Worldes

Preliminary Comments

CertiK Assessed on Mar 18th, 2025

CertiK Assessed on Mar 18th, 2025

Worldes

These preliminary comments were prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/18/2025	N/A

CODEBASE

WorldesIO Github

WorldesPropertyRights WorldesApprove DVMFactory DSPFactory WorldesDvmProxy WorldesDspProxy WorldesRWATokenFactory

View All in Codebase Page

COMMITS

- 1c96baaa789764a3a87376b08cd1b6f446fa9acd
- d4199a7d833053f78de6ae87d8aa416d9393ff58

View All in Codebase Page

Highlighted Centralization Risks

- Privileged role can mint tokens
- Transfers can be paused
- Initial owner token share is 100%
- Has blacklist/whitelist

Vulnerability Summary

9	2	0	0	2	0	5
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Pending

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
6	Major	1 Resolved, 1 Acknowledged, 4 Pending	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
3	Minor	1 Resolved, 1 Acknowledged, 1 Pending	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

0 Informational	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.
0 Discussion	The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | WORLDES

CODEBASE | WORLDES

Repository

WorldesIO Github

WorldesPropertyRights WorldesApprove DVMFactory DSPFactory WorldesDvmProxy WorldesDspProxy WorldesRWATokenFactory WorldesMineProxy ERC20Mine WorldesMineRegistry WorldesRWAToken

Commit

- 1c96baaa789764a3a87376b08cd1b6f446fa9acd
- d4199a7d833053f78de6ae87d8aa416d9393ff58

AUDIT SCOPE | WORLDES

107 files audited ● 22 files with Pending findings ● 1 file with Acknowledged findings ● 3 files with Resolved findings ● 81 files without findings

ID	Repo	File	SHA256 Checksum
● WPR	WorldesIO/smart-contract	asset/WorldesPropertyRights.sol	08876fe4b4dc21555453c78f35f52eebef5567c0cd1703125a8d8089772ca03a
● WRW	WorldesIO/smart-contract	asset/WorldesRWAToken.sol	a69c9bd85a8692d2861d823cb3b74e8af1bef77bed3d6d850bc24d3a3bcadda4
● DSW	WorldesIO/smart-contract	factory/DSPFactory.sol	8bf270ebd95c0dcf2d1b9030090ce55335e46ae572218979f7bbffaf4008e5a0
● DVW	WorldesIO/smart-contract	factory/DVMFactory.sol	87493ed908be90c521b9c4b6657dde064bc5f776e66c2a397d6035968ca1c302
● WRI	WorldesIO/smart-contract	helper/WorldesRouterHelper.sol	3f8ca0416dc1f597889cdc46a3460632a6aaa152f2c5a50e275b1bc7b71fad1f
● FRW	WorldesIO/smart-contract	libraries/FeeRateModel.sol	34fe31d7c876fef8eb16350875d183c461620ad2ffc9bd5d620ae00bb15deadf
● IOW	WorldesIO/smart-contract	libraries/InitializableOwnable.sol	4571a49ae59edbbea7cfb2ae8a96b6e8608eeafe2b2230822426804a6d690d96
● OWO	WorldesIO/smart-contract	libraries/Ownable.sol	a173ddc6ccc6e1d2c1c2a9d7343efe619ac2a06c45869f1f25ddf4eaf2424548
● IOI	WorldesIO/smart-contract	limitOrder/InitializableOwnable.sol	ac8ed40b9377611a26f23a53972f5354a2f8817b37f287cac8c5bd0158ef7dee
● WGO	WorldesIO/smart-contract	limitOrder/WorldesGaslessTrading.sol	6982fe96367acf6fcb655864fde924e2bffff67cfaffa980f29ca7c6dca5c301
● WLW	WorldesIO/smart-contract	limitOrder/WorldesLimitOrder.sol	deccefc6f334efeda93b3aae5bafe8a96c3f85f520d97977c9ab239318bceaa0
● WLI	WorldesIO/smart-contract	limitOrder/WorldesLimitOrderBot.sol	6d2c041bcbb369bb18bc1d878339389c3b87fb2f511317cdb5e2322bea1c91bc

