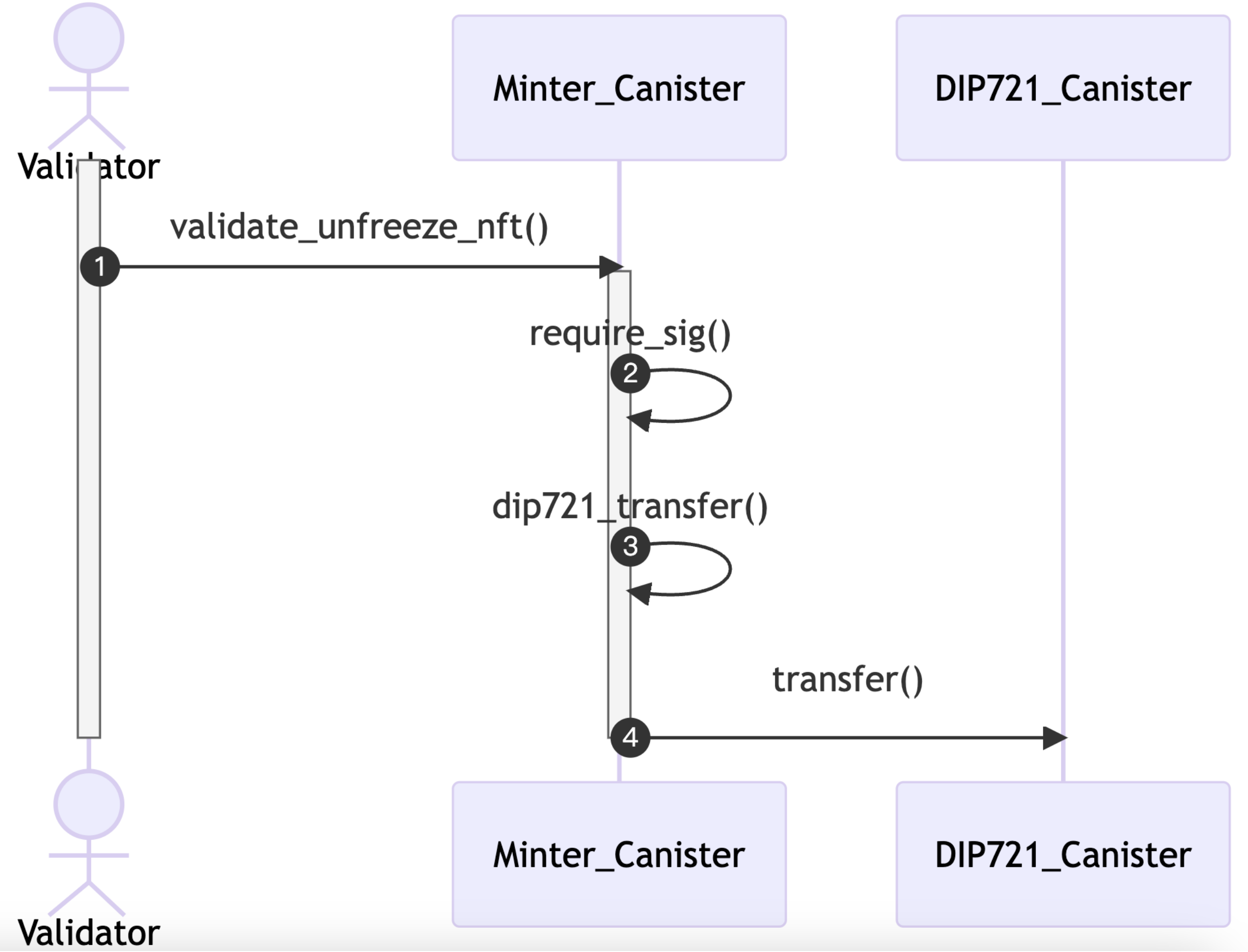
Users Unfreeze NFT



Validator Mints NFT



Validator Refunds NFT


Validator
Minter_Canister
DIP721_Canister
validate_unfreeze_nft()
1
require_sig()
2
dip721_transfer()
3
transfer()
4
Validator
Minter_Canister
DIP721_Canister

