ETM555

DESIGN of INFORMATION

SYSTEMS

ERC721 Token Based

Supply Chain Traceability

System

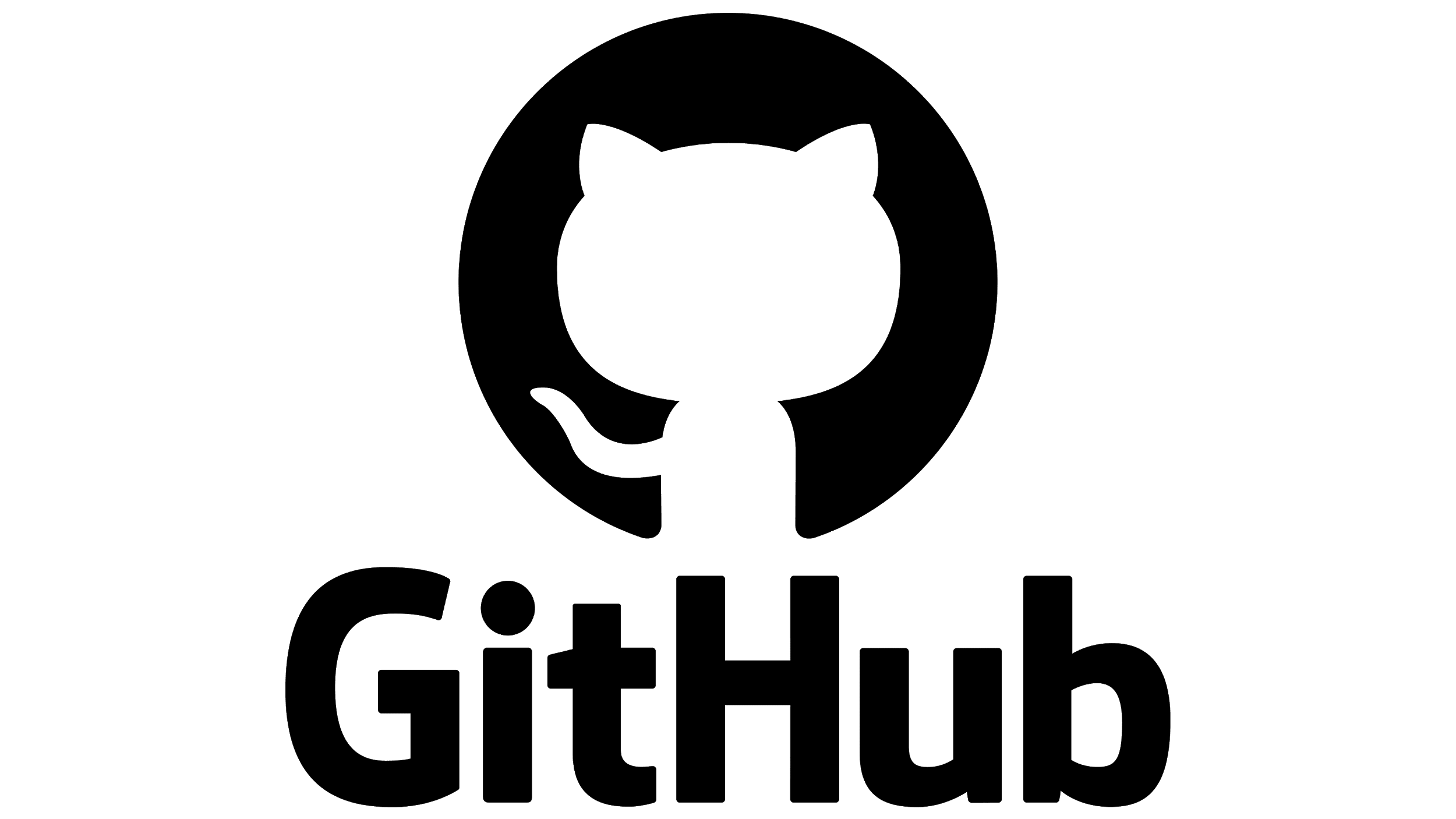
Instructor: Can Ozturan

Fall 2020/2021

Sevgican VAROL

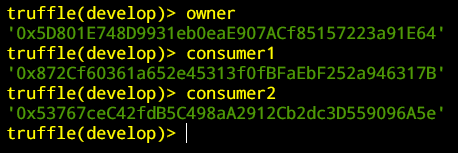
Gorkem ATES

Deniz MEMIS

[](https://github.com/d8niz/etm555-hw)