ID	Repo	File	SHA256 Checksum
BMW	WorldesIO/smart-contract	mine/BaseMine.sol	21786adfe76300cac06c80711c03dfd18bd0b5d2cc665527660a7e338a5fb995
RVW	WorldesIO/smart-contract	mine/RewardVault.sol	32804ef86786212b458a1715098e04ca68485ab32466b234537bdb10316d4ceb
WMR	WorldesIO/smart-contract	mine/WorldesMineRegistry.sol	66380d1acc91e186d00d4f9f8ed588a5b4bfe19a2521ac490f2ac49a759d7a04
WAW	WorldesIO/smart-contract	proxy/WorldesApprove.sol	ea7c579bceeaeaa03341b09727542db93d4d5960a7d8a33cbb40fb86465818d3
WAI	WorldesIO/smart-contract	proxy/WorldesApproveProxy.sol	66139fc0ddcd3b123b708120ff05327b1e64aca0f8d5eb5c5e7f0e8fe479295d
WDI	WorldesIO/smart-contract	proxy/WorldesDvmProxy.sol	f4a00d91c5f2c527951a2ce67d6284fca38f1cdc5345fa403e5b5ee7a82277c3
WMP	WorldesIO/smart-contract	proxy/WorldesMineProxy.sol	7a4bfb4678ffb51d0117857f51673f02d3e75db903d18ac878edc694202c5dfb
WPW	WorldesIO/smart-contract	asset/WorldesPropertyRights.sol	e1d10d3413839cbd94204078524ca49edaa399e1d304345f96663a6e581cd8ab
WWA	WorldesIO/smart-contract	asset/WorldesRWAToken.sol	f373488894dcdc19a62e505217b6aaccfb89ad4957637bf8254f41c61bba5ef7
ECM	mainnet	contracts/mining/ERC20Mine.sol	dd6333a1d1e40213a9364e24e0786b9aefc1c799175b90ed5591ee6e188f6336
WRA	WorldesIO/smart-contract	asset/WRAToken.sol	691331364cfe1812f1afd93fbf3cf62f9e80775db3ddba48f3c519c7ea317422
DMW	WorldesIO/smart-contract	libraries/DecimalMath.sol	9c062520ead8bf696b851d01cb670ee3e14db5d891ba43546439f8eb88162627
DSO	WorldesIO/smart-contract	stablePool/implements/DSPFunding.sol	838c7a012fd9e9d8bb45182b2c807645998e63b6959c48b6eb52a7dc6239bfa9
DPV	WorldesIO/smart-contract	stablePool/implements/DSPVault.sol	0b67bf0170978478d44e07b8011300bdff228c73264ffb6318301b188d74abb2
WRT	WorldesIO/smart-contract	asset/WorldesRWATokenFactory.sol	95f75fb9cf4cbc79de9af23e99e95ed80c96031018de8d38957b9b9fadcaf703

ID	Repo	File	SHA256 Checksum
IER	WorldesIO/smart-contract	interfaces/IERC20.sol	f0294e8517da4078b9f30c12946c2398c74d53e8117968ba5a4430c68ab9273f
IWE	WorldesIO/smart-contract	interfaces/IWETH.sol	60533273f1c41babb10978d7f1f5adb06dddbdedfef645c0634312212cd83ba9
IWW	WorldesIO/smart-contract	interfaces/IWorldes.sol	84fa99c992dc2b200f9df65043a4455be37490b672985f84631e9dc89aa209d8
IWA	WorldesIO/smart-contract	interfaces/IWorldesApprove.sol	8e682187b438d1d4d3bab050e9cd338ca625939a789cfe41fa6f8ba99155ccb9
IWP	WorldesIO/smart-contract	interfaces/IWorldesApproveProxy.sol	d896e0c8196de85aa79862d891aba320145e4de697b4ef2a077145f31b93d872
IWC	WorldesIO/smart-contract	interfaces/IWorldesCallee.sol	b0ee88f628dabb313320235a865bcc6c613064e28394acc5194728aaaad6a6b6
IWD	WorldesIO/smart-contract	interfaces/IWorldesDvmProxy.sol	b21c6c07b6770916a85036dace58735dd89bafb606f08bc8159d447cf46b6525
CFW	WorldesIO/smart-contract	libraries/CloneFactory.sol	45aa003726db6165049c2a8830885afd99991c4564e507019183848edcb9c827
PMP	WorldesIO/smart-contract	libraries/PMMPricing.sol	7cb428a049b670b8843fd921069c8c22ec5951795f5ed71d83f7d923a0344022
RGW	WorldesIO/smart-contract	libraries/ReentrancyGuard.sol	fc7b12d0faeb7840afc9e6936dcb536b509278bbcc8cb21e6fb234ce4de61033
SEC	WorldesIO/smart-contract	libraries/SafeERC20.sol	1e1a8b4c6fba18221d1e18ae10a359f4ef6369e8fe8b56a402c3b4deebbaa47d
SMW	WorldesIO/smart-contract	libraries/SafeMath.sol	475f6c28b8c9ea9a60294fbb94b8466cc9a8820d3f269ae75b55bc2107bce38c
UEC	WorldesIO/smart-contract	libraries/UniversalERC20.sol	4b293c4cc76aba9d5c4e39c80673dd08318e5fd6c4afe6a458c0862d79f34ead
WMW	WorldesIO/smart-contract	libraries/WorldesMath.sol	74a0f422f96919a7eed1f04dd81bbc14d59a80017bca279cbb442b234073b083
IEW	WorldesIO/smart-contract	limitOrder/interfaces/IERC1271Wallet.sol	837041b47cd0df302f82f41cb438ecd86fa66ec1d16f73894a05b38953fb82be