FINDINGS | XP NETWORK - DFINITY INTEGRATION

8	0	3	3	0	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for XP Network - Dfinity Integration. Through this audit, we have uncovered 8 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
XPT-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
XPT-02	Potential Loss Of NFT	Logical Issue	Major	Resolved
XPT-14	**Canister Upgrade Centralization Risk**	**Centralization / Privilege**	**Major**	**Acknowledged**
XPT-03	Potential Not Intended Looping	Control Flow	Medium	Resolved
XPT-04	Potential ICP Transfer Failure	Logical Issue	Medium	Resolved
XPT-13	Lacks Of Storing `fee_block` Into `FEEBLOCK_STORE`	Logical Issue	Medium	Resolved
XPT-08	Missing Check For Vectors Length Equality	Logical Issue	Informational	Resolved
XPT-09	Duplicate `unsafe` Block Code	Volatile Code	Informational	Resolved

XPT-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**src/minter/src/lib.rs: 325, 333, 344, 383, 395, 411, 431, 447, 470**	● **Acknowledged**

Description

In the `lib` contract, the `validator` role and accounts with the validator's signature have authority over the following functions:

- set_pause(): set the contract state to pause, and other functions that cause contract data to be updated will not be executed;
- set_group_key(): change the public key for canister signature verification;
- withdraw_fees(): withdraw all ICP assets in the canister to any account;
- add_whitelist(): add any NFT contract to the whitelist for cross-chain transfers;
- clean_logs(): remove event logs;
- validate_transfer_nft(): validator verifies the event and then mints NFT;
- validate_unfreeze_nft(): validator verifies event and then unlocks NFT;
- validate_transfer_nft_batch(): validator verifies event and then mints NFT in batch;
- validate_unfreeze_nft_batch(): validator verifies event and then unlocks NFT in batch;

Any compromise to the accounts with the signature may allow a hacker to take advantage of this authority, replace the public key, and transfer NFTs to anyone.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We recommend carefully managing the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

- Passing canister control to a decentralized governance system such as the Internet Computer's Service Nervous System (SNS), so that changes to the canister are only executed if the SNS community approves them collectively through voting.
- Implement a DAO on the IC from scratch. Furthermore, users will need to verify that the DAO is controlled by itself.
- Create an immutable canister smart contract by removing the canister controller completely. However, note that this implies that the canister cannot be upgraded.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

`[CertiK]` :
A new privileged function was added in the commit hash: bd2840ceb79288fa44ac1138adfcf781c2b85d42.

In the `lib` contract, the `validator` role and accounts with the validator's signature have authority over the function:

- set_fee_group_key(): change the public key for fee signature verification;

`[XP-Network]` :
The described functions are called by a network of validators. It requires 2/3 of their votes for a function to succeed. FROST algorithm built over Schnorr signature is used to generate a threshold signature which is then verified on-chain by the bridge smart contract.

So, we consider it to be handled by the bridge components external to the contract.

XPT-02 | POTENTIAL LOSS OF NFT

Category	Severity	Location	Status
Logical Issue	● Major	src/minter/src/lib.rs: 582~588	● Resolved

Description

The `withdraw_nft()` function, which is defined as an external function accessible by any account on the ICP chain, unlocks the source NFT which was locked on an external chain by burning wrapped NFT on the ICP chain.

However, its implementation lacks verification codes for amount of NFT transaction fee and ownership of wrapped NFT, allowing malicious users to call the `withdraw_nft()` function with fake arguments and obtain the source NFT on the external chain for only the ICP transfer fee.

Scenario

1. Bob on ICP accesses the `xpnft` contract and lists the wrapped NFT that belongs to Alice. Alice has a Wrapped NFT, token_id = 10, that mapping a NFT on BSC.
2. Bob on ICP pays for the zero to `minter` contract, and the fee block index is, for example, 10000, which doesn't exist in `minter` contract.
3. Bob on ICP calls the `withdraw_nft()` function with the following arguments: `tx_fee_block=10000, burner=Wrapped NFT, token_id=10, chain_nonce=4, to=Bob_on_BSC` .
4. Now, because no NFT transaction fee validation exists, the function `require_tx_fee()` executes successfully.
5. And the code at L602 will call the function `burnNFT()` defined in `xpnft.mo` . There is no code logic for checking the burned wrapped NFT's ownership; Alice's Wrapped NFT is burned successfully by Bob on ICP.
6. The validators detect the wrapped NFT burn event and unlock Alice's NFT to Bob on BSC.
7. As a result, Bob can only pay the ICP transfer fee to get Alice's NFT on BSC.

Recommendation

We recommend modifying the code to prevent any ICP user from calling the `withdraw_nft()` function with fake arguments. Potential solution is:

1. validating the NFT cross-chain transaction fee;
2. verifying that the caller of the `withdraw_nft()` function is the owner of the `xpnft` .

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 44d080bb369f8e4e1984beb315c34c4c386ac73f.

