SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
09. January, 2022

For

PAWN MY NFT

NFT DEFI AT YOUR FINGERTIPS

Disclaimer

Version	Date	Description
1.0	09. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Ethereum (ERC 20)

Website
https://pawnmynft.io/

Telegram
https://t.me/pawnmynft

Twitter
https://twitter.com/PawnMyNFT

Instagram
https://www.instagram.com/pawnmynft/

Youtube
https://www.youtube.com/channel/UClmIb9i2Uxqac4V7aJOBoGg

Description

NFT Defi at your fingertips.

Project Engagement

During the 5th of January 2022, **Pawn My NFT Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Github
    - https://github.com/cryptored007/pawnmynft_smartcontract
    - Commit: 28ba18870041eb0d436ba45b620ab24b399f34bf

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	1
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/security/Pausable.sol	2
@openzeppelin/contracts/security/ReentrancyGuard.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	2
@openzeppelin/contracts/token/ERC721/IERC721.sol	3
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	2
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	2
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	2

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

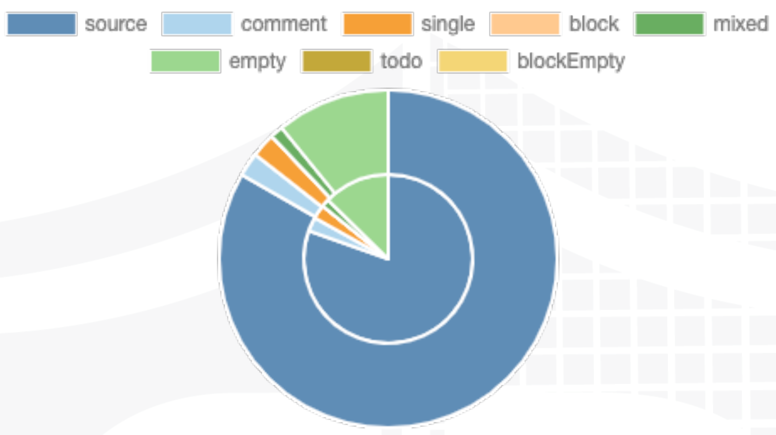
v1.0

File Name	SHA-1 Hash
contracts/PawnMyNFTMortgage.sol	12b218520395b3cae263769556ea5925a7fc6eb9
contracts/Locker.sol	e132c3bdc8050e00f92c6075503492be777369da
contracts/Lib.sol	b7af2a7ab617f3c81da86265f97de81386cbc671
contracts/PawnMyNFT.sol	f2015da992f91220f9022dba346ee30b438f1b52

Metrics

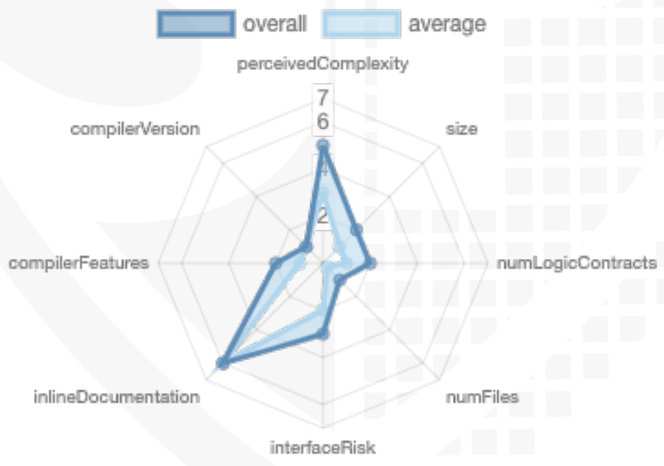
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	2	0	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	27	0

Version	External	Internal	Private	Pure	View
1.0	10	29	0	10	13

State Variables

Version	Total	Public
1.0	16	11

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.4`			`yes` (1 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0				`yes`	`yes`	

