9th of September 2022

Yoku Pay Security Report

A decentralized crypto payment processor for NFT Marketplaces

9th of September

Security Report Yoku Pay

Structure

1. General Notes

Yoku Pay is a multi-chain payment processor that allows customers to interact with smart contracts of decentralized NFT Marketplaces through multiple cryptocurrencies. The unifying goal is to improve the customer experience by lowering the hurdle for purchase and eliminating the stress of going to a third party outside the App. Yoku Pay strives to offer a service with an analogy to PayPal. It should be easy and swift for anyone to pay with their desired crypto on any decentralized NFT Marketplace.

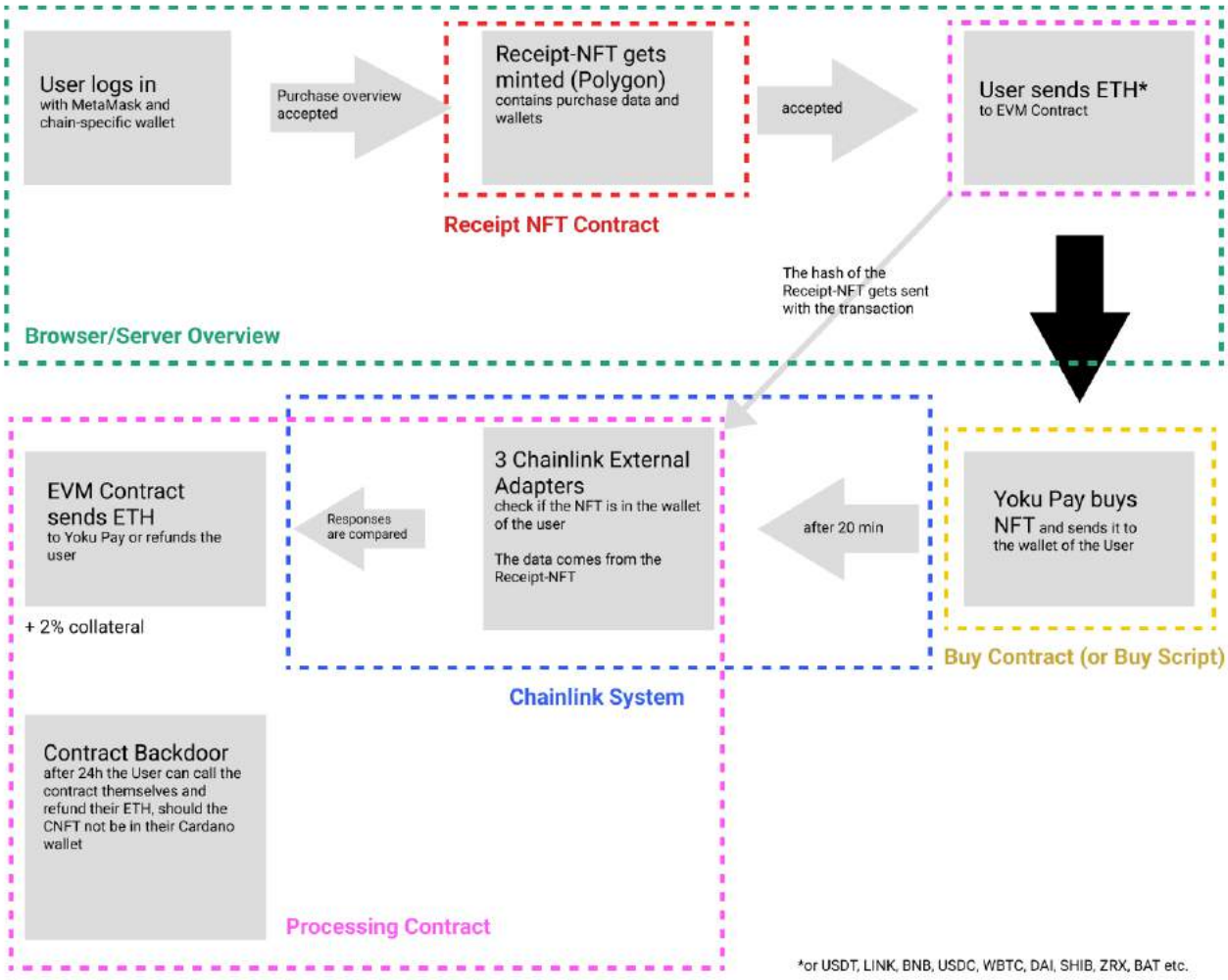
This Security Report aims to shine a light on the general state of security of the entire Yoku Pay system. Each Chapter is dedicated to an integral component of the whole system and will touch on the security concerns in everyday usage and problems that may come up in certain edge cases.

The structure of each chapter will be as follows:

Section	Explaination
GitHub Link	Links to the GitHub File that contains the Code for the discussed component.
Description	General overview of the component and description of its functionality in the system.
EEA EthTrust Certification Standard*	The testing of the smart contract against the official EthTrust Certification Standard [Source]
Audit-Tool Results*	The results of common security tools that have ran the components code.
Systematic Security	Discussion of security consideration that are better represented in text, than in numbers or graphs.

*where applicable

This is where the individual components act in our system:



Should you have any further questions or remarks on this security report, please email us at yoku@yokupass.com or join our discord https://discord.gg/VpVmRTH4.

2. Receipt NFT Contract

GitHub Link *https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Contracts/Receipt_NFT.sol*

Description

The Receipt NFT Contract mints the Polygon NFT, which displays and saves the metrics of the NFT that the user wants to purchase. Apart from the receipt utility for the user, the NFT is also used in other processes by different contracts to reference the destination of the purchased NFT and includes metrics like the price, fees, and environment variables. A minted Receipt-NFT looks as follows:

Name	Type	Description
PaymentAddress	*Blockchain Address*	The address that pays into the Processing Contract. *Note: The user could technically change their address after the Receipt-NFT was minted and the system would still work and deliver the NFT to their wallet.*
ReceiveAddress	*Blockchain Address*	The address where the customer will receive their purchased NFT.
TokenID	*Integer*	The Identifier of the NFT.
Timestamp	*Timestamp*	The Timestamp of the creation of this Receipt-NFT.
ExchangeRate	*Integer*	The exchange rate of the target cryptocurrency (NFT Marketplace) and the payment cryptocurrency (customer). e.g.: Cardano/Ethereum This number is multiplied by 10^{18}.
Amount	*Integer*	The amount the NFT is listed for on the NFT Marketplace converted to the payment cryptocurrency. This number is multiplied by 10^{18} (to receive the value in e.g.: Wei or Lovelace).
Collateral	*Integer*	The 2% collateral amount that is used to compensate for price fluctuations in the exchange rate over the next minutes until the money gets paid out by the Processing Contract. The collateral is also in the payment cryptocurrency. This number is multiplied by 10^{18} (to receive the value in e.g.: Wei or Lovelace).