ID	Repo	File	SHA256 Checksum
IWO	WorldesIO/smart-contract	limitOrder/interfaces/IWorldesApproveProxy.sol	d18cb7276cbf9f5c8edd55b26a8d5e2deac9aec87c705524ba3a13c1424d2c9b
ADW	WorldesIO/smart-contract	limitOrder/ArgumentsDecoder.sol	32df2417d41b79bdba2725abeb53baf2628700ebf3a3774daf04b6c120a1b2d4
ERC	WorldesIO/smart-contract	mine/ERC20Mine.sol	d70d4435a5341f6c0c38ff1f9cdd9bb8f0ff5db603618e7e1606d3da9eac5ce4
WDW	WorldesIO/smart-contract	proxy/WorldesDspProxy.sol	c5364c4e0ae5f24df934db3395879895a2e412761b8dbb36441225668016eaed
DSI	WorldesIO/smart-contract	stablePool/implements/DSP.sol	ece92386cd6af4b0035ff57f1c5546a3d5867f4cfe933d44dcdc4963f6aee637
DPS	WorldesIO/smart-contract	stablePool/implements/DSPStorage.sol	4cca7f9ea0b549dbe86942041031eb65ac5ad73210d8ecf2c15612b9ad53a3c7
DPT	WorldesIO/smart-contract	stablePool/implements/DSPTrader.sol	1fa7a8d57d14532fd9e77a05866ae01eb065700aeae3ed9f9b9e26a92289b374
IDS	WorldesIO/smart-contract	stablePool/interfaces/IDSP.sol	24fcac4202ddae710b0b2b9ec0324476d29333b15aac1a8296e84abef1037c35
DVI	WorldesIO/smart-contract	vendingMachine/implements/DVM.sol	1c851e44ffbfab5e73a53a8f57056d604eb0c2a6806bf356794c7eaeb3aa575a
DVO	WorldesIO/smart-contract	vendingMachine/implements/DVMFunding.sol	855e6bdfe54a66933ddfcbe6e39cb6268bfe1760c285e1116661795103c624c6
DMS	WorldesIO/smart-contract	vendingMachine/implements/DVMStorage.sol	939cf6bd7be76f342358863a07e2bb45e43d7886c972ce69e044f2a347cead67
DMT	WorldesIO/smart-contract	vendingMachine/implements/DVMTrader.sol	3adbbc7294a56c3fa38addb2feefd831a535bb99fb1071eb7f12068843a5c373
DMV	WorldesIO/smart-contract	vendingMachine/implements/DVMVault.sol	39ed5022487915c38999c94e778c2fa800dce52b29c8842de2741dc4f3138d41
IDV	WorldesIO/smart-contract	vendingMachine/interfaces/IDVM.sol	cecb2725c254cc7a4f7aab3c6be9a636a10994c7317d567a585a0c93a9279f7e
WRO	WorldesIO/smart-contract	asset/WRAToken.sol	691331364cfe1812f1afd93fbf3cf62f9e80775db3ddba48f3c519c7ea317422

ID	Repo	File	SHA256 Checksum
WRF	WorldesIO/smart-contract	asset/WorldesRWATokenFactory.sol	df23b77ab498a84c486795178d02b6b07157529eb23f33eb1cd6c787b56f76f6
DSP	WorldesIO/smart-contract	factory/DSPFactory.sol	8bf270ebd95c0dcf2d1b9030090ce55335e46ae572218979f7bbffaf4008e5a0
DVM	WorldesIO/smart-contract	factory/DVMFactory.sol	87493ed908be90c521b9c4b6657dde064bc5f776e66c2a397d6035968ca1c302
WRH	WorldesIO/smart-contract	helper/WorldesRouterHelper.sol	3f8ca0416dc1f597889cdc46a3460632a6aaa152f2c5a50e275b1bc7b71fad1f
IEC	WorldesIO/smart-contract	interfaces/IERC20.sol	f0294e8517da4078b9f30c12946c2398c74d53e8117968ba5a4430c68ab9273f
IWT	WorldesIO/smart-contract	interfaces/IWETH.sol	60533273f1c41babb10978d7f1f5adb06dddbdedfef645c0634312212cd83ba9
IWI	WorldesIO/smart-contract	interfaces/IWorldes.sol	84fa99c992dc2b200f9df65043a4455be37490b672985f84631e9dc89aa209d8
IAW	WorldesIO/smart-contract	interfaces/IWorldesApprove.sol	8e682187b438d1d4d3bab050e9cd338ca625939a789cfe41fa6f8ba99155ccb9
IAP	WorldesIO/smart-contract	interfaces/IWorldesApproveProxy.sol	d896e0c8196de85aa79862d891aba320145e4de697b4ef2a077145f31b93d872
ICW	WorldesIO/smart-contract	interfaces/IWorldesCallee.sol	b0ee88f628dabb313320235a865bcc6c613064e28394acc5194728aaaad6a6b6
IDP	WorldesIO/smart-contract	interfaces/IWorldesDvmProxy.sol	b21c6c07b6770916a85036dace58735dd89bafb606f08bc8159d447cf46b6525
CFI	WorldesIO/smart-contract	libraries/CloneFactory.sol	45aa003726db6165049c2a8830885afd99991c4564e507019183848edcb9c827
DMI	WorldesIO/smart-contract	libraries/DecimalMath.sol	9c062520ead8bf696b851d01cb670ee3e14db5d891ba43546439f8eb88162627
FRM	WorldesIO/smart-contract	libraries/FeeRateModel.sol	34fe31d7c876fef8eb16350875d183c461620ad2ffc9bd5d620ae00bb15deadf
IOO	WorldesIO/smart-contract	libraries/InitializableOwnable.sol	4571a49ae59edbbea7cfb2ae8a96b6e8608eeafe2b2230822426804a6d690d96