Scope of Work

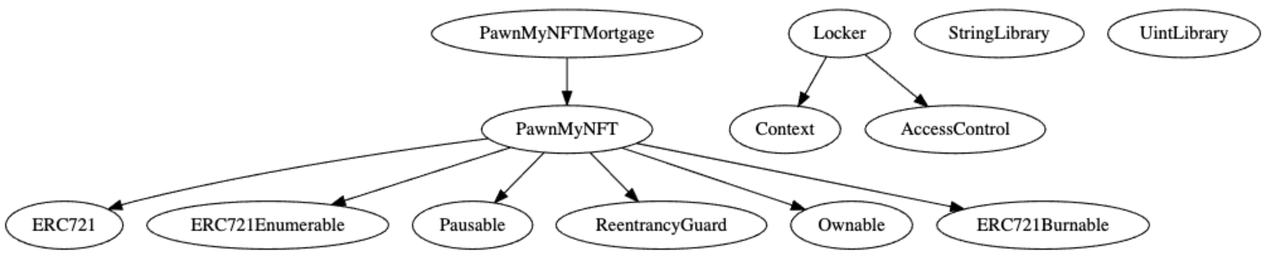
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

v1.0



Verify Claims

Correct implementation of ERC721 standard

Tested	Verified
✓	✓

PawnMyNFT

Functions

[✓] balanceOf(address) is present
    [✓] balanceOf(address) -> () (correct return value)
    [✓] balanceOf(address) is view
[✓] ownerOf(uint256) is present
    [✓] ownerOf(uint256) -> () (correct return value)
    [✓] ownerOf(uint256) is view
[✓] safeTransferFrom(address,address,uint256,bytes) is present
    [✓] safeTransferFrom(address,address,uint256,bytes) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] safeTransferFrom(address,address,uint256) is present
    [✓] safeTransferFrom(address,address,uint256) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
    [✓] transferFrom(address,address,uint256) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
    [✓] approve(address,uint256) -> () (correct return type)
    [✓] Approval(address,address,uint256) is emitted
[✓] setApprovalForAll(address,bool) is present
    [✓] setApprovalForAll(address,bool) -> () (correct return type)
    [✓] ApprovalForAll(address,address,bool) is emitted
[✓] getApproved(uint256) is present
    [✓] getApproved(uint256) -> () (correct return value)
    [✓] getApproved(uint256) is view
[✓] isApprovedForAll(address,address) is present
    [✓] isApprovedForAll(address,address) -> () (correct return value)
    [✓] isApprovedForAll(address,address) is view
[✓] supportsInterface(bytes4) is present
    [✓] supportsInterface(bytes4) -> () (correct return value)
    [✓] supportsInterface(bytes4) is view
[✓] name() is present
    [✓] name() -> () (correct return value)

[✓] name() is view
[✓] symbol() is present
[✓] symbol() -> () (correct return value)
[✓] tokenURI(uint256) is present
[✓] tokenURI(uint256) -> () (correct return value)

Events
[✓] Transfer(address,address,uint256) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed
[✓] parameter 2 is indexed
[✓] Approval(address,address,uint256) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed
[✓] parameter 2 is indexed
[✓] ApprovalForAll(address,address,bool) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed

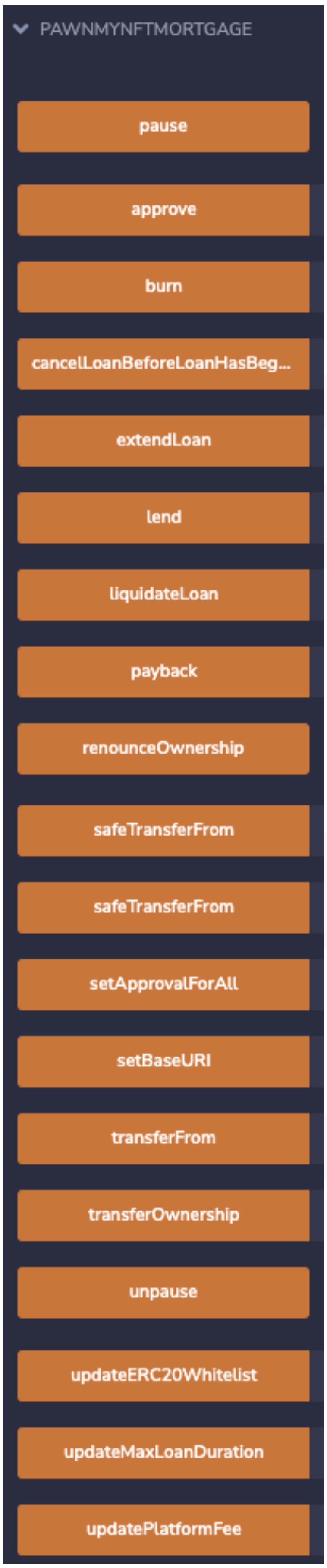
PawnMyNFTMortgage

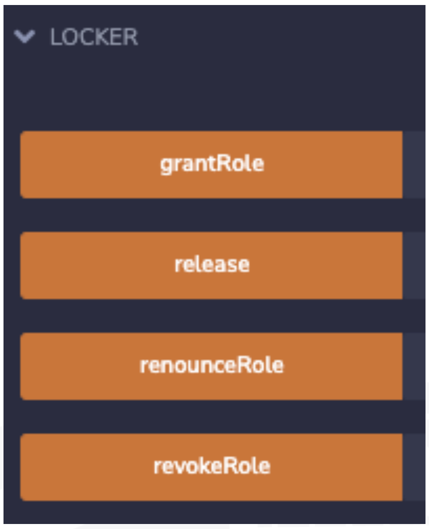
Functions
[✓] balanceOf(address) is present
[✓] balanceOf(address) -> () (correct return value)
[✓] balanceOf(address) is view
[✓] ownerOf(uint256) is present
[✓] ownerOf(uint256) -> () (correct return value)
[✓] ownerOf(uint256) is view
[✓] safeTransferFrom(address,address,uint256,bytes) is present
[✓] safeTransferFrom(address,address,uint256,bytes) -> () (correct return type)
[✓] Transfer(address,address,uint256) is emitted
[✓] safeTransferFrom(address,address,uint256) is present
[✓] safeTransferFrom(address,address,uint256) -> () (correct return type)
[✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
[✓] transferFrom(address,address,uint256) -> () (correct return type)
[✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
[✓] approve(address,uint256) -> () (correct return type)
[✓] Approval(address,address,uint256) is emitted
[✓] setApprovalForAll(address,bool) is present

