privacy-risks-of-gas_used)

Was this helpful? Edit on GitHub Export as PDF

Gas Evaporation & Tracking

The equivalent of padding for gas_used When a smart contract is executed, CosmWasm meters how much gas the execution consumes while the program is running. On Secret Network, this all happens within the enclave. However, once execution completes, the total amount of gas that was consumed leaves the enclave and is made public in transaction metadata.

Privacy Risks of gas_used

The publicgas_used result of a contract execution message reflects the sum of costs for the actual instructions that were executed. Within a typical smart contract, most of its methods consume different amounts of gas. With SNIP-20 for example, an attacker could deduce which method a transaction executed with a high degree of certainty;create_viewing_key vs.transfer vs.increase_allowance , each of which produce a distinctgas_used amount in the public transaction metadata.

In some cases, this can leak even more information such as which code path was taken, the value of a conditional, how many iterations a loop completed, and so on.

Gas Evaporation

Evaporation is a concept that was introduced to overcome the privacy risks associated withgas_used . Evaporation refers to the practice of deliberately consuming extra gas during execution in order to pad thegas_used amount before it leaves the enclave.

With evaporation, clients can now instruct contracts on exactly how much gas their execution should consume, yielding a consistentgas_used across all methods.

A contract can evaporate an arbitrary amount of gas using the built-in API functiongas_evaporate . In order to evaporate anyremaining gas, thecheck_gas API function allows contracts to calculate the difference between the amount consumed up to that point, and the amount the client wants the contract to consume by the end of its execution:

```

Copy ExecuteMsg::UseExact{ amount }=>{ letgas_used:u64=deps.api.check_gas()?; letto_evaporate=amount-gas_usedasu32; deps.api.gas_evaporate(to_evaporate)?; Ok(Response::default()) }

```

See the evaporation example contract'ssource code for more details.

Note that there is no API function to get the amount of actual gas remaining. Instead, that information must come from the client. This practice ensures that users are still able to execute multiple messages in a single transaction by specifying gas limits to each contract execution individually.

Gas Tracking

The newcheck_gas API function essentially gives contracts read access to CosmWasm's internal gas metering. Calling this function more than once during execution allows contracts to track changes in the amount of gas consumed. This simple feature enables a realm of interesting use cases not previously possible.

In its simplest form, gas tracking can be used as a development aid to inspect the amount of gas used by certain blocks of code, giving developers more insight and help with optimizing their gas footprint.

Opportunistic Execution

A more advanced use case involving gas tracking isopportunistic execution, where contracts take advantage of excess gas that would otherwise be evaporated by performing work that needs to be done anyway. That work could be anything from generic housecleaning duties, such as processing items in a queue, to more intentional duties, such as notarizing previous user actions to protect against tx-replay attacks.

To give an example, imagine an NFT auction contract that accepts bids on listed items. When a listing expires, the listed NFT should be transferred to the highest bidder. In typical contract design, the transfer would only actually happen once the winning bidder calls a method to "claim" their winnings. However, with opportunistic execution, the contract can automatically perform the transfer during any execution. For example, someone creating a viewing key could end up paying the gas fees to complete the transfer of a previous listing's NFT to its winning bidder.

Previous Limiting storage access data leaks Next Confidential Computing Layer Last updated7 months ago