ID	Repo	File	SHA256 Checksum
OWN	WorldesIO/smart-contract	libraries/Ownable.sol	a173ddc6ccc6e1d2c1c2a9d7343efe619ac2a06c45869f1f25ddf4eaf2424548
PMM	WorldesIO/smart-contract	libraries/PMMPricing.sol	7cb428a049b670b8843fd921069c8c22ec5951795f5ed71d83f7d923a0344022
RGI	WorldesIO/smart-contract	libraries/ReentrancyGuard.sol	fc7b12d0faeb7840afc9e6936dcb536b509278bbcc8cb21e6fb234ce4de61033
SER	WorldesIO/smart-contract	libraries/SafeERC20.sol	1e1a8b4c6fba18221d1e18ae10a359f4ef6369e8fe8b56a402c3b4deebbaa47d
SMI	WorldesIO/smart-contract	libraries/SafeMath.sol	475f6c28b8c9ea9a60294fbb94b8466cc9a8820d3f269ae75b55bc2107bce38c
UER	WorldesIO/smart-contract	libraries/UniversalERC20.sol	4b293c4cc76aba9d5c4e39c80673dd08318e5fd6c4afe6a458c0862d79f34ead
WMI	WorldesIO/smart-contract	libraries/WorldesMath.sol	74a0f422f96919a7eed1f04dd81bbc14d59a80017bca279cbb442b234073b083
IEO	WorldesIO/smart-contract	limitOrder/interfaces/IERC1271Wallet.sol	837041b47cd0df302f82f41cb438ecd86fa66ec1d16f73894a05b38953fb82be
IAO	WorldesIO/smart-contract	limitOrder/interfaces/IWorldesApproveProxy.sol	d18cb7276cbf9f5c8edd55b26a8d5e2deac9aec87c705524ba3a13c1424d2c9b
ADO	WorldesIO/smart-contract	limitOrder/ArgumentsDecoder.sol	32df2417d41b79bdba2725abeb53baf2628700ebf3a3774daf04b6c120a1b2d4
IIO	WorldesIO/smart-contract	limitOrder/InitializableOwnable.sol	ac8ed40b9377611a26f23a53972f5354a2f8817b37f287cac8c5bd0158ef7dee
WGT	WorldesIO/smart-contract	limitOrder/WorldesGaslessTrading.sol	6982fe96367acf6fcb655864fde924e2bffff67cfaffa980f29ca7c6dca5c301
WLO	WorldesIO/smart-contract	limitOrder/WorldesLimitOrder.sol	deccefc6f334efeda93b3aae5bafe8a96c3f85f520d97977c9ab239318bceaa0
WLB	WorldesIO/smart-contract	limitOrder/WorldesLimitOrderBot.sol	6d2c041bcbb369bb18bc1d878339389c3b87fb2f511317cdb5e2322bea1c91bc
BMI	WorldesIO/smart-contract	mine/BaseMine.sol	21786adfe76300cac06c80711c03dfd18bd0b5d2cc665527660a7e338a5fb995

ID	Repo	File	SHA256 Checksum
ERM	WorldesIO/smart-contract	mine/ERC20Mine.sol	d70d4435a5341f6c0c38ff1f9cdd9bb8f0ff5db603618e7e1606d3da9eac5ce4
RVI	WorldesIO/smart-contract	mine/RewardVault.sol	32804ef86786212b458a1715098e04ca68485ab32466b234537bdb10316d4ceb
WMO	WorldesIO/smart-contract	mine/WorldesMineRegistry.sol	66380d1acc91e186d00d4f9f8ed588a5b4bfe19a2521ac490f2ac49a759d7a04
WAO	WorldesIO/smart-contract	proxy/WorldesApprove.sol	ea7c579bceeaeaa03341b09727542db93d4d5960a7d8a33cbb40fb86465818d3
WAP	WorldesIO/smart-contract	proxy/WorldesApproveProxy.sol	66139fc0ddcd3b123b708120ff05327b1e64aca0f8d5eb5c5e7f0e8fe479295d
WDP	WorldesIO/smart-contract	proxy/WorldesDspProxy.sol	c5364c4e0ae5f24df934db3395879895a2e412761b8dbb36441225668016eaed
WDO	WorldesIO/smart-contract	proxy/WorldesDvmProxy.sol	f4a00d91c5f2c527951a2ce67d6284fca38f1cdc5345fa403e5b5ee7a82277c3
WPI	WorldesIO/smart-contract	proxy/WorldesMineProxy.sol	7a4bfb4678ffb51d0117857f51673f02d3e75db903d18ac878edc694202c5dfb
DPP	WorldesIO/smart-contract	stablePool/implements/DSP.sol	ece92386cd6af4b0035ff57f1c5546a3d5867f4cfe933d44dcdc4963f6aee637
DSF	WorldesIO/smart-contract	stablePool/implements/DSPFunding.sol	061417ed699bd527e6d1617fd1f0f3ef1a2525e49fdecf3c5a8696c591ad8a7a
DSS	WorldesIO/smart-contract	stablePool/implements/DSPStorage.sol	4cca7f9ea0b549dbe86942041031eb65ac5ad73210d8ecf2c15612b9ad53a3c7
DST	WorldesIO/smart-contract	stablePool/implements/DSPTrader.sol	1fa7a8d57d14532fd9e77a05866ae01eb065700aeae3ed9f9b9e26a92289b374
DSV	WorldesIO/smart-contract	stablePool/implements/DSPVault.sol	0b67bf0170978478d44e07b8011300bdff228c73264ffb6318301b188d74abb2
IDW	WorldesIO/smart-contract	stablePool/interfaces/IDSP.sol	24fcac4202ddae710b0b2b9ec0324476d29333b15aac1a8296e84abef1037c35
DMM	WorldesIO/smart-contract	vendingMachine/implements/DVM.sol	1c851e44ffbfab5e73a53a8f57056d604eb0c2a6806bf356794c7eaeb3aa575a