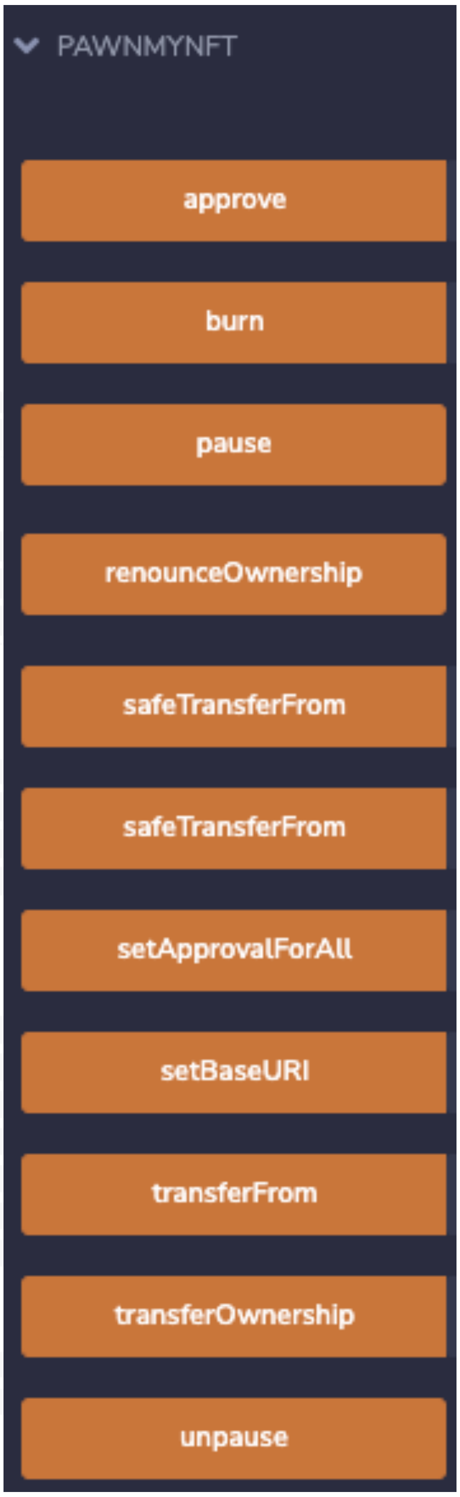
```
    [✓] setApprovalForAll(address,bool) -> () (correct return type)
    [✓] ApprovalForAll(address,address,bool) is emitted
[✓] getApproved(uint256) is present
    [✓] getApproved(uint256) -> () (correct return value)
    [✓] getApproved(uint256) is view
[✓] isApprovedForAll(address,address) is present
    [✓] isApprovedForAll(address,address) -> () (correct return value)
    [✓] isApprovedForAll(address,address) is view
[✓] supportsInterface(bytes4) is present
    [✓] supportsInterface(bytes4) -> () (correct return value)
    [✓] supportsInterface(bytes4) is view
[✓] name() is present
    [✓] name() -> () (correct return value)
    [✓] name() is view
[✓] symbol() is present
    [✓] symbol() -> () (correct return value)
[✓] tokenURI(uint256) is present
    [✓] tokenURI(uint256) -> () (correct return value)

## Check events
[✓] Transfer(address,address,uint256) is present
    [✓] parameter 0 is indexed
    [✓] parameter 1 is indexed
    [✓] parameter 2 is indexed
[✓] Approval(address,address,uint256) is present
    [✓] parameter 0 is indexed
    [✓] parameter 1 is indexed
    [✓] parameter 2 is indexed
[✓] ApprovalForAll(address,address,bool) is present
    [✓] parameter 0 is indexed
    [✓] parameter 1 is indexed
```

Write functions of contract







Deployer cannot burn or lock user funds

File	Name	Exist	Teste d	Verified
PawnMyNFT	cannot lock	✓	✓	✓
	cannot burn	✓	✓	✗
PawnMyNFTM ortgage	cannot lock	✓	✓	✗
	cannot burn	✓	✓	✗

Comments:

v1.0

- Deployer can set maximumLoanDuration to the lowest
- Deployer can set platformFee up to 1000
    - There is not limitation downwards
- Deployer can lock by pausing the contract

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
PawnMyNFT	Deployer cannot pause	✓	✓	✗
PawnMyNFT Mortgage	Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Deployer can pause contract
  - Following functions are not callable if contract is paused by the owner
    - PawnMyNFTMortgage
      - lend
      - Payback
    - PawnMyNFT
      - _beforeTokenTransfer
        - That function is used in ERC721 following functions
          - _mint
          - _burn
          - _transfer

Overall checkup (Smart Contract Security)