Name	Type	Description
YokuFee	*Integer*	The 1% Fee of the value of Amount to compensate Yoku Pay for the service. The YokuFee is also in the payment cryptocurrency. This number is multiplied by 10^{18} (to receive the value in e.g.: Wei or Lovelace).
OperatingFees	*Integer*	The OperatingFees consist of transaction- and exchange fees that incur at later stages of the payment processing system. They are shown in the payment cryptocurrency. This number is multiplied by 10^{18} (to receive the value in e.g.: Wei or Lovelace).
TotalAmount	*Integer*	The Amount, Collateral, YokuFee and OperatingFees combined.
***Chain-specific information (e.g. MarketID, NFT Contract, etc.)**	*Miscellaneous*	Depending on the Marketplace, different specific values are needed for future reference in contracts or Chainlink adapters.

EEA EthTrust Certification Standard

This contract adheres to the EEA EthTrust Security Level [S].

Requirement	Description	**Contract**
[S] No `CREATE2`	Tested code *MUST NOT* contain a `CREATE2` instruction.	Check Does not contain a CREATE2 instruction
[S] No tx.origin	Tested code *MUST NOT* contain a `tx.origin` instruction	Check Does not contain a tx.origin instruction
[S] No Exact Balance Check	Tested code *MUST NOT* test that the balance of an account is exactly equal to (i.e. ==) a specified amount or the value of a variable.	Check Does not test any balance
[S] No Conflicting Inheritance	Tested code *MUST NOT* include more than one variable, or operative function with different code, with the same name	Check Does not contain variables, or operative functions with different code, but with the same name

Requirement	Description	**Contract**
[S] No Hashing Consecutive Variable Length Arguments	Tested Code *MUST NOT* use `abi.encodePacked()` with consecutive variable length arguments.	Check Does not contain a abi.encodePacked()
[S] No Self-destruct	Tested code *MUST NOT* contain the `selfdestruct()` instruction or its now-deprecated alias `suicide()`	Check Does not contain selfdestruct() or suicide()
[S] No assembly()	Tested Code *MUST NOT* contain the `assembly()` instruction	Check Does not contain an assembly() instruction
[S] No Unicode Direction Control Characters	Tested code MUST NOT contain any of the Unicode Direction Control Characters `U+2066`, `U+2067`, `U+2068`, `U+2029`, `U+202A`, `U+202B`, `U+202C`, `U+202D`, or `U+202E`	Check Does not contain any of the Unicode Direction Control Characters
[S] Check External Calls Return	Tested Code that makes external calls using the Low-level Call Functions (i.e. `call`, `delegatecall`, `staticcall`, `send`, and `transfer`) *MUST* check the returned value from each usage to determine whether the call failed.	Check Does not us any custom low-level call functions, just basic ERC-721 functions
[S] Use Check-Effects-Interaction	Tested code that makes external calls *MUST* use the Checks-Effects-Interactions pattern to protect against Re-entrancy Attacks	Check Does not make any external calls
[S] No delegatecall()	Tested Code *MUST NOT* contain the delegatecall() instruction	Check Does not contain a delegatecall() instruction
[S] No Overflow/Underflow	Tested code *MUST NOT* use a version of Solidity older than 0.8.0	Check Solidity Version ^0.8.7.0
[S] Explicit Storage	Tested code *MUST NOT* use a version of Solidity older than 0.5.0	Check Solidity Version ^0.8.7.0
[S] Explicit Constructors	Tested code *MUST NOT* use a version of Solidity older than 0.4.22	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2022-5 with `.push()`	Tested code that copies `bytes` arrays from calldata or memory whose size is not a multiple of 32 bytes, and has an empty `.push()` instruction that writes to the resulting array, *MUST NOT* use a version of Solidity older than 0.8.15.	Check Contract does not contain any .push() instruction
[S] Compiler Bug SOL-2022-3	Tested code that • uses `memory` and `calldata` pointers for the same function, **and** • changes the data location of a function during inheritance, **and** • performs an internal call at a location that is only aware of the original function signature from the base contract *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2022-2	Tested code with a nested array that • passes it to an external function, **or** • passes it as input to `abi.encode()`, **or** • uses it in an event *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Contract does not contain any nested arrays
[S] Compiler Bug SOL-2022-1	Tested code that • uses Number literals for a `bytesNN` type shorter than 32 bytes, **or** • uses String literals for any `bytesNN` type, **and** passes such literals to `abi.encodeCall()` as the first parameter, *MUST NOT* use version 0.8.11 nor 0.8.12 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-4	Tested Code that uses custom value types shorter than 32 bytes *SHOULD* not use version 0.8.8 of Solidity.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2021-2	Tested code that uses `abi.decode()` on byte arrays as `memory`, *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.4.16 and 0.8.3 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-1	Tested code that has 2 or more occurrences of an instruction `keccak(mem,length)` where • the values of *mem* are equal, **and** • the values of *length* are unequal, **and** • the values of *length* are not multiples of 32, *MUST NOT* use the bytecode optimzier with a version of Solidity older than 0.8.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-push	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array using `push()` *MUST NOT* a use version of Solidity older than 0.7.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-10	Tested code that copies an array of types shorter than 16 bytes to a longer array *MUST NOT* a use version of Solidity older than 0.7.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-9	Tested code that defines Free Functions *MUST NOT* use version 0.7.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-8	Tested code that calls internal library functions with calldata parameters called via `using for` *MUST NOT* use version 0.6.9 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-6	Tested code that accesses an array slice using an expression for the starting index that can evaluate to a value other than zero *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.6.0 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2020-7	Tested code that passes a string literal containing two consecutive backslash ("\") characters to an encoding function or an external call *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.5.14 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-5	Tested code that defines a contract that does not include a constructor but has a base contract that defines a constructor not defined as `payable` *MUST NOT* use a version of Solidity between 0.4.5 and 0.6.7 (inclusive), **unless** it meets the Overriding Requirement **[M] Check Constructor Payment**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-4	Tested code that makes assignments to tuples that • have nested tuples, **or** • include a pointer to an external function, **or** • reference a dynamically sized `calldata` array *MUST NOT* use a version of Solidity older than 0.6.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-3	Tested code that declares arrays of size larger than 2^256-1 *MUST NOT* use a version of Solidity older than 0.6.5.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-1	Tested code that declares variables inside a `for` loop that contains a `break` or `continue` statement *MUST NOT* use the Yul Optimizer with version 0.6.0 nor a version of Solidity between 0.5.8 and 0.5.15 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-length	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array by assigning the `length` attribute *MUST NOT* use a version of Solidity older than 0.6.0.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2019-10	Tested code *MUST NOT* use the combination of all of • the ABIEncoderV2 • the Optimizer • the yulOptimizer • version 0.5.14 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bugs SOL-2019-3,6,7,9	Tested code that uses `struct` **or** arrays *MUST NOT* use the ABIEncoderV2 option with a version of Solidity between 0.4.16 and 0.5.10 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-8	Tested code that assigns an array of signed integers to an array of a different type *MUST NOT* use a version of Solidity between 0.4.7 and 0.5.9 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-5	Tested code that calls an uninitialized internal function pointer in the constructor *MUST NOT* use a version between 0.4.5 and 0.4.25 (inclusive) nor a version between 0.5.0 and 0.5.7 (inclusive) of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-4	Tested code that uses events containing contract types, in libraries, *MUST NOT* use a version of Solitidy between 0.5.0 and 0.5.7.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-2	Tested code that makes index access to `bytesNN` types with a second parameter (not the index) whose compile-time value evaluates to 31 *MUST NOT* use the Optimizer with versions 0.5.5 nor 0.5.6 of Solitidy.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-1	Tested code that nests bitwise shifts to produce a total shift of more than 256 bits **and** compiles for the `Constantinople` or later EVM version *MUST NOT* use the Optimizer option with version 0.5.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-4	Tested code that has a match for the regexp `[^/]\\*\\*+[^/0-9]` *MUST NOT* use a version of Solidity older than 0.4.25.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2018-3	Tested code that uses a `struct` in events *MUST NOT* use a version of Solidity between 0.4.17 and 0.4.24 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-2	Tested code that calls a function matching the regexp returns[^;{]*\\[\\s*[^\\] \\t\\r\\n\ \\v\\f][^\\]]*\\]\\s*\\[\\s*[^\\] \\t\\r\ \n\\v\\f][^\\]]*\\][^{;]*[;{] *MUST NOT* use a version of Solidity older than 0.4.22.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-1	Tested code that both a new-style constructor (using the `constructor` keyword) and an old-style constructor (a function with the same name as the contract), which are not exactly the same *MUST NOT* use version 0.4.22 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-5	Tested code with a function that is `payable` whose name consists only of any number of zeros ("0"), and does not have a fallback function, *MUST NOT* use a version of Solidity older than 0.4.18.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-4	Tested code that uses the `delegatecall()` instruction *MUST NOT* use a version of Solidity older than 0.4.15.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-3	Tested code that uses the `ecrecover()` pre-compile *MUST NOT* use a version of Solidity older than 0.4.14	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-2	Tested code with functions that accept 2 or more parameters, of which any but the last are of `bytesNN` type *MUST NOT* use a version of Solidity older than 0.4.12.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-1	Tested code that contains any number that **either** begins with `0xff` and ends with `00`, **or** begins with `0x00` and ends with `ff`, twice, **OR** uses such a number in the constructor, *MUST NOT* use the Optimizer with a version of Solidity older than 0.4.11.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-11	Tested code that uses a version older than 0.4.7 of Solidity *MUST NOT* call the Identity Contract **UNLESS** it meets the Overriding Requirement **[M] Compiler Bug Check Identity Calls**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-10	Tested code *MUST NOT* use the Optimizer option with version 0.4.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-9	Tested code that use variables of a type shorter than 17 bytes *MUST NOT* use a version of Solidity older than 0.4.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-8	Tested code that uses the `sha3()` instruction *MUST NOT* use the Optimizer option with a version of Solidity older than 0.4.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-7	Tested code that uses `delegatecall()` from a function that can receive Ether to call a Library Function *MUST NOT* use versions 0.4.0 or 0.4.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-6	Tested code that sends Ether *MUST NOT* use a version of Solidity older than 0.4.0 **unless** it meets the overriding requirement **[M] Compiler Bug No Zero Ether Send**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-5	Tested code that creates a dynamically sized array with a `length` that can be zero *MUST NOT* use a version of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-4	Tested code that creates a `Jump Destination` opcode *MUST NOT* use the Optimizer with versions of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-3	Tested code that compares the values of data of type `bytesNN` *MUST NOT* use a version of Solidity older than 0.3.3.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-2	Tested code that uses arrays, with data types whose size is less than 17 bytes *MUST NOT* use a version of Solidity older than 0.3.1.	Check Solidity Version ^0.8.7.0
[S] No Ancient Compilers	Tested code *MUST NOT* use a version of Solidity older than 0.3.	Check Solidity Version ^0.8.7.0

Audit-Tool Results

Used Audit Tools for this component were **MythX** from Consensys, **SOLIDITY STATIC ANALYSIS** from Remix, and the **SOLHINT LINTER** from protofire.

MythX:

MythX is a security analysis service for Ethereum smart contracts. Read More: *https://mythx.io/*

Type of Analysis	**High Vulnerabilities**	**Medium Vulnerabilities**	**Low Vulnerabilities**	**URL**
Deep Analysis	0	0	2	https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Security/MythX%20Receipt.pdf

SOLIDITY STATIC ANALYSIS:

The Solidity Static Analysis plugin performs static analysis on Solidity smart contracts once they are compiled. It checks for security vulnerabilities and lousy development practices, among other issues. It can be activated from the Remix Plugin Manager. Read More: *https://remix-ide.readthedocs.io/en/latest/static_analysis.html*

Type of Warning	**Description**	**Position in Code**	**Statement**
-	-	-	-

The Solidity Static Analysis did not throw any security-related Warnings for this contract.

SOLHINT LINTER:

Solhint is an open-source project for linting Solidity code. This project provides both Security and Style Guide validations. Read More: *https://protofire.github.io/solhint/docs/rules.html*

Type of Error	**Position in Code**	**Statement**
Error: Code contains empty blocks	Pos: 14:57:	There is no reason to input something in the constructor.

Systematic Security

The Receipt-NFT Contract builds on the OpenZeppelin[1] Contracts project, which is probably the most famous repository for boilerplate Solidity Smart Contracts in the space. For its purpose to just mint Receipt-NFTs, it does not need to be fancy, and we tried to keep the contract as simple as possible. The Audit of OpenZeppelin themselves can be found here.[2]

The primary function of the entire contract, safeMint() has the modifier onlyOwner:

```
function safeMint(address to, string memory uri) onlyOwner public {...}
```

This has the effect that only we can mint Receipt NFTs, so the chance falls dramatically for manipulation of the whole system.

Other possible security hijacks like reentrancy attacks are not the issue for the Receipt-NFT contract since it doesn't hold any funds and is just for minting the Receipt NFTs. The main concern is that only we can mint these NFTs and that the information is always authentic. How the access is secured was just described. Further security concerns that include the Receipt-NFT on its own will be covered in the other chapters of this security report.

Unit Test

```
const { expect } = require("chai");

describe("NFT", function() {
  it("It should deploy the contract, mint a token, and resolve to the right URI", async function() {
    const NFT = await ethers.getContractFactory("MyNFT");
    const nft = await NFT.deploy();
    const URI = "QmWJBNeQAm9Rh4YaW8GFRnSgwa4dN889VKm9poc2DQPBkv";
    await nft.deployed();
    await nft.mint("0xd72203b26D887a60Af3a11178eF4A48BE8DecbA6", URI)
    expect(await nft.tokenURI(1)).to.equal(URI)
  });
});
```

[1] https://docs.openzeppelin.com/contracts/2.x/api/token/erc721.

[2] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/audit/2018-10.pdf.

3. Ethereum/Polygon/BSC - Processing Contract

GitHub Link *https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Contracts/Processing.sol*

Description

The Processing Contract is the only smart contract the customer directly interacts with when using Yoku Pay on a website. Its purpose is to accept a common cryptocurrency and lock it until the customer receives the desired NFT.

EEA EthTrust Certification Standard

This contract adheres to the EEA EthTrust Security Level [S].

Requirement	Description	**Contract**
[S] No `CREATE2`	Tested code *MUST NOT* contain a `CREATE2` instruction.	Check Does not contain a CREATE2 instruction
[S] No tx.origin	Tested code *MUST NOT* contain a `tx.origin` instruction	Check Does not contain a tx.origin instruction
[S] No Exact Balance Check	Tested code *MUST NOT* test that the balance of an account is exactly equal to (i.e. ==) a specified amount or the value of a variable.	Check Does not contain any „exactly equal to“ balance checks of accounts
[S] No Conflicting Inheritance	Tested code *MUST NOT* include more than one variable, or operative function with different code, with the same name	Check Does not contain variables, or operative functions with different code, but with the same name
[S] No Hashing Consecutive Variable Length Arguments	Tested Code *MUST NOT* use `abi.encodePacked()` with consecutive variable length arguments.	Check Does not contain a abi.encodePacked()
[S] No Self-destruct	Tested code *MUST NOT* contain the `selfdestruct()` instruction or its now-deprecated alias `suicide()`	Check Does not contain selfdestruct() or suicide()
[S] No assembly()	Tested Code *MUST NOT* contain the `assembly()` instruction	Check Does not contain an assembly() instruction

Requirement	Description	**Contract**
[S] No Unicode Direction Control Characters	Tested code MUST NOT contain any of the Unicode Direction Control Characters `U+2066`, `U+2067`, `U+2068`, `U+2029`, `U+202A`, `U+202B`, `U+202C`, `U+202D`, or `U+202E`	Check Does not contain any of the Unicode Direction Control Characters
[S] Check External Calls Return	Tested Code that makes external calls using the Low-level Call Functions (i.e. `call`, `delegatecall`, `staticcall`, `send`, and `transfer`) *MUST* check the returned value from each usage to determine whether the call failed.	Check transfer() calls are error-handled. As seen in the lines 110-151
[S] Use Check-Effects-Interaction	Tested code that makes external calls *MUST* use the Checks-Effects-Interactions pattern to protect against Re-entrancy Attacks	Check First the msg.sender is checked with their time of deposit. Then their balance is checked and only then funds will be transferred. Also the function has a Re-entrancy lock modifier.
[S] No delegatecall()	Tested Code *MUST NOT* contain the delegatecall() instruction	Check Does not contain a delegatecall() instruction
[S] No Overflow/Underflow	Tested code *MUST NOT* use a version of Solidity older than 0.8.0	Check Solidity Version ^0.8.7.0
[S] Explicit Storage	Tested code *MUST NOT* use a version of Solidity older than 0.5.0	Check Solidity Version ^0.8.7.0
[S] Explicit Constructors	Tested code *MUST NOT* use a version of Solidity older than 0.4.22	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2022-5 with `.push()`	Tested code that copies `bytes` arrays from calldata or memory whose size is not a multiple of 32 bytes, and has an empty `.push()` instruction that writes to the resulting array, *MUST NOT* use a version of Solidity older than 0.8.15.	Check Does not user any .push() instruction

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2022-3	Tested code that • uses `memory` and `calldata` pointers for the same function, **and** • changes the data location of a function during inheritance, **and** • performs an internal call at a location that is only aware of the original function signature from the base contract *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2022-2	Tested code with a nested array that • passes it to an external function, **or** • passes it as input to `abi.encode()`, **or** • uses it in an event *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Does not user any nested arrays
[S] Compiler Bug SOL-2022-1	Tested code that • uses Number literals for a `bytesNN` type shorter than 32 bytes, **or** • uses String literals for any `bytesNN` type, **and** passes such literals to `abi.encodeCall()` as the first parameter, *MUST NOT* use version 0.8.11 nor 0.8.12 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-4	Tested Code that uses custom value types shorter than 32 bytes *SHOULD* not use version 0.8.8 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-2	Tested code that uses `abi.decode()` on byte arrays as `memory`, *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.4.16 and 0.8.3 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2021-1	Tested code that has 2 or more occurrences of an instruction `keccak(mem,length)` where • the values of *mem* are equal, **and** • the values of *length* are unequal, **and** • the values of *length* are not multiples of 32, *MUST NOT* use the bytecode optimzier with a version of Solidity older than 0.8.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-push	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array using `push()` *MUST NOT* a use version of Solidity older than 0.7.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-10	Tested code that copies an array of types shorter than 16 bytes to a longer array *MUST NOT* a use version of Solidity older than 0.7.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-9	Tested code that defines Free Functions *MUST NOT* use version 0.7.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-8	Tested code that calls internal library functions with calldata parameters called via `using for` *MUST NOT* use version 0.6.9 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-6	Tested code that accesses an array slice using an expression for the starting index that can evaluate to a value other than zero *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.6.0 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-7	Tested code that passes a string literal containing two consecutive backslash ("\") characters to an encoding function or an external call *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.5.14 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2020-5	Tested code that defines a contract that does not include a constructor but has a base contract that defines a constructor not defined as `payable` *MUST NOT* use a version of Solidity between 0.4.5 and 0.6.7 (inclusive), **unless** it meets the Overriding Requirement **[M] Check Constructor Payment**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-4	Tested code that makes assignments to tuples that • have nested tuples, **or** • include a pointer to an external function, **or** • reference a dynamically sized `calldata` array *MUST NOT* use a version of Solidity older than 0.6.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-3	Tested code that declares arrays of size larger than 2^256-1 *MUST NOT* use a version of Solidity older than 0.6.5.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-1	Tested code that declares variables inside a `for` loop that contains a `break` or `continue` statement *MUST NOT* use the Yul Optimizer with version 0.6.0 nor a version of Solidity between 0.5.8 and 0.5.15 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-length	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array by assigning the `length` attribute *MUST NOT* use a version of Solidity older than 0.6.0.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-10	Tested code *MUST NOT* use the combination of all of • the ABIEncoderV2 • the Optimizer • the yulOptimizer • version 0.5.14 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bugs SOL-2019-3,6,7,9	Tested code that uses `struct` **or** arrays *MUST NOT* use the ABIEncoderV2 option with a version of Solidity between 0.4.16 and 0.5.10 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2019-8	Tested code that assigns an array of signed integers to an array of a different type *MUST NOT* use a version of Solidity between 0.4.7 and 0.5.9 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-5	Tested code that calls an uninitialized internal function pointer in the constructor *MUST NOT* use a version between 0.4.5 and 0.4.25 (inclusive) nor a version between 0.5.0 and 0.5.7 (inclusive) of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-4	Tested code that uses events containing contract types, in libraries, *MUST NOT* use a version of Solitidy between 0.5.0 and 0.5.7.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-2	Tested code that makes index access to `bytesNN` types with a second parameter (not the index) whose compile-time value evaluates to 31 *MUST NOT* use the Optimizer with versions 0.5.5 nor 0.5.6 of Solitidy.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-1	Tested code that nests bitwise shifts to produce a total shift of more than 256 bits **and** compiles for the `Constantinople` or later EVM version *MUST NOT* use the Optimizer option with version 0.5.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-4	Tested code that has a match for the regexp `[^/]\\*\\**[^/0-9 ]` *MUST NOT* use a version of Solidity older than 0.4.25.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-3	Tested code that uses a `struct` in events *MUST NOT* use a version of Solidity between 0.4.17 and 0.4.24 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2018-2	Tested code that calls a function matching the regexp returns[^;{]*\\[\\s*[^\\] \\t\\r\\n\ \v\\f][^\\]]*\\]\\s*\\[\\s*[^\\] \\t\\r\ \n\\v\\f][^\\]]*\\][^{;]*[;{] *MUST NOT* use a version of Solidity older than 0.4.22.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-1	Tested code that both a new-style constructor (using the `constructor` keyword) and an old-style constructor (a function with the same name as the contract), which are not exactly the same *MUST NOT* use version 0.4.22 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-5	Tested code with a function that is `payable` whose name consists only of any number of zeros ("0"), and does not have a fallback function, *MUST NOT* use a version of Solidity older than 0.4.18.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-4	Tested code that uses the `delegatecall()` instruction *MUST NOT* use a version of Solidity older than 0.4.15.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-3	Tested code that uses the `ecrecover()` pre-compile *MUST NOT* use a version of Solidity older than 0.4.14	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-2	Tested code with functions that accept 2 or more parameters, of which any but the last are of `bytesNN` type *MUST NOT* use a version of Solidity older than 0.4.12.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-1	Tested code that contains any number that **either** begins with `0xff` and ends with `00`, **or** begins with `0x00` and ends with `ff`, twice, **OR** uses such a number in the constructor, *MUST NOT* use the Optimizer with a version of Solidity older than 0.4.11.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-11	Tested code that uses a version older than 0.4.7 of Solidity *MUST NOT* call the Identity Contract **UNLESS** it meets the Overriding Requirement **[M] Compiler Bug Check Identity Calls**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-10	Tested code *MUST NOT* use the Optimizer option with version 0.4.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-9	Tested code that use variables of a type shorter than 17 bytes *MUST NOT* use a version of Solidity older than 0.4.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-8	Tested code that uses the `sha3()` instruction *MUST NOT* use the Optimizer option with a version of Solidity older than 0.4.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-7	Tested code that uses `delegatecall()` from a function that can receive Ether to call a Library Function *MUST NOT* use versions 0.4.0 or 0.4.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-6	Tested code that sends Ether *MUST NOT* use a version of Solidity older than 0.4.0 **unless** it meets the overriding requirement **[M] Compiler Bug No Zero Ether Send**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-5	Tested code that creates a dynamically sized array with a `length` that can be zero *MUST NOT* use a version of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-4	Tested code that creates a `Jump Destination` opcode *MUST NOT* use the Optimizer with versions of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-3	Tested code that compares the values of data of type `bytesNN` *MUST NOT* use a version of Solidity older than 0.3.3.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-2	Tested code that uses arrays, with data types whose size is less than 17 bytes *MUST NOT* use a version of Solidity older than 0.3.1.	Check Solidity Version ^0.8.7.0
[S] No Ancient Compilers	Tested code *MUST NOT* use a version of Solidity older than 0.3.	Check Solidity Version ^0.8.7.0

Audit-Tool Results

Used Audit Tools for this component were **MythX** from Consensys, **SOLIDITY STATIC ANALYSIS** from Remix, and the **SOLHINT LINTER** from protofire.

MythX:

MythX is a security analysis service for Ethereum smart contracts. Read More: *https://mythx.io/*

Type of Analysis	**High Vulnerabilities**	**Medium Vulnerabilities**	**Low Vulnerabilities**	**URL**
Deep Analysis	0	0	2	https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Security/MythX%20Processing.pdf

SOLIDITY STATIC ANALYSIS:

The Solidity Static Analysis plugin performs static analysis on Solidity smart contracts once they are compiled. It checks for security vulnerabilities and lousy development practices, among other issues. It can be activated from the Remix Plugin Manager. Read More: *https://remix-ide.readthedocs.io/en/latest/static_analysis.html*

Type of Warning	**Description**	**Position in Code**	**Statement**
Block timestamp	Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.	(1) Pos: 83:35: (2) Pos: 143:43:	We use the „block.timestamp“ for the user withdraw function (line 164), where the user can withdraw their funds after 24 hours, if the NFT hasn't been delivered. Our use of the „block.timestamp“ adheres to the 15 second rule and is therefore safe to use.

SOLHINT LINTER:

Solhint is an open-source project for linting Solidity code. This project provides both Security and Style Guide validations. Read More: *https://protofire.github.io/solhint/docs/rules.html*

Type of Warning	Position in Code	Statement
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	Pos: 39	The constructor is at line 39 in the contract. It does not need any visibility. This warning is thrown, since the linter is set to {“ignoreConstructors”:false} Link
Avoid to make time-based decisions in your business logic	(1) Pos: 83:35: (2) Pos: 143:43:	See last page

Systematic Security

In general, the only functions that any person apart from us could interact with are deposit() and withdraw(). The other functions are only callable by the Chainlink oracle or the contract owner.

Also, the calculations are interesting to highlight at this point. Solidity does not allow for decimal values, so an easy division to determine the collateral or payback is not self-evident. Thankfully we worked with Wei and coded our equations such that the division of any value (Solidity rounds to Integers) is always at the end of a calculation. The withdrawYoku() function does its math right by adhering to these principles.

In the following, we address common Solidity exploits:

Transaction ordering dependence

A transaction ordering dependence attack[3] is impossible with the Processing Contract since no price can be updated, and each transaction value is unique depending on the desired NFT. The oracle checks if the value of the crypto sent corresponds to the value set in the receipt NFT. If the user should send an amount that does not compare to the agreed value in the receipt NFT, then the contract will flag that transaction and keep it locked until the user tries to withdraw it manually. The manipulation of the price should, in no case, happen accidentally. Still, for any edge case, we didn’t think of, we will allow the user to withdraw their false transaction value again.

Timestamp dependence

We use the „block.timestamp“ for the withdraw() function, where users can withdraw their funds after 24 hours if the NFT hasn’t been delivered. Our use of the „block.timestamp“ adheres to the 15-second rule[4] and is, therefore, safe to use.

Reentrancy and cross-function vulnerabilities

A reentrancy attack, in simple terms, happens if an opposing party makes recursive calls to a function that allows withdrawing funds from the contract. The main problem is that depending on the code; the balance cannot update as soon as the next call comes in and therefore lets that person withdraw more funds than intended.[5] A famous example of this hack is „The DAO“ hack. We have secured our withdrawal function with a locking mechanism that locks the function from calls until the earlier call has been handled. This action protects the contract from reentrancy attacks. Reentrancy attacks are not possible with the withdraw() function.

---

[3] https://medium.com/coinmonks/solidity-transaction-ordering-attacks-1193a014884e

[4] https://cryptomarketpool.com/block-timestamp-manipulation-attack/

[5] https://hackernoon.com/hack-solidity-reentrancy-attack

In the end, it is fair to say that the Processing contract does adhere to the current best practices and does not show any vulnerabilities by opposing parties.

4. NFT Purchase Contract

GitHub Link *https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Contracts/Purchase_OpenTheta_NFT.sol*

Description

The NFT purchase contract is a Solidity contract triggered by an API that purchases NFTs from EVM-compatible NFT Marketplaces and sends them to the customers' wallet in the same transaction. This contract has the purpose of increasing the transparency and efficiency of Yoku Pay.

EEA EthTrust Certification Standard

This contract adheres to the EEA EthTrust Security Level [S].
**Please check the comment for [S] Check External Calls Return*

Requirement	Description	**Contract**
[S] No `CREATE2`	Tested code *MUST NOT* contain a `CREATE2` instruction.	Check Does not contain a CREATE2 instruction
[S] No tx.origin	Tested code *MUST NOT* contain a `tx.origin` instruction	Check Does not contain a tx.origin instruction
[S] No Exact Balance Check	Tested code *MUST NOT* test that the balance of an account is exactly equal to (i.e. ==) a specified amount or the value of a variable.	Check Does not test any balances
[S] No Conflicting Inheritance	Tested code *MUST NOT* include more than one variable, or operative function with different code, with the same name	Check Does not contain variables, or operative functions with different code, but with the same name
[S] No Hashing Consecutive Variable Length Arguments	Tested Code *MUST NOT* use `abi.encodePacked()` with consecutive variable length arguments.	Check Does not contain a abi.encodePacked()
[S] No Self-destruct	Tested code *MUST NOT* contain the `selfdestruct()` instruction or its now-deprecated alias `suicide()`	Check Does not contain selfdestruct() or suicide()
[S] No assembly()	Tested Code *MUST NOT* contain the `assembly()` instruction	Check Does not contain an assembly() instruction

Requirement	Description	**Contract**
[S] No Unicode Direction Control Characters	Tested code MUST NOT contain any of the Unicode Direction Control Characters `U+2066`, `U+2067`, `U+2068`, `U+2029`, `U+202A`, `U+202B`, `U+202C`, `U+202D`, or `U+202E`	Check Does not contain any of the Unicode Direction Control Characters
[S] Check External Calls Return	Tested Code that makes external calls using the Low-level Call Functions (i.e. `call`, `delegatecall`, `staticcall`, `send`, and `transfer`) *MUST* check the returned value from each usage to determine whether the call failed.	Check Does not contain low-level call functions, however contains transferFrom(). Note: The transaction would revert, if the transfer of the NFT fails, since everything happens in one function
[S] Use Check-Effects-Interaction	Tested code that makes external calls *MUST* use the Checks-Effects-Interactions pattern to protect against Re-entrancy Attacks	Check Re-entrancy is not viable, since the contract does not hold any funds
[S] No delegatecall()	Tested Code *MUST NOT* contain the delegatecall() instruction	Check Does not contain a delegatecall() instruction
[S] No Overflow/Underflow	Tested code *MUST NOT* use a version of Solidity older than 0.8.0	Check Solidity Version ^0.8.7.0
[S] Explicit Storage	Tested code *MUST NOT* use a version of Solidity older than 0.5.0	Check Solidity Version ^0.8.7.0
[S] Explicit Constructors	Tested code *MUST NOT* use a version of Solidity older than 0.4.22	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2022-5 with `.push()`	Tested code that copies `bytes` arrays from calldata or memory whose size is not a multiple of 32 bytes, and has an empty `.push()` instruction that writes to the resulting array, *MUST NOT* use a version of Solidity older than 0.8.15.	Check Does not contain a .push() instruction

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2022-3	Tested code that • uses `memory` and `calldata` pointers for the same function, **and** • changes the data location of a function during inheritance, **and** • performs an internal call at a location that is only aware of the original function signature from the base contract *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2022-2	Tested code with a nested array that • passes it to an external function, **or** • passes it as input to `abi.encode()`, **or** • uses it in an event *MUST NOT* use a version of Solidity between 0.6.9 and 0.8.13 (inclusive).	Check Does not contain any nested arrays
[S] Compiler Bug SOL-2022-1	Tested code that • uses Number literals for a `bytesNN` type shorter than 32 bytes, **or** • uses String literals for any `bytesNN` type, **and** passes such literals to `abi.encodeCall()` as the first parameter, *MUST NOT* use version 0.8.11 nor 0.8.12 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-4	Tested Code that uses custom value types shorter than 32 bytes *SHOULD* not use version 0.8.8 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2021-2	Tested code that uses `abi.decode()` on byte arrays as `memory`, *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.4.16 and 0.8.3 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2021-1	Tested code that has 2 or more occurrences of an instruction `keccak(mem,length)` where • the values of *mem* are equal, **and** • the values of *length* are unequal, **and** • the values of *length* are not multiples of 32, *MUST NOT* use the bytecode optimzier with a version of Solidity older than 0.8.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-push	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array using `push()` *MUST NOT* a use version of Solidity older than 0.7.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-10	Tested code that copies an array of types shorter than 16 bytes to a longer array *MUST NOT* a use version of Solidity older than 0.7.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-9	Tested code that defines Free Functions *MUST NOT* use version 0.7.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-8	Tested code that calls internal library functions with calldata parameters called via `using for` *MUST NOT* use version 0.6.9 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-6	Tested code that accesses an array slice using an expression for the starting index that can evaluate to a value other than zero *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.6.0 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-7	Tested code that passes a string literal containing two consecutive backslash ("\") characters to an encoding function or an external call *MUST NOT* use the ABIEncoderV2 with a version of Solidity between 0.5.14 and 0.6.7 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2020-5	Tested code that defines a contract that does not include a constructor but has a base contract that defines a constructor not defined as `payable` *MUST NOT* use a version of Solidity between 0.4.5 and 0.6.7 (inclusive), **unless** it meets the Overriding Requirement **[M] Check Constructor Payment**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-4	Tested code that makes assignments to tuples that • have nested tuples, **or** • include a pointer to an external function, **or** • reference a dynamically sized `calldata` array *MUST NOT* use a version of Solidity older than 0.6.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-3	Tested code that declares arrays of size larger than 2^256-1 *MUST NOT* use a version of Solidity older than 0.6.5.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-1	Tested code that declares variables inside a `for` loop that contains a `break` or `continue` statement *MUST NOT* use the Yul Optimizer with version 0.6.0 nor a version of Solidity between 0.5.8 and 0.5.15 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2020-11-length	Tested code that copies an empty byte array to storage, and subsequently increases the size of the array by assigning the `length` attribute *MUST NOT* use a version of Solidity older than 0.6.0.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-10	Tested code *MUST NOT* use the combination of all of • the ABIEncoderV2 • the Optimizer • the yulOptimizer • version 0.5.14 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bugs SOL-2019-3,6,7,9	Tested code that uses `struct` **or** arrays *MUST NOT* use the ABIEncoderV2 option with a version of Solidity between 0.4.16 and 0.5.10 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2019-8	Tested code that assigns an array of signed integers to an array of a different type *MUST NOT* use a version of Solidity between 0.4.7 and 0.5.9 (inclusive).	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-5	Tested code that calls an uninitialized internal function pointer in the constructor *MUST NOT* use a version between 0.4.5 and 0.4.25 (inclusive) nor a version between 0.5.0 and 0.5.7 (inclusive) of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-4	Tested code that uses events containing contract types, in libraries, *MUST NOT* use a version of Solitidy between 0.5.0 and 0.5.7.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-2	Tested code that makes index access to `bytesNN` types with a second parameter (not the index) whose compile-time value evaluates to 31 *MUST NOT* use the Optimizer with versions 0.5.5 nor 0.5.6 of Solitidy.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2019-1	Tested code that nests bitwise shifts to produce a total shift of more than 256 bits **and** compiles for the `Constantinople` or later EVM version *MUST NOT* use the Optimizer option with version 0.5.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-4	Tested code that has a match for the regexp `[^/]\\*\\**[^/0-9 ]` *MUST NOT* use a version of Solidity older than 0.4.25.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-3	Tested code that uses a `struct` in events *MUST NOT* use a version of Solidity between 0.4.17 and 0.4.24 (inclusive).	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2018-2	Tested code that calls a function matching the regexp returns[^;{]*\\[\\s*[^\\] \\t\\r\\n\ \v\\f][^\\]]*\\]\\s*\\[\\s*[^\\] \\t\\r\ \n\\v\\f][^\\]]*\\][^{;]*[;{] *MUST NOT* use a version of Solidity older than 0.4.22.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2018-1	Tested code that both a new-style constructor (using the `constructor` keyword) and an old-style constructor (a function with the same name as the contract), which are not exactly the same *MUST NOT* use version 0.4.22 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-5	Tested code with a function that is `payable` whose name consists only of any number of zeros ("0"), and does not have a fallback function, *MUST NOT* use a version of Solidity older than 0.4.18.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-4	Tested code that uses the `delegatecall()` instruction *MUST NOT* use a version of Solidity older than 0.4.15.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-3	Tested code that uses the `ecrecover()` pre-compile *MUST NOT* use a version of Solidity older than 0.4.14	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-2	Tested code with functions that accept 2 or more parameters, of which any but the last are of `bytesNN` type *MUST NOT* use a version of Solidity older than 0.4.12.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2017-1	Tested code that contains any number that **either** begins with `0xff` and ends with `00`, **or** begins with `0x00` and ends with `ff`, twice, **OR** uses such a number in the constructor, *MUST NOT* use the Optimizer with a version of Solidity older than 0.4.11.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-11	Tested code that uses a version older than 0.4.7 of Solidity *MUST NOT* call the Identity Contract **UNLESS** it meets the Overriding Requirement **[M] Compiler Bug Check Identity Calls**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-10	Tested code *MUST NOT* use the Optimizer option with version 0.4.5 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-9	Tested code that use variables of a type shorter than 17 bytes *MUST NOT* use a version of Solidity older than 0.4.4.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-8	Tested code that uses the `sha3()` instruction *MUST NOT* use the Optimizer option with a version of Solidity older than 0.4.3.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-7	Tested code that uses `delegatecall()` from a function that can receive Ether to call a Library Function *MUST NOT* use versions 0.4.0 or 0.4.1 of Solidity.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-6	Tested code that sends Ether *MUST NOT* use a version of Solidity older than 0.4.0 **unless** it meets the overriding requirement **[M] Compiler Bug No Zero Ether Send**.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-5	Tested code that creates a dynamically sized array with a `length` that can be zero *MUST NOT* use a version of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-4	Tested code that creates a `Jump Destination` opcode *MUST NOT* use the Optimizer with versions of Solidity older than 0.3.6.	Check Solidity Version ^0.8.7.0
[S] Compiler Bug SOL-2016-3	Tested code that compares the values of data of type `bytesNN` *MUST NOT* use a version of Solidity older than 0.3.3.	Check Solidity Version ^0.8.7.0

Requirement	Description	**Contract**
[S] Compiler Bug SOL-2016-2	Tested code that uses arrays, with data types whose size is less than 17 bytes *MUST NOT* use a version of Solidity older than 0.3.1.	Check Solidity Version ^0.8.7.0
[S] No Ancient Compilers	Tested code *MUST NOT* use a version of Solidity older than 0.3.	Check Solidity Version ^0.8.7.0

Audit-Tool Results

Used Audit Tools for this component were **MythX** from Consensys.

MythX:

MythX is a security analysis service for Ethereum smart contracts. Read More: *https://mythx.io/*

Type of Analysis	**High Vulnerabilities**	**Medium Vulnerabilities**	**Low Vulnerabilities**	**URL**
Deep Analysis	0	0	3	https://github.com/YokuPass/YokuPay_Open_Source/blob/main/Security/MythX%20buycontract.pdf

Systematic Security

The Security for the NFT Purchase contract is simple since it contains only one function. The contract does not store any funds, and should something go wrong; then it will revert the transaction from the beginning. The code is so concise that it fits under this paragraph:

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.7.0;

import "https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/token/ERC721/ERC721.sol";

contract BuyContract {

    function buyNFT(address NFTcontract, uint marketID, uint NFTid, uint price, address customer) public
payable {
        openThetaContract(0xbB5f35D40132A0478f6aa91e79962e9F752167EA).createMarketSale{value:price}
(NFTcontract, marketID);
        ERC721 token = ERC721(NFTcontract);
        token.transferFrom(address(this), customer, NFTid);
    }
}

interface openThetaContract {
    function createMarketSale(address value, uint256 value2) external payable;
}
```

5. Chainlink Oracles System

GitHub Link: *https://github.com/YokuPass/YokuPay_Open_Source/tree/main/YokuPay_External_Adapter*

Description

The Chainlink Oracles System notifies the processing contract that has locked the funds whether the desired service has been delivered to the user. In the case of Yoku Pay implemented on an NFT Marketplace, the Oracles check if the desired NFT has entered the customer's wallet. This information gets passed to the processing contract, unlocking the funds back to the customer or us. The primary purpose is to ensure the authenticity and correctness of the data, which gets passed to the processing contract.

Systematic Security

General Oracle security

The oracle system is based on an official docker image from Chainlink[6] and runs in a separate docker environment. Server protection is described in *Server Overview (Chapter 7).*

External Adapter Error handling

The external adapter handles erroneous inputs by the oracle or third parties by not processing receipt NFTs suspected to be fraudulent.

Decentralization of the External Adapters

To persevere decentralization of our overarching system, we utilize multiple external adapters (hosted by different parties) and compare the results in the oracle. The most returned result will be passed over to the smart contract. Even if an external adapter hypothetically would have been compromised or manipulated, the authenticity of information for the contract is protected.

[6] https://chain.link

6. Browser Overview

GitHub Link *https://github.com/YokuPass/YokuPay_Open_Source/tree/main/YokuPay_Website*

Description

The Browser Overview is a combined term for all UI elements accessible to users before, during, and after using Yoku Pay on a website.

Systematic Security

To guarantee a secure browser overview that users interact with, we have taken a list of measures that will be explained in the following paragraphs.

JSON Web Token

We use JSON Web Tokens (JWT) to store the purchase information. This gives us the security to store sensitive information on the website. The JSON Web Token is a JSON-based RFC 7519[7] standardized access token. The payment process is possible only if the JWT exists. Due to the symmetric encryption, the information can only be accessed with the JWT-secret.

```
var data = qs.stringify({
  orderID: state.state.order,
});

var config = {
  method: "post",
  url: "https://receipt-nft.yokupass.com/database/jwt",
  headers: {
    Authorization: process.env.REACT_APP_BEAR,
    "Content-Type": "application/x-www-form-urlencoded",
  },
  data: data,
};

axios(config)
  .then((json) => {
    console.log(json.data);
    const decode = jwtDecoder(json.data.token);
    setJWT(json.data.token);
```

Cookies

To store data that is needed during the payment process, we use Cookies. This allows users to refresh or leave the website without losing the purchase information. After every purchase or session ends, the cookies are deleted.

HTTPS Protocol

HTTPS (HyperText Transfer Protocol Secure) is an internet communication protocol that protects the integrity and confidentiality of data traffic between the user's computer and the website. Users expect a secure environment when they use a website.

[7] https://tools.ietf.org/html/rfc7519.

Environment variables

Environment variables keep static process information and are part of the running environment. These variables can not be changed after running the environment (Website/Server). They contain the necessary sensitive information like authentication keys or other data. Only the process in the environment can access the needed information.

Error handling - loss of information

The error handling ensures a smooth payment process. If, in any case, necessary information or data gets lost during the process, it will be forced to start over.

Security against Attacks

Linking Manipulation

The marketplace needs to send the purchase information if a user gets linked to the Yoku Pay Website. It is essential to create a secure pipeline so that it is nearly impossible to manipulate the provided data during linking. Therefore we are using JSON Web Tokens. The Store creates the JWT and sends it to the Yoku Pay Website during linking. The JWT only can be accessed or created with the correct JWT secret.

Verified orders

Every purchase gets an order ID to identify the order later. In addition, the order ID ensures that the purchase is valid. Therefore the store generates an order ID and adds it to the order database. The current order ID gets checked during the payment process to prevent invalid orders. In addition, the JWT, which contains the purchase information from the store, gets reloaded after every page change or reload.

No purchase information

The process must be stopped if an error happens and the purchase information gets lost. Therefore Yoku Pay ensures that the purchase information is always available. In case of a mistake, the user can contact us (Yoku Pay) or go back to the store and try it again.

Change purchase information

Through the JSON Web Token and Cookie structure, there is no way to change the purchase information (JSON Web Token / Cookie). The only possible way is to get the JSON Web Token secret, create a new one, and edit the Cookie with JWT data inside. This seems unrealistic because, therefore, the secret needs to be stolen.

JWT Secret Brute Forcing

Another way to compromise a JWT is to brute force the secret. Because we use a symmetric 256-bit random generated secret, there are 2^256 possibilities. It takes trillions of years to brute force a 256-bit Chiffre.

7. Server Overview

GitHub Link *https://github.com/YokuPass/YokuPay_Open_Source/tree/main/YokuPay_Services*

Description

The Server Overview includes ChainLink external adapters, Cardano Node, PSQL Databases, and processing APIs.

Systematic Security

API Authentification (Bearer Token)

The API authentication ensures that only justified parties can do API calls. The server will return an error if the API call has no valid authentication header. The Bearer Token Usage is defined in RFC 6750.

Error handling (Invalid Input)

The error handling ensures that in case of invalid inputs, the server does not break. If the input is invalid, the server can return an error. This error handling is used in all APIs provided by us (YokuPay).

Environment variables

Environment variables keep static process information and are part of the running environment. These variables can not be changed after running the environment (Website/Server). They contain the necessary sensitive information like authentication keys or other data. Only the process in the environment can access the needed information.

Secure Shell (SSH)

To interact with our servers, we use the SSH protocol. This is natively provided and managed by our server provider. SSH is a cryptographic network protocol for the secure operation of network services over unsecured networks. Thus, every device that wants to access the server with a shell must be registered with its RSA key.

Security against Attacks

DDOS protection

Our server host natively ensures DDOS protection. This protects our server from DDOS attacks.

2-Factor-Authentication

The installed 2-Factor-Authentication is provided by Google and ensures only valid devices have access. To unlock the server, you must enter the authentication code from the authenticator app.

Firewall

Firewalls are a must-have to ensure that servers are safe. They filter incoming and outgoing traffic to allow only specific services and lockout unsafe ones.

9th of September

8. End Note

In conclusion, we hope to have demonstrated that our processes and code are secure with this security paper. If you have any remarks, ideas, or criticism, don't hesitate to contact us at our details below. In the future, we would like to simulate a stress test on the assembled system and integrate error-reporting pipelines into a dashboard.

Team Yoku

About the authors

Jan Christian Swoboda

Law Student
EBS Universität für Wirtschaft und Recht

Co-Founder at Yoku Pay and Blockchain Developer since March 2021

jan.swoboda@students.ebs.edu

Philipp Storz

Trainee
Wiesheu GmbH

Co-Founder at Yoku Pay and Developer since December 2020

philipp.storz43@gmail.com