# Entities

* On truffle develop console we have the following three entities:
  + **Owner** (contract deployer)
  + **Consumer1** (simulating a consumer/intermediary)
  + **Consumer2** (simulating another consumer/intermediary)



# StateVerification Contract Interactions

* **StateVerification** contract is deployed and interacted as **‘stateAuthority’** object
* Created entities are approved by ‘stateAuthority’ using **‘insertVerifiedEntity**(address \_addr)’ function as follows:
  + This method has **isOwner**() modifier, allowing only the contract owner to add entities to the whitelist.
* **Owner** is whitelisted:



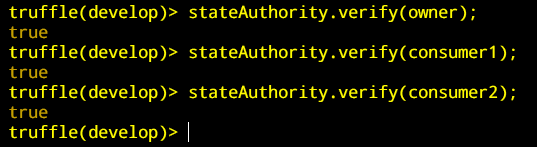
* **Consumer1** is whitelisted:



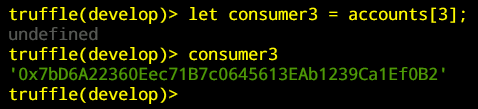
* **Consumer2** is whitelisted:



* Aforementioned entities are verified by **‘stateAuthority’** using ‘**verify**(address \_addr)’ function as follows:



* Another consumer/intermediary called Consumer3 is created as follows:



* A consumer could be removed from the whitelist using ‘**removeVerifiedEntity**(address \_address)’ function as follows:



# ProductProvenance Contract Interactions

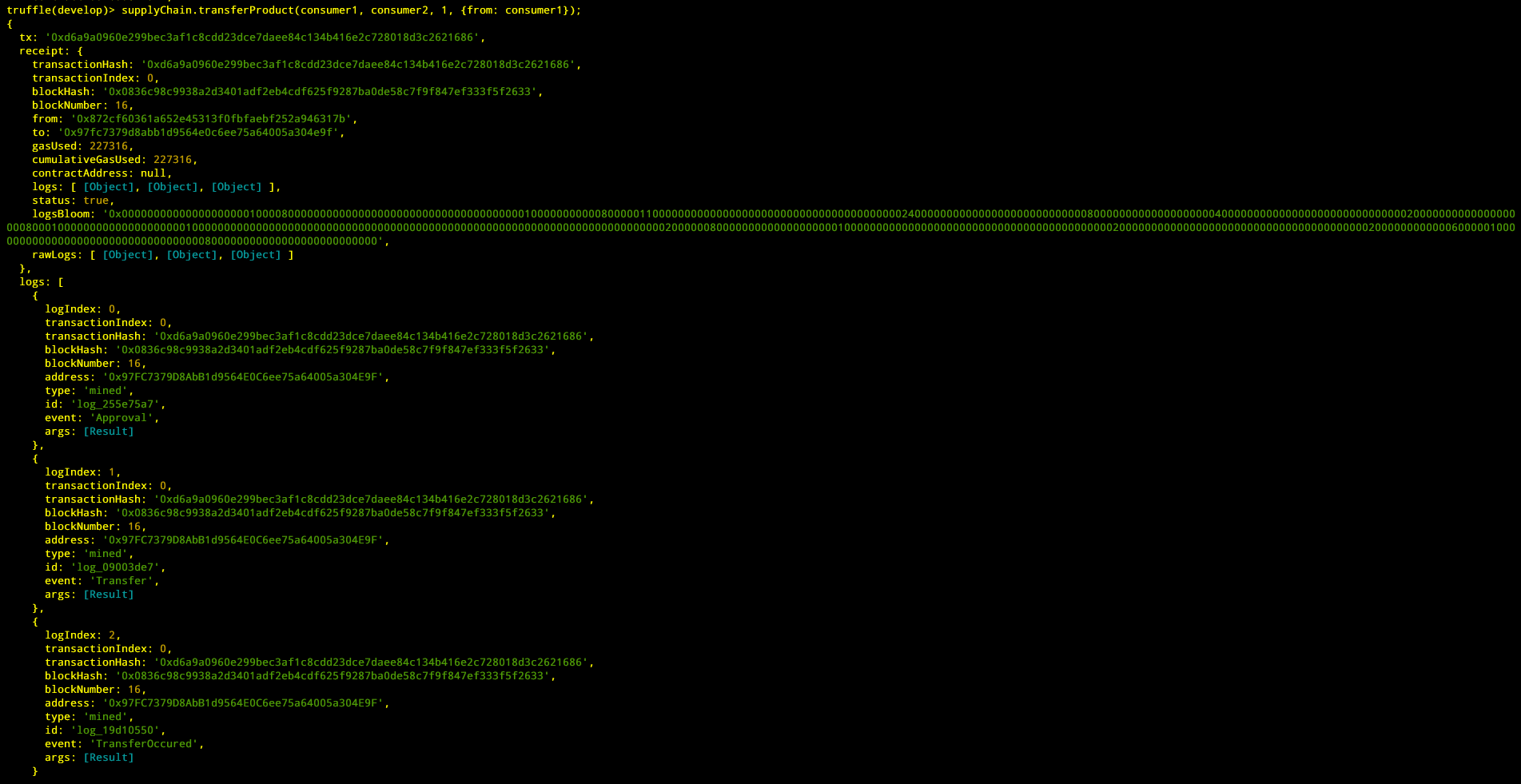
* **ProductProvenance** contract is deployed and can be interacted as **‘supplyChain**’
* A whitelisted consumer (i.e Consumer1, Consumer2) is allowed to mint a token/product.
  + This operation is done using **mintProduct**() function.
  + This function uses the built-in **\_mint**() function, and replicates its default behaviour.
  + Minted token is transferred to message sender from the contract owner (**Owner**)
* **Consumer1** minted a token:



* **TokenID** starts from zero and incremented at each minting operation. Hence one can trace product provenance by using **traceTokenHistory**(uint256 tokenId) function as follows:



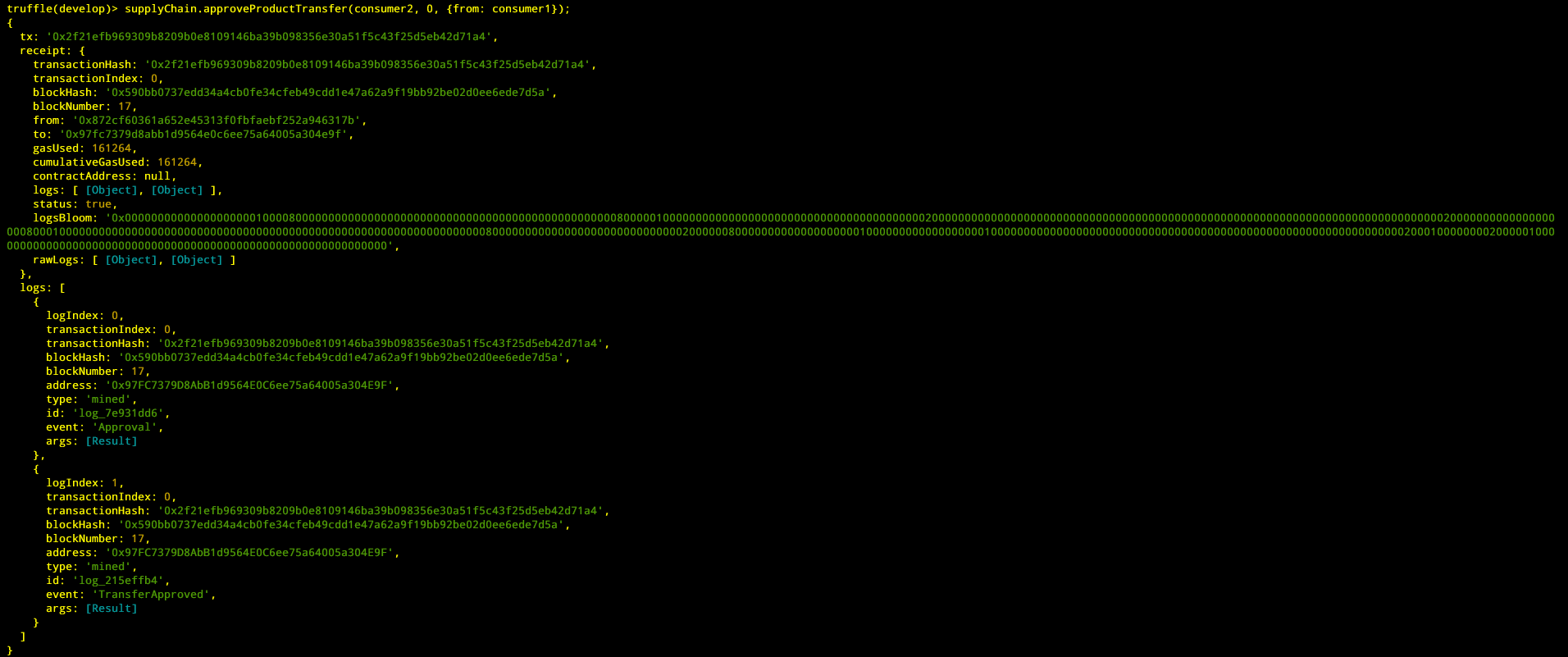
* To transfer a token/product,
  + **The token owner** can directly transfer using **transferProduct**( address \_addressFrom, address \_addressTo, uint256 tokenId ) function
  + Any other consumer should be approved by **the token owner** to transfer the token on behalf of the owner using **approveProductTransfer**(address \_addressTo, uint256 tokenId) function.
* **Consumer1 (as token owner)** can directly transfer the token to **Consumer2** as follows:



* Let’s trace the token history post-transfer, notice that **transfer is occured**:



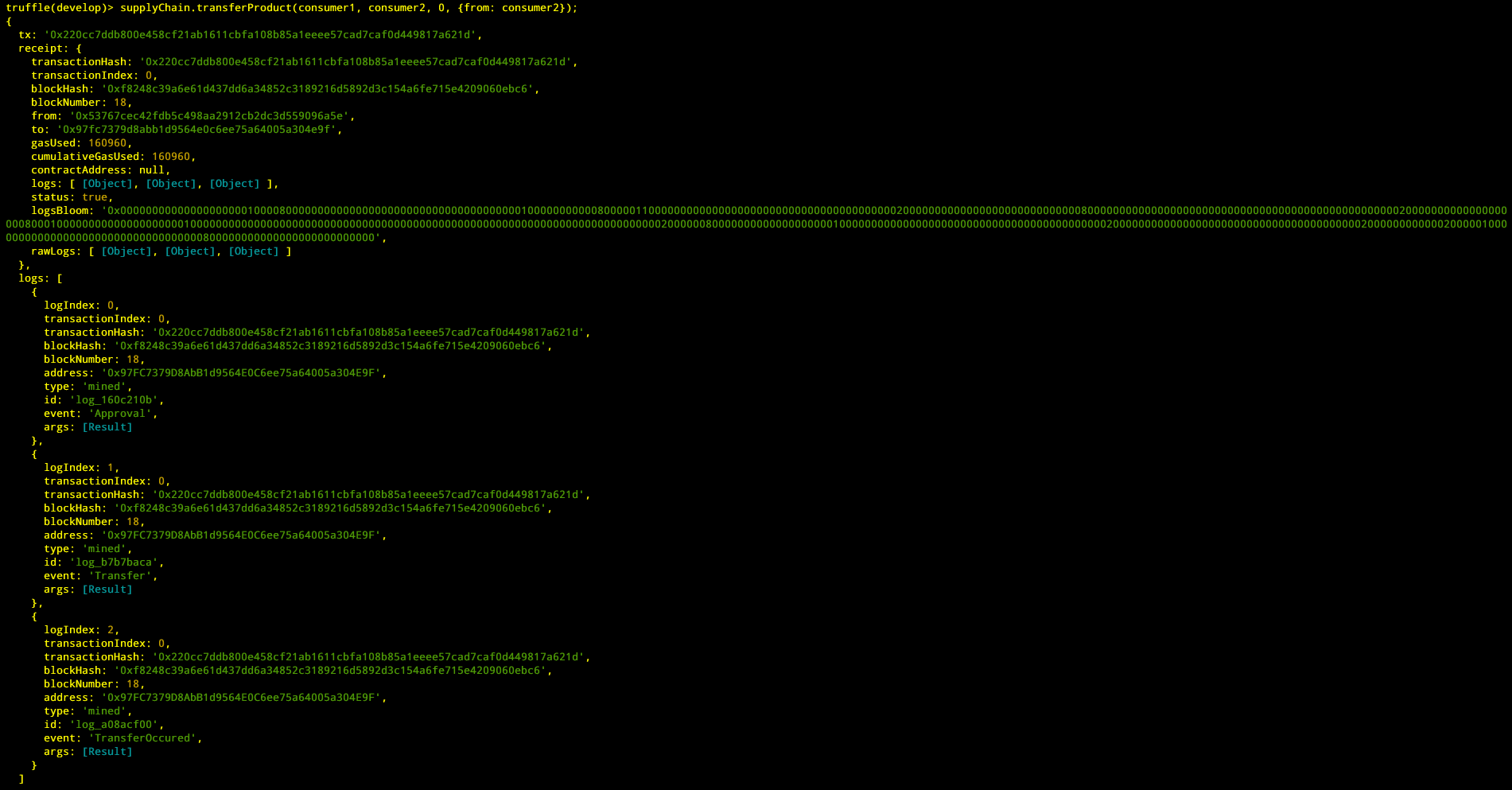
* **Consumer2** can transfer the token of **Consumer1** to himself, but first **Consumer1** should approve **Consumer2** for a specific token to be transferred:



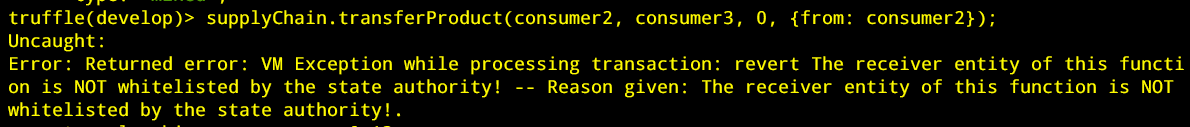
* Let’s trace the token history post-transfer-approval, notice that the **transfer is approved**:



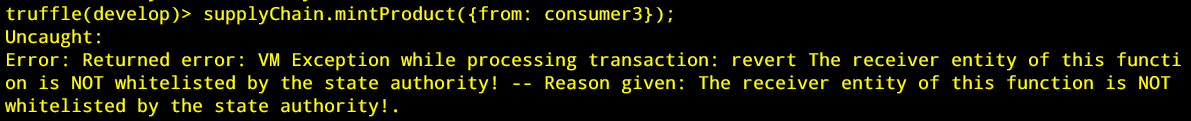
* **Consumer2** could now complete the transfer, since it is approved by **Consumer1**:



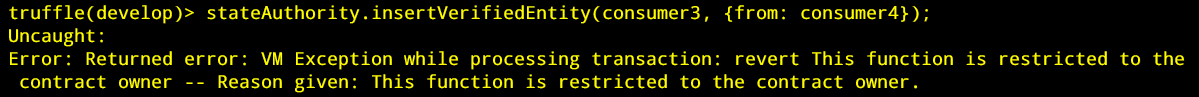
* Let’s trace the token history post-transfer-approval, notice that the **transfer is occured**:
* Let’s try to make a transfer to a non-whitelisted entity:
  + The receiver entity of this function is **NOT** whitelisted by the state authority!



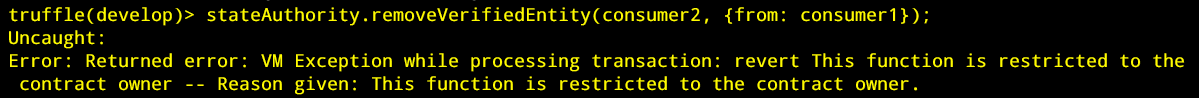
* Let’s try to mint a token as a non-whitelisted entity:
  + The receiver entity of this function is **NOT** whitelisted by the state authority!



* Let’s try to whitelist an entity as a consumer/intermediary:
  + This function is **restricted** to the contract owner!



* Let’s try to remove a whitelisted address from state registry as a **white-listed** consumer/intermediary:
  + This function is **restricted** to the contract owner!



* Let’s try to trace a token/product history as a non white-listed entity:
  + The caller entity of this function is **NOT** whitelisted by the state authority!