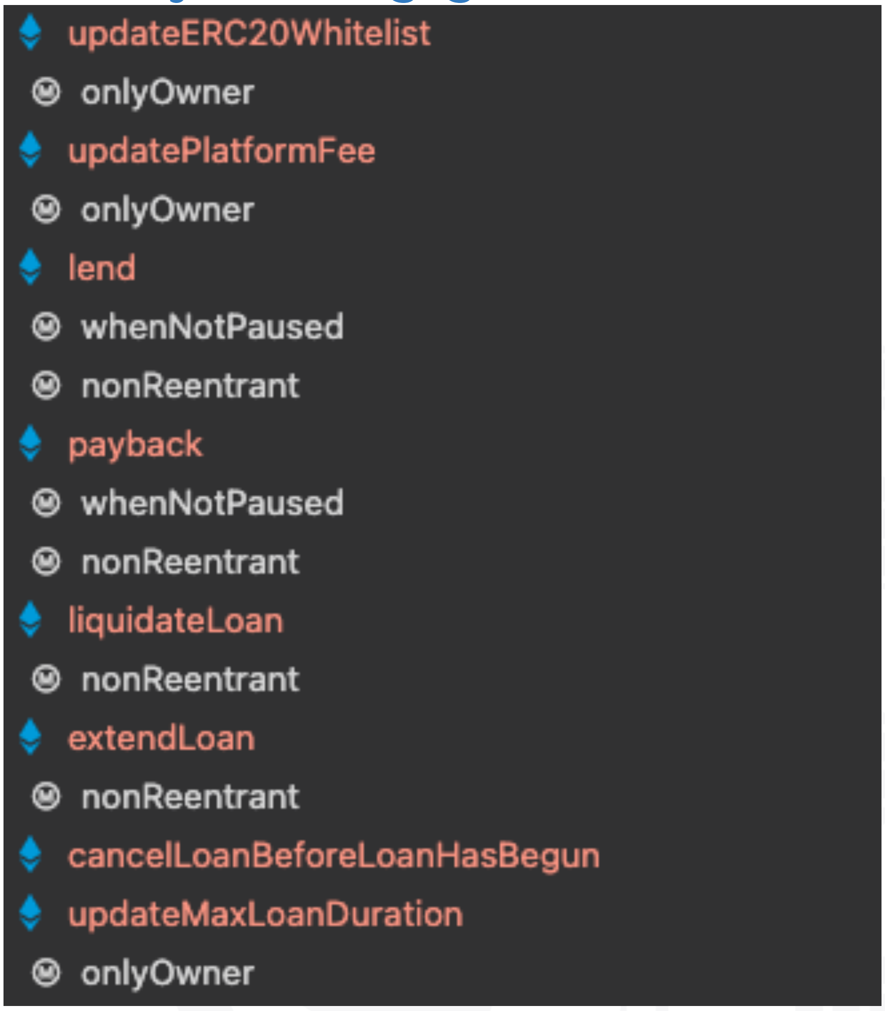
Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

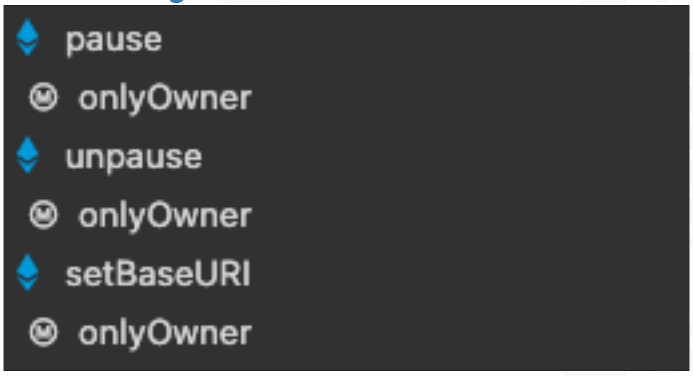
Modifiers

PawnMyNFTMortgage



- updateERC20Whitelist
  - onlyOwner
- updatePlatformFee
  - onlyOwner
- lend
  - whenNotPaused
  - nonReentrant
- payback
  - whenNotPaused
  - nonReentrant
- liquidateLoan
  - nonReentrant
- extendLoan
  - nonReentrant
- cancelLoanBeforeLoanHasBegun
- updateMaxLoanDuration
  - onlyOwner