XPT-14 | CANISTER UPGRADE CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**src/minter/src/lib.rs**	● **Acknowledged**

Description

Canister smart contracts are deployed and managed by controllers. Among other capabilities, the controllers can update the canister's settings, install/upgrade the running code or even delete the canister. Changing the code for the canisters they control so canister code is mutable, unlike smart contracts on other blockchains and the controllers have complete control over the assets like ICP tokens or Bitcoins held by the canister they manage. This feature brings canisters closer to typical software and makes them suitable for a broad range of applications where software logic can be changed on an as-needed basis.

For critical applications like those used in DeFI, mutability can be dangerous; the controller could change a benign canister into a canister that steals assets.

Recommendation

We recommend to ensure that the canister is immutable or has decentralized governance.

Alleviation

[`XPNetwork`]:

We are using off-chain FROST: Flexible Round-Optimized Schnorr Threshold Signatures (https://eprint.iacr.org/2020/852.pdf).

It does not have a smart contract address - it is a Schorr-based multisignature verifiable on-chain.

The bridge uses the FROST group key, which can be used like the Schnorr public key for multi-signature verification. However, only if at least the threshold of validators submit their share of signatures will it be valid and result in the `group_key` during verification.

XPT-03 | POTENTIAL NOT INTENDED LOOPING

Category	Severity	Location	Status
Control Flow	● Medium	src/minter/src/lib.rs: 401	● Resolved

Description

There will be a potential infinite loop for method `clean_logs()` in the `minter` canister.
In that method, there is no validation for input parameter `action`.
If `action.from_action` is bigger than `action.to_action`, it will case the not intended looping at L401.

```
401          while action.from_action != action.to_action {
402              bmap.remove(&action.from_action);
403              action.from_action += Nat::from(1u32);
404          }
```

Recommendation

We recommend refactoring the code to prevent such infinite loop, such as adding validation check for the input parameter `action` or remediating the code at L401 as below:

```
401      while action.from_action <= action.to_action {
```

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 44d080bb369f8e4e1984beb315c34c4c386ac73f.

XPT-04 | POTENTIAL ICP TRANSFER FAILURE

Category	Severity	Location	Status
Logical Issue	● Medium	src/minter/src/lib.rs: 352~375	● Resolved

Description

In the execution of a transfer operation within the Ledger Canister, a fee must be paid, and it will check if the source account holds a sufficient amount of ICP to cover the transfer amount plus the fee.

In the rust code, the logic is to withdraw all the ICP balance from the `minter` contract, and the `amount` passed to the `transfer` method is equal to the total ICP balance, with a standard fee of 10^-4 ICP. The sum of the `amount` and `fee` will exceed the amount of ICP held by the `minter` contract, resulting in the transfer failing. The actual transaction amount should be `bal- fee` .

Recommendation

We recommend that users modify the code to prevent the `withdraw_fees()` function call from failing due to an insufficient ICP balance in the `minter` contract. A potential solution is to change the `amount` parameter value on line 364 to `bal - DEFAULT_FEE` .

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 44d080bb369f8e4e1984beb315c34c4c386ac73f.

XPT-13 | LACKS OF STORING `fee_block` INTO `FEEBLOCK_STORE`

Category	Severity	Location	Status
Logical Issue	● Medium	src/minter/src/lib.rs: 203	● Resolved

Description

From a design perspective, the `FEEBLOCK_STORE` variable was created to prevent duplicate fee_block submissions.

```
async fn require_tx_fee(
    canister_id: &Principal,
    caller: &Principal,
    fee_block: BlockIndex,
) -> Result<u64, BridgeError> {
    if FEEBLOCK_STORE.with(|store| store.borrow().contains(&fee_block)) {
        return Err(BridgeError::InvalidFee);
    }
    ...
}
```

Since the function does not store `fee_block` into `FEEBLOCK_STORE`, `FEEBLOCK_STORE` is always empty, which means that this check is meaningless. If the fee oracle can guarantee that the incoming `fee_block` will never be repeated, then there is no problem. But we think the best way to deal with it is to store the verified fee_block in `FEEBLOCK_STORE`. In this way, every time a new `fee_block` comes in, the requirement can check whether it already exists in the `FEEBLOCK_STORE`, thus ensuring that the same `fee_block` is not submitted repeatedly.

Recommendation

We recommend storing `fee_block` into `FEEBLOCK_STORE`.

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 3b65b44a7aa4690acdbdce01706a0dfdc6ce6d0f.

XPT-08 | MISSING CHECK FOR VECTORS LENGTH EQUALITY

Category	Severity	Location	Status
Logical Issue	● Informational	src/minter/src/lib.rs: 462~463, 486~487	● Resolved

Description

Loop iterates over `action.token_urls[i]` vector (i=0... size of token_urls) and accesses `action.mint_with[i]` vector without asserting that `action.token_urls` and `action.mint_with` have equal lengths.

```
    for (i, token_url) in action.token_urls.into_iter().enumerate() {
        xpnft_mint(action.mint_with[i], token_url, action.to)
```

If the length of `action.mint_with` is less than `action.token_urls` , a panic will be caused due to the subscript being out of bounds.

The similar issue also exists in the function `validate_unfreeze_nft_batch()` .

Recommendation

We recommend adding a check to ensure that the two vectors have equal length. For example:

```
assert_eq!(action.mint_with.len(), action.token_urls.len());
```

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 44d080bb369f8e4e1984beb315c34c4c386ac73f.

XPT-09 | DUPLICATE `unsafe` BLOCK CODE

Category	Severity	Location	Status
Volatile Code	● Informational	src/minter/src/lib.rs: 58, 117	● Resolved

Description

The `unsafe` block in Line 58 is duplicate and could be replaced by calling function `config_ref()`.

```
58          let conf = unsafe { CONFIG.as_ref().unwrap() };
```

Recommendation

We recommend refactor the code in Line 58 by using `config_ref()`.

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit 44d080bb369f8e4e1984beb315c34c4c386ac73f.

OPTIMIZATIONS | XP NETWORK - DFINITY INTEGRATION

ID	Title	Category	Severity	Status
XPT-10	Missing Input Validation	Logical Issue	Optimization	● Resolved
XPT-11	Unnecessary Borrow	Language Specific	Optimization	● Resolved
XPT-12	Redundant Clone	Language Specific	Optimization	● Resolved

XPT-10 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Logical Issue	● Optimization	src/minter/src/lib.rs: 395, 402	● Resolved

Description

The logic of `clean_logs()` function is to delete a specified `action` item. If the designated `action` number does not exist, the `clean_logs()` function will not throw an error, but it will waste Cycles.

To avoid the pointless consumption of Cycles, it is necessary to check the existence of the `action` number.

Recommendation

We recommend adding code to detect the existence of the `action` .

Alleviation

[`CertiK`]:

`XPNetwork` team modified the function `clean_logs` logic to delete all log information when calling it and solved this issue in commit 08eb7ce6164b2bd02abe9669f64408a0e36486bf

XPT-11 | UNNECESSARY BORROW

Category	Severity	Location	Status
Language Specific	● Optimization	src/minter/src/lib.rs: 207~208	● Resolved

Description

The above line creates new references, which are immediately dereferenced by the compiler.

```
207  let caller_acc = AccountIdentifier::new(&caller, &DEFAULT_SUBACCOUNT);
208  let canister_acc = AccountIdentifier::new(&canister_id, &DEFAULT_SUBACCOUNT);
```

However, since the `caller` and `canister_id` are already references of the `Principal` type, which meet the parameter type requirements of the function, there is no need to create new references again.

Recommendation

We recommend removing the references. For example:

```
207  let caller_acc = AccountIdentifier::new(caller, &DEFAULT_SUBACCOUNT);
208  let canister_acc = AccountIdentifier::new(canister_id, &DEFAULT_SUBACCOUNT);
```

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit f883024e8a825339eb922b31a0f49884523b170f.

XPT-12 | REDUNDANT CLONE

Category	Severity	Location	Status
Language Specific	● Optimization	src/minter/src/lib.rs: 152	● Resolved

Description

It is unnecessary to use `clone` as this value is dropped without further use.

```
152  let key = PublicKey::new(config_ref().group_key.clone());
```

Recommendation

We recommend removing the unnecessary `clone()` calls.

Alleviation

[`CertiK`]:

`XPNetwork` team heeded the advice and resolved the finding in the commit d4a3dd90c76d90cf0a1c0ab4c5d4042f848d60ec.

APPENDIX | XP NETWORK - DFINITY INTEGRATION

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as functions restricted to a privileged set of users.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as unintended deviations from the original business logic of the code base.
Control Flow	Control Flow findings refer to the access control imposed on functions, such as functions being callable by unauthorized users.
Volatile Code	Specifics may differ between runtime environment and (virtual) machine, however in principle findings indicate that assumptions that one may assume by reading code, may not hold, as there maybe other factors that may influence the state, which may lead to other issues (e.g. logical or control flow issues).
Language Specific	Language Specific findings are issues that would only arise within Rust, e.g., Needless borrow.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK 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. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.