Contract storage

Attacks related to the encrypted storage and query access to a contract

Partial Storage Rollback During Contract Runtime

The curren verification scheme can verify that when reading from a field in storage, the value received from the host has been written by the same contract instance to the same field in storage.

But it cannot (yet) verify that the value is the most recent value that was stored there. This means a malicious host can (offline) run a transaction, and then selectively provide outdated values for some fields of the storage. In the worst case, this causes a contract to expose old secrets with new permissions, or new secrets with old permissions.

The contract can protect against this by either (e.g.) making sure that pieces of information that have to be synced with each other are saved under the same field (so they are never observed as desynchronized) or (e.g.) somehow verify their validity when reading them from two separate fields of storage.

Two Contracts With The Samecontract_key Could Deanonymize Their States

If an attacker creates a contract with the samecontract_key as another contract, the state of the original contract can potentially be deanonymized.

Creating such a key is possible by instantiating the contract 2 times within the same block ascontract_key = creator + blockheight + codehash. Later upgrading the 2nd contract could retrieve even more information.

For example, an original contract with a permissioned getter, such that only whitelisted addresses can query the getter. In the malicious contract, the attacker can set themselves as the owner and decrypt the state of the original contract using a permissioned getter.

Last updated7 months ago On this page *Partial Storage Rollback During Contract Runtime* Two Contracts With The Same contract_key Could Deanonymize Their States

Was this helpful? Edit on GitHub Export as PDF