ID	Repo	File	SHA256 Checksum
DVF	WorldesIO/smart-contract	vendingMachine/implements/DVMFunding.sol	855e6bdfe54a66933ddfcbe6e39cb6268bfe1760c285e1116661795103c624c6
DVS	WorldesIO/smart-contract	vendingMachine/implements/DVMStorage.sol	939cf6bd7be76f342358863a07e2bb45e43d7886c972ce69e044f2a347cead67
DVT	WorldesIO/smart-contract	vendingMachine/implements/DVMTrader.sol	3adbbc7294a56c3fa38addb2feefd831a535bb99fb1071eb7f12068843a5c373
DVV	WorldesIO/smart-contract	vendingMachine/implements/DVMVault.sol	39ed5022487915c38999c94e778c2fa800dce52b29c8842de2741dc4f3138d41
IDM	WorldesIO/smart-contract	vendingMachine/interfaces/IDVM.sol	cecb2725c254cc7a4f7aab3c6be9a636a10994c7317d567a585a0c93a9279f7e

APPROACH & METHODS | WORLDES

This report has been prepared for Worldes to discover issues and vulnerabilities in the source code of the Worldes project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | WORLDES

Overview

Worldes utilizes an advanced market maker algorithm known as Proactive Market Maker (PMM), designed to reduce risks for liquidity providers (LPs) and maintain portfolio stability. Worldes offers two key features: Stable Pool and Vending Machine.

Vending Machine: Designed for long-tail tokens, the DVM provides a groundbreaking solution to ensure market liquidity for less widely traded assets.

Stable Pool: Tailored for stablecoins, the DSP delivers efficient and stable liquidity management, catering specifically to the needs of stablecoin trading pairs.

Disclaimer

Only the differences from the Worldes' commit ca7d926e9387f304ccea4ab72860e882242109af and the forked well-known PMM repos listed below were reviewed. The audit scope only includes the delta part between these two. The detailed file list is in the above audit scope section.

Exchange Commit: d055deeace98871c05a0ec30537e631d58c60224

Limit Order Commit: 44b5a7d7aed10f9ac38c743e7b53ba39b0c1367c

FINDINGS | WORLDES

9	0	6	0	3	0	0
Total Findings	Critical	Major	Medium	Minor	Informational	Discussion

This report has been prepared to discover issues and vulnerabilities for Worldes. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
ASS-01	**Initial Token Distribution**	**Centralization**	**Major**	**Acknowledged**
ASS-02	**Centralized Balance Manipulation**	**Centralization**	**Major**	**Pending**
ECM-01	**Withdrawal Centralized Risk In `ERC20Mine` Contract**	**Centralization**	**Major**	**Pending**
SMA-01	**Centralization Related Risks**	**Centralization**	**Major**	**Pending**
WIO-02	Initial Depositor Can Set Quote Target To Zero To Affect Pool Functionality	Logical Issue	Major	Resolved
WWA-01	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Pending**
BMM-01	Block.Timestamp Should Be Used Instead Of Block.Number On Arbitrum	Logical Issue	Minor	Pending
WPR-01	Invalid Use Of Access Control Modifier	Logical Issue	Minor	Resolved
WPW-01	Lack Of Existence Check In `setTokenURI()` Function	Logical Issue	Minor	Acknowledged

ASS-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	Major	asset/WRAToken.sol (pre): 9; asset/WorldesRWAToken.sol (pre): 42	Acknowledged

Description

In the `WorldRealAssetToken` contract, all of the WRA tokens are sent to the contract deployer. This is a centralization risk because the deployer can distribute tokens without obtaining the consensus of the community. Any compromise to the address may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

In the `WorldesRWAToken` contract, the `initialSupply * 10 ** intitialDecimals` amount of the WESToken tokens are sent to the `to` address. This is a centralization risk because the address can distribute tokens without obtaining the consensus of the community. Any compromise to the address may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

Alleviation

[Worldes Team, 05/31/2024]: The team acknowledged this issue and stated that they will address the issue in the future, which will not be included in this audit engagement.

[CertiK, 05/31/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

ASS-02 | CENTRALIZED BALANCE MANIPULATION

Category	Severity	Location	Status
Centralization	Major	asset/WorldesPropertyRights.sol (pre): 92; asset/WorldesRWAToken.sol (pre): 82	Pending

Description

In the contract `WorldesRWAToken`, the role `_owner` has the authority to mint tokens via `mint()` to an arbitrary account. The total supply has a cap `_MAX_SUPPLY_`, while this cap can be increased by the `_WPR_ADDRESS_` address via the `setMaxSupply()` function.

In the contract `WorldesPropertyRights`, the role "minter" has the authority to mint tokens via `safeMint()` to an arbitrary account without sanity restriction.

Any compromise to the `_owner` and "minter" accounts may allow a hacker to take advantage of this authority and manipulate users' balances. For example, the hacker could also update his/her balance to a large number, sell these tokens, and cause the token price to drop.

Recommendation

We recommend the team makes efforts to restrict access to the private key of the privileged account. A strategy of multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to mint more tokens or engage in similar balance-related operations.
Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently *fully* resolve the risk:

Short Term:

A multi signature (⅔, ⅗) wallet *mitigate* the risk by avoiding a single point of key management failure.

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers' addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.

- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A DAO for controlling the operation *mitigate* the risk by applying transparency and decentralization.

- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the multi-signers' addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

The following actions can *fully* resolve the risk:

- Renounce the ownership and never claim back the privileged role.
  OR
- Remove the risky functionality.
  OR
- Add minting logic (such as a vesting schedule) to the contract instead of allowing the owner account to call the sensitive function directly.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Worldes Team, 03/17/2025]: The team acknowledged the issue and adopted the multisign solution to ensure the private key management process at the current stage. The WorldesPropertyRights contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x0a7f36ddeb9dceb4061dd088853f18d49971d4a61d6f719d69ca7444809e7e4c

The WorldesRWAToken contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe: https://arbiscan.io/tx/0x43a27958c9880ff8b82bd0ed4cef95e856b64565f3195f238ac051f1403f3578
- The 3 multisign addresses:

    1. EOA:0x4B1302f82cb0A96DbDeeb52F23f3848EFB2B7f7c
    2. EOA:0xA3627C24FBF5d4e2640948A4cB0e0451B6F3d324
    3. EOA:0xb52c9B17f9696c4A907786F4979021f45323f058

[CertiK, 03/17/2025]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team to periodically revisit the private key security management of all above-listed addresses.

ECM-01 | WITHDRAWAL CENTRALIZED RISK IN `ERC20Mine` CONTRACT

Category	Severity	Location	Status
Centralization	Major	contracts/mining/ERC20Mine.sol (ERC20Mine): 15	Pending

Description

In the `ERC20Mine` contract 0x6D8BC557570E84d1E9a8541D9beb4a0f3636e9B4, the `_owner` has the authority to set `_ROBOT_ADDRESS_` using the `setRobotAddress()` function. The `_ROBOT_ADDRESS_` is granted the authority to withdraw all staked tokens from the contract via the `withdrawByRobot()` function.

This centralized authority introduces a significant security risk: if either the `_owner` or the `_ROBOT_ADDRESS_` accounts are compromised, a hacker could exploit this authority to withdraw all staked tokens, potentially causing irreparable damage to the project and its stakeholders.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Worldes Team, 03/17/2025]: The team acknowledged the issue and adopted the multisign solution to ensure the private key management process at the current stage. The WorldesMineProxy contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0xf167fce751ea0fd0d30c25dc6acda1a3a0e9a148a56ac9b7d67f06fbcfb85f79
- The 3 multisign addresses:
  1. EOA:0x4B1302f82cb0A96DbDeeb52F23f3848EFB2B7f7c
  2. EOA:0xA3627C24FBF5d4e2640948A4cB0e0451B6F3d324
  3. EOA:0xb52c9B17f9696c4A907786F4979021f45323f058

[CertiK, 03/17/2025]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team to periodically revisit the private key security management of all above-listed addresses.

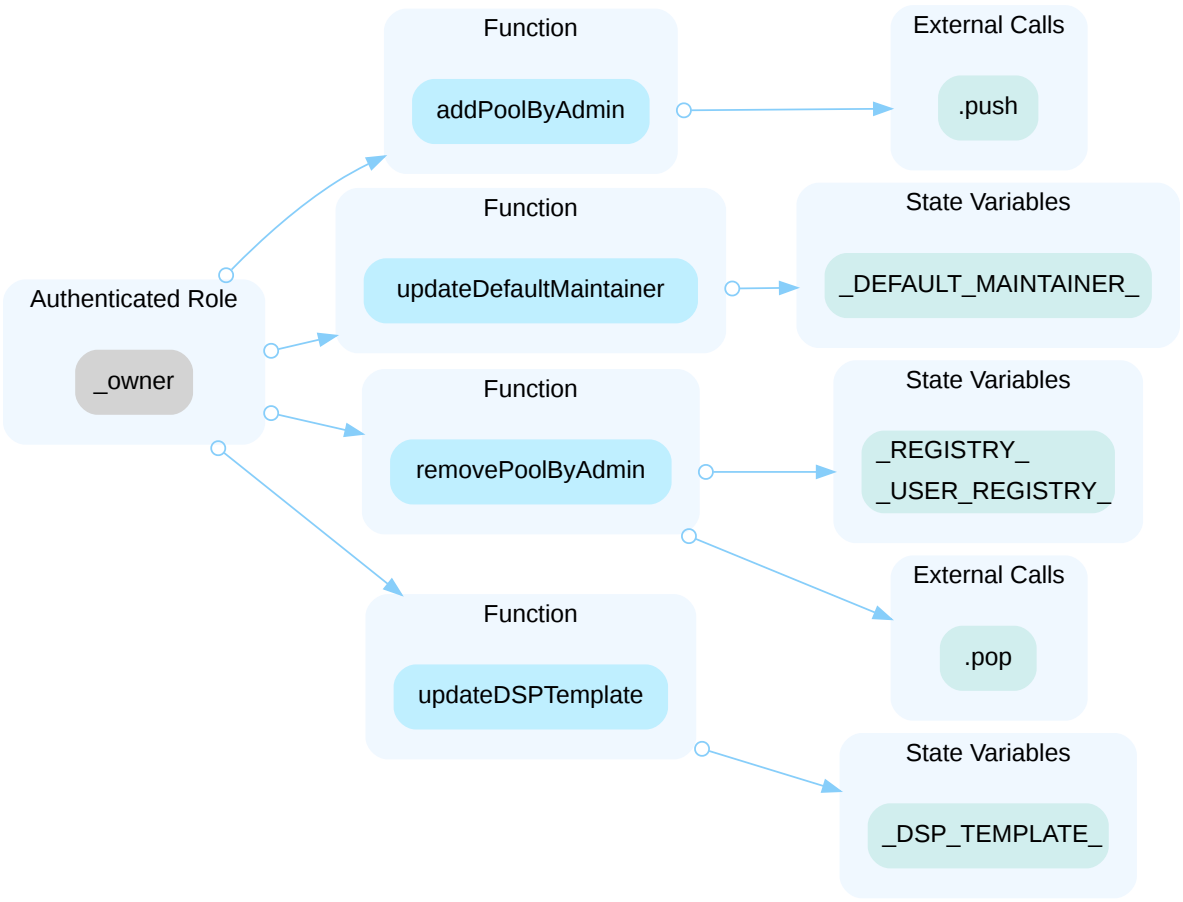
SMA-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Major	asset/WorldesPropertyRights.sol (pre): 62, 66, 70, 76, 81, 87, 92; asset/WorldesRWAToken.sol (pre): 45, 49, 53, 57, 61, 65, 69, 74, 78, 82; factory/DSPFactory.sol (pre): 93, 97, 101, 112; factory/DVMFactory.sol (pre): 99, 103, 107, 118; helper/WorldesRouterHelper.sol (pre): 106, 122; libraries/FeeRateModel.sol (pre): 23; libraries/InitializableOwnable.sol (pre): 45, 50; libraries/Ownable.sol (pre): 39, 44; limitOrder/InitializableOwnable.sol (pre): 45, 50; limitOrder/WorldesGaslessTrading.sol (pre): 81, 168, 173, 178; limitOrder/WorldesLimitOrder.sol (pre): 62, 115, 120, 125; limitOrder/WorldesLimitOrderBot.sol (pre): 46, 72, 102, 107, 112; mine/BaseMine.sol (pre): 148, 181, 203, 225, 242; mine/RewardVault.sol (pre): 37, 43; mine/WorldesMineRegistry.sol (pre): 43, 61, 94, 99; proxy/WorldesApprove.sol (pre): 48, 57, 63, 70; proxy/WorldesApproveProxy.sol (pre): 51, 56, 62, 67, 71; proxy/WorldesDvmProxy.sol (pre): 69, 73; proxy/WorldesMineProxy.sol (pre): 128; asset/WorldesPropertyRights.sol (pre2): 95, 103; asset/WorldesRWAToken.sol (pre2): 58	Pending

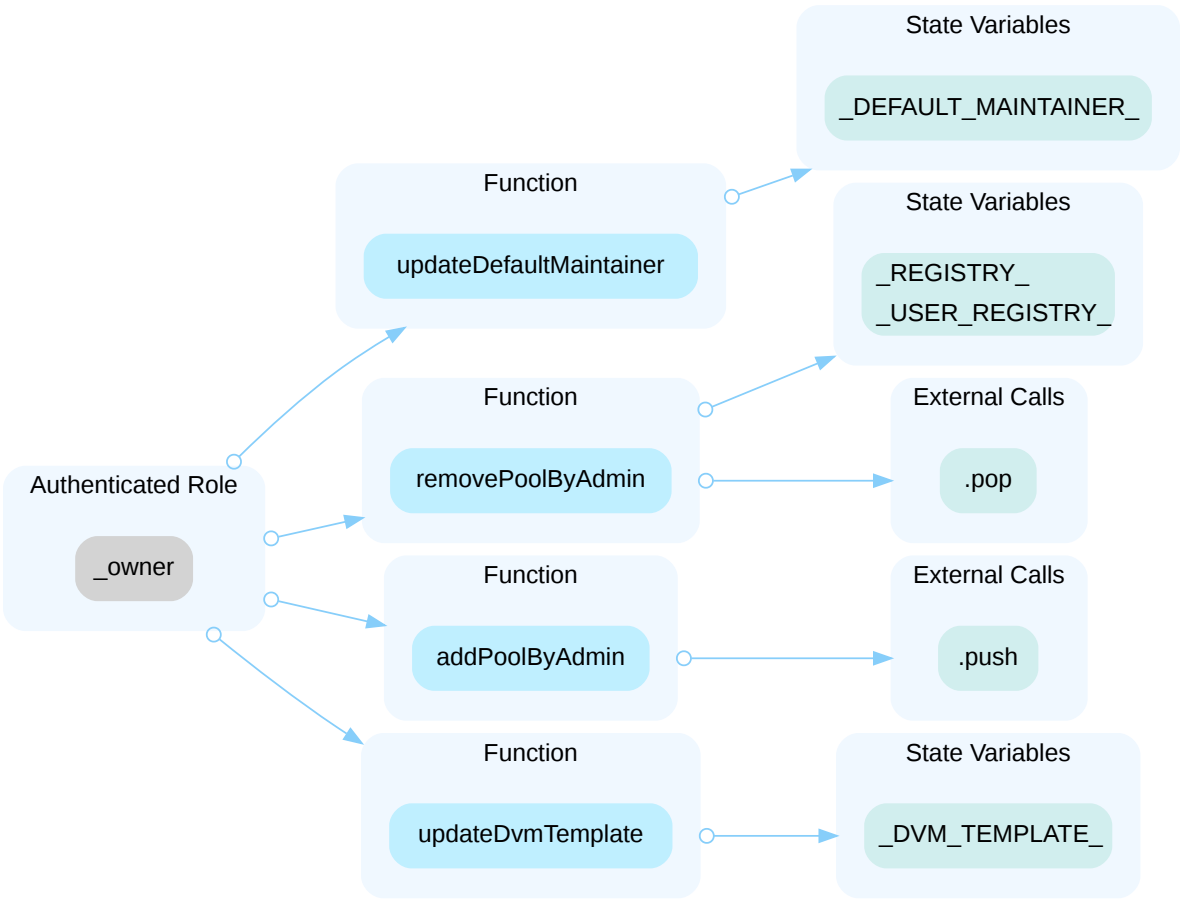
Description

Commit ca7d926e9387f304ccea4ab72860e882242109af

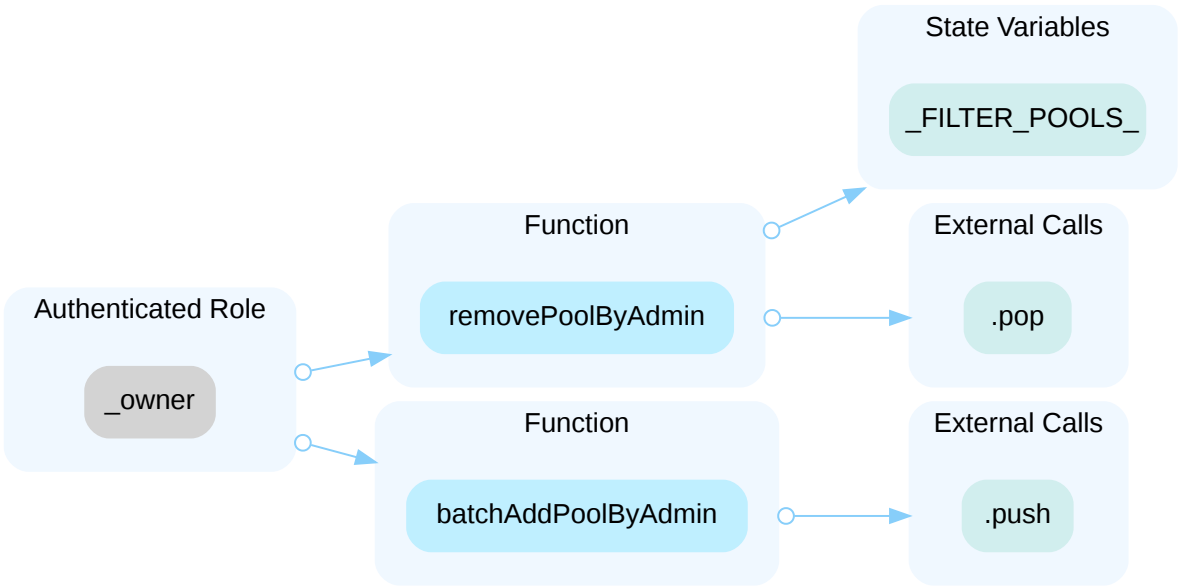
In the contract `DSPFactory` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and update `_DSP_TEMPLATE_`, `_DEFAULT_MAINTAINER_`, add or remove `_REGISTRY_`, `_USER_REGISTRY_`.



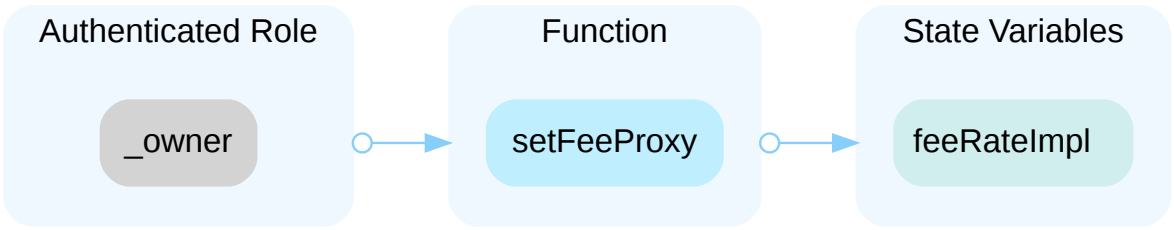
In the contract `DVMFactory` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and update `_DVM_TEMPLATE_`, `_DEFAULT_MAINTAINER_`, add or remove `_REGISTRY_`, `_USER_REGISTRY_`.



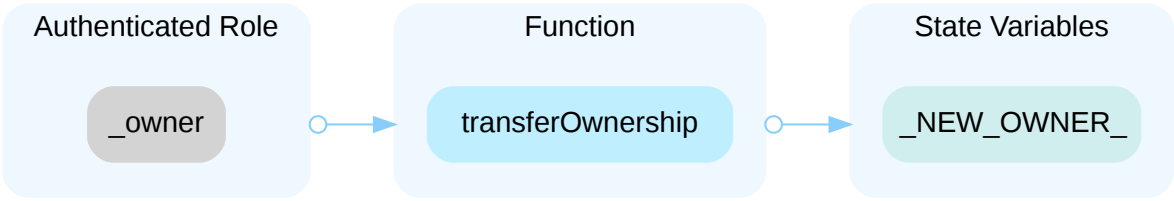
In the contract `WorldesRouterHelper` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove `_FILTER_POOLS_`.



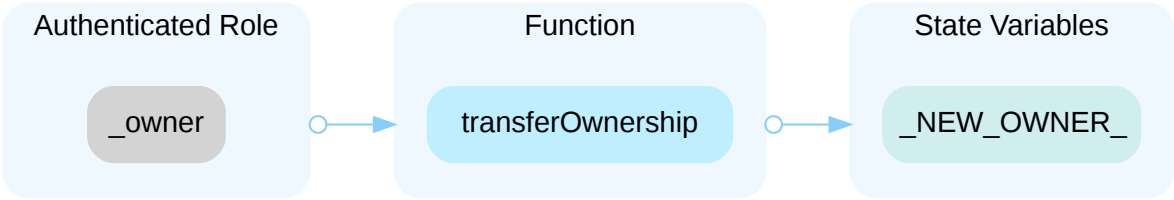
In the contract `FeeRateModel` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set `feeRateImpl`.