PawnMyNFT

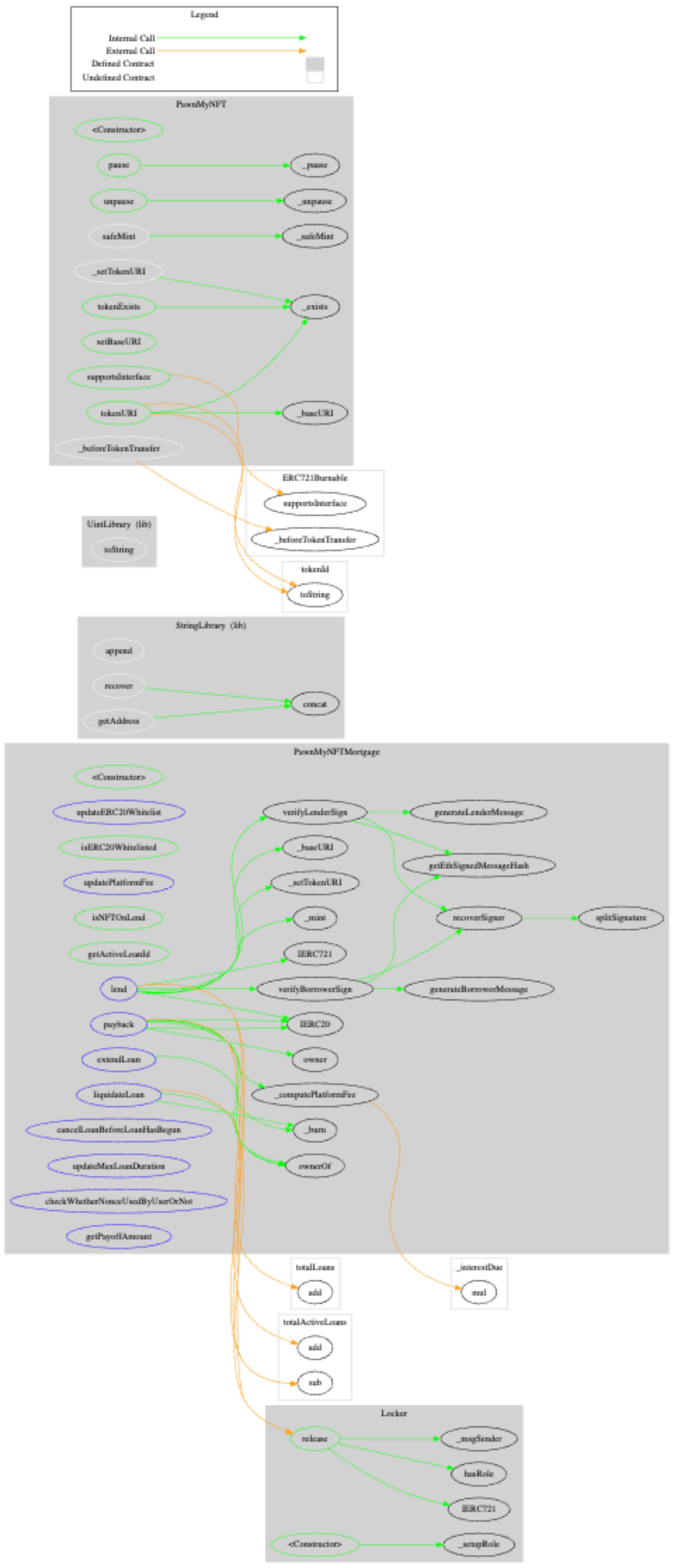


- pause
  - onlyOwner
- unpause
  - onlyOwner
- setBaseURI
  - onlyOwner

Comments

- Deployer can set following state variables without any limitations
  - maximumLoanDuration
- Deployer can enable/disable following state variables
  - _paused

CallGraph



Legend
Internal Call
External Call
Defined Contract
Undefined Contract
PawnMyNFT
<Constructor>
pause
_pause
unpause
_unpause
safeMint
_safeMint
_setTokenURI
tokenExists
_exists
setBaseURI
supportsInterface
tokenURI
_baseURI
_beforeTokenTransfer
ERC721Burnable
supportsInterface
_beforeTokenTransfer
UintLibrary (lib)
toString
tokenId
toString
StringLibrary (lib)
append
recover
concat
getAddress
PawnMyNFTMortgage
<Constructor>
updateERC20Whitelist
isERC20Whitelisted
updatePlatformFee
isNFTOnLend
getActiveLoanId
lend
payback
extendLoan
liquidateLoan
cancelLoanBeforeLoanHasBegun
updateMaxLoanDuration
checkWhetherNonceUsedByUserOrNot
getPayoffAmount
verifyLenderSign
_baseURI
_setTokenURI
_mint
IERC721
verifyBorrowerSign
IERC20
owner
_computePlatformFee
_burn
ownerOf
generateLenderMessage
getEthSignedMessageHash
recoverSigner
splitSignature
generateBorrowerMessage
totalLoans
add
_interestDue
mul
totalActiveLoans
add
sub
Locker
release
_msgSender
hasRole
IERC721
<Constructor>
_setupRole

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/PawnMyNFTMortgage.sol	1	—	565	485	410	17	171	🖥🧮🧪
📝	contracts/Locker.sol	1	—	37	33	27	1	22	🧮
📚	contracts/Lib.sol	2	—	127	101	93	1	220	🧮🧪
📝	contracts/PawnMyNFT.sol	1	—	97	79	65	1	53	—
📝📚	**Totals**	**5**	—	**826**	**698**	**595**	**20**	**466**	🖥🧮🧪

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	PawnMyNFTMortgage	Local variables shadowing	108	Rename the local variables that shadow another component

Informational issues

Issue	File	Type	Line	Description
#1	PawnMyNFT	Functions that are not used	38	Remove unused functions
#2	PawnMyNFTMortgage	Misspelling	69	Change nfnftTokenId to nftTokenId in event
#3	PawnMyNFTMortgage	Naming convention	155	If you using mixedCase naming convention, then continue to use it with all other variables Recommendation: Change _erc721Anderc20contracts to _erc721AndErc20Contracts

#4	PawnMyNFT	Imported files were not used	4, 6, 12	Imported files were not used in the code - ERC20 - SafeMath - IERC721
#5	PawnMyNFTMortgage	Imported files were not used	5, 6, 7, 8, 9, 10, 11, 15, 19, 20	Imported files were not used in the code - ERC721 - IERC721 - ERC721Enumerable - Pausable - ReentrancyGuard - Ownable - ERC721Burnable - Lib Every imported library except of PawnMyNFT and Locker are imported in PawnMyNFT and can be used in PawnMyNFTMortgage because of inheritance PawnMyNFTMortgage contract didn't use any functions from the Lib.sol file
#6	Locker	Imported files were not used	4, 6	Imported files were not used in the code - IERC20 - SafeERC20
#7	PawnMyNFTMortgage	Unnecessary import	12, 18	You don't need to import SafeMath library in solidity versionne 0.8.x and higher because it is already implemented. You can use the operators +, -, * and / as before. By removing library make sure to replace all SafeMath functionalities which you used (e.g. x.add(y)) with the normal operators above.
#8	PawnMyNFT	Unnecessary import	12	See description above

#9	PawnMyNFTMortgage	Comment style error	185, 189, 193, 400	If you start every require statement error message in uppercase after "::" make sure to continue with it instead of starting some messages with a lower case
#10	PawnMyNFTMortgage	Comments are missing	-	Some variables/functions were commented but other not. Please comment the rest of your code for readability
#11	PawnMyNFTMortgage	Wrong sorting of contract	-	We recommend you to order your source file in the following recommend way 1. Type declarations 2. State variables 3. Events 4. Functions

Audit Comments

09. January 2022:

- Please fix import issues because it seems to that it was only copy pasted in several files
    - Read informational issues above for more information
- PawnMyNFTMortgage must have allowance to spent tokens/nfts in loan contract for payback/lend function
    - Deployer can lock those functions by enable pausing state variable
- Read whole report for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**NOT PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	**PASSED**
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	**PASSED**
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	**PASSED**
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	**PASSED**
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	**PASSED**
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	**PASSED**
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	**PASSED**

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY