BABYLON SMART CONTRACT AUDIT REPORT

17.01.2023

CONTENTS

SUMMARY

SEVERITY	NUMBER OF FINDINGS
CRITICAL	1
HIGH	1
MEDIUM	1
LOW	1
INFORMATIONAL	2

TOTAL: 6

SCOPE

The analyzed resources were sent in an archive with the following SHA256 hash:
608e66170227d3d643efbe8c41cb56bbac6d26143da996540ada cd2a572e151b

The issues described in the report were fixed in the following version (SHA256 hash):
538faea4f3b076576ac138e26152450213b9ca9203d770af83fd66 e456f315d5

WEAKNESSES

This section contains the list of discovered weaknesses.

1. STEALING ETH USING REENTRANCY IN MINTING PASS

SEVERITY: Critical

PATH: BabylonCore.sol:participate:L89-114

REMEDIATION: the call to mint should be done at the end of the function or a reentrancy guard should be added to the function

STATUS: fixed

DESCRIPTION:

In **BabylonCore.sol:participate** a user can buy tickets, which are minting pass NFTs and get minted to the user. The minting pass contracts use **_safeMint**, which makes an external call with **onERC721Received** to the receiver user and thus that execution is passed to the user upon minting.

As a result, a reentrancy vulnerability exists because there are state changing effects after the call to mint on L102, e.g. the current tickets total on L104 and the listing's state on L110.

This vulnerability can be abused to inflate listing's current tickets beyond the total tickets amount and refund/burn tickets without cancelling the listing. The creator could buy and refund a part of the tickets but still gain the full amount of **ETH** when the listing is finalized, netting a positive amount and thus, effectively, stealing **ETH**.

```
function participate(uint256 id, uint256 tickets) external payable {
  ListingInfo storage listing =  _listingInfos[id];
  require(
     _tokensController.checkApproval(listing.creator, listing.item),
     "BabylonCore: Token is no longer owned or approved to the controller"
  );
  require(listing.state == ListingState.Active, "BabylonCore: Listing state should be
active");
  require(block.timestamp >= listing.timeStart, "BabylonCore: Too early to participate");
  uint256 current = listing.currentTickets;
  require(current + tickets <= listing.totalTickets, "BabylonCore: no available tickets");
  uint256 totalPrice = listing.price * tickets;
  require(msg.value == totalPrice, "BabylonCore: msg.value doesn't match price for
tickets");

  IBabylonMintPass(listing.mintPass).mint(msg.sender, tickets);

  listing.currentTickets = current + tickets;

  emit NewParticipant(id, msg.sender, tickets);

  if (listing.currentTickets == listing.totalTickets) {
     listing.randomRequestId = _randomProvider.requestRandom(id);
     listing.state = ListingState.Resolving;

     emit ListingResolving(id, listing.randomRequestId);
  }
}
```

2. ANYONE CAN CREATE A LISTING FOR ANOTHER USER'S APPROVED/LISTED ERC721

SEVERITY: High

PATH: TokensController.sol:checkApproval:L47-61

REMEDIATION: see description

STATUS: fixed

DESCRIPTION:

A user can create a listing for an **ERC721** or **ERC1155** in the **BabylonCore** contract. This contract checks whether the controller is approved through **TokensController.sol:checkApproval** for the specified asset and if so, create the listing.

However, in case of **ERC721** the approval check fails to check whether the listing creator (**msg.sender**) is the owner, it only checks whether the controller is approved for the **NFT ID**.

As a result, anyone can create a listing for other users' **NFTs** that have been approved for the **TokensController** contract. Any ongoing listing for an **NFT** can therefore be recreated by any user because the **NFT** would be approved during the listing.

This would create fake listings where users might buy tickets, but the listing can never succeed because the **NFT** cannot be transferred from the malicious actor.

```
function checkApproval(
  address creator,
  IBabylonCore.ListingItem calldata item
) external view returns (bool) {
  if (item.itemType == IBabylonCore.ItemType.ERC721) {
     address operator = IERC721(item.token).getApproved(item.identifier);
     return address(this) == operator;
  } else if (item.itemType == IBabylonCore.ItemType.ERC1155) {
     bool approved = IERC1155(item.token).isApprovedForAll(creator, address(this));
     uint256 amount = IERC1155(item.token).balanceOf(creator, item.identifier);
     return (approved && (amount >= item.amount));
  }

  return false;
}
```

The approval check for **ERC721** should check whether the creator parameter is the owner of the asset.

For example:

```
if (item.itemType == IBabylonCore.ItemType.ERC721) {
  address operator = IERC721(item.token).getApproved(item.identifier);
  address owner = IERC721(item.token).ownerOf(item.identifier);
  return creator == owner && address(this) == operator;
}
```

3. USERS CAN RELIST THEIR TOKENS

SEVERITY: Medium

PATH: BabylonCore.sol:startListing:L54-87

REMEDIATION: add a check for require(_ids[user][token][identifier] == 0) so that a user could only list their token once

STATUS: fixed

DESCRIPTION:

When a user creates a listing for their token, the listing **ID** is stored in **_ids[token][identifier]**. However, this listing **ID** is never used or checked again, e.g. to check whether the user has already listed the token.

Furthermore, the token is never transferred into the **BabylonCore** contract, only an approval is required.

As a result, a user can relist their token any number of times. This will overwrite **_ids** and create a new and active listing every time in **_listingInfos**. Other users can buy tickets from any of those listings, while only one of them can be finalized. The others would get stuck.

```
function startListing(
  ListingItem calldata item,
  IEditionsExtension.EditionInfo calldata edition,
  uint256 timeStart,
  uint256 price,
  uint256 totalTickets,
  uint256 donationBps
) external {
  [..]
  uint256 newListingId = _lastListingId + 1;
  _ids[item.token][item.identifier] = newListingId;
  ListingInfo storage listing = _listingInfos[newListingId];
  [..]
}
```

4. MISSING UPPER LIMIT CHECKS

SEVERITY: Low

PATH: BabylonCore.sol:startListing:L62-68

REMEDIATION: add constant variables with upper limits and check these against the parameters in BabylonCore.sol:startListing

STATUS: acknowledged, addressed on the frontend

DESCRIPTION:

The function **BabylonCore.sol:startListing** doesn't check an upper limit for some variables.

For example:

- **timeStart** (creator can start a listing and indicate a timestamp for a very long period of time)
- **totalTickets** (creator can start a listing and indicate a very large number of tickets)
- **price** (creator can start listing and indicate a very high price for tickets)

As a result this would create fake listings that potentially can never succeed.

```
function startListing(
  ListingItem calldata item,
  IEditionsExtension.EditionInfo calldata edition,
  uint256 timeStart,
  uint256 price,
  uint256 totalTickets,
  uint256 donationBps
) external;
```

5. MISSING ZERO ADDRESS CHECKS

SEVERITY: Informational

PATH: TokensController.sol:setBabylonCore (L43-45), BabylonEditionsExtension.sol:setBabylonCore (L60-66), RandomProvider.sol:setBabylonCore (L46-48)

REMEDIATION: add a check to validate the parameter core against the zero address

For example:

require(core != address(0), "NOT_ZERO");

STATUS: fixed

DESCRIPTION:

In all contracts the parameter **core** for the function **setBabylonCore** is not checked against the zero address. If this address is set to it, then no one can call the functions **TokensController.sol:sendItem**, **BabylonEditionsExtension.sol:mintEdition**, **RandomProvider.sol:fulfillRandomWords** and **RandomProvider.sol:requestRandom**.

```
contract TokensController is ITokensController, Ownable {
  [..]
  function setBabylonCore(address core) external onlyOwner {
     _core = core;
  }
  [..]
}

contract RandomProvider is IRandomProvider, Ownable, VRFConsumerBaseV2 {
  [..]
  function setBabylonCore(IBabylonCore core) external onlyOwner {
     _core = core;
  }
  [..]
}

contract BabylonEditionsExtension is Ownable, IEditionsExtension,
ICreatorExtensionTokenURI {
  [..]
  function setBabylonCore(address core) external onlyOwner {
     _core = core;
  }
  [..]
}
```

6. MALICIOUS ACTOR CAN LIST FAKE TOKENS

SEVERITY: Informational

PATH: BabylonCore.sol:startListing:L54-87

REMEDIATION: add and check against a whitelist of ERC721/ERC1155 tokens that have been approved by trusted users

STATUS: acknowledged, addressed on the frontend

DESCRIPTION:

Any malicious actor can create a customized **ERC721/ERC1155** collection and create a listing for it. There are no checks against a whitelist of tokens to protect users from getting scammed.

```
function startListing(
  ListingItem calldata item,
  IEditionsExtension.EditionInfo calldata edition,
  uint256 timeStart,
  uint256 price,
  uint256 totalTickets,
  uint256 donationBps
) external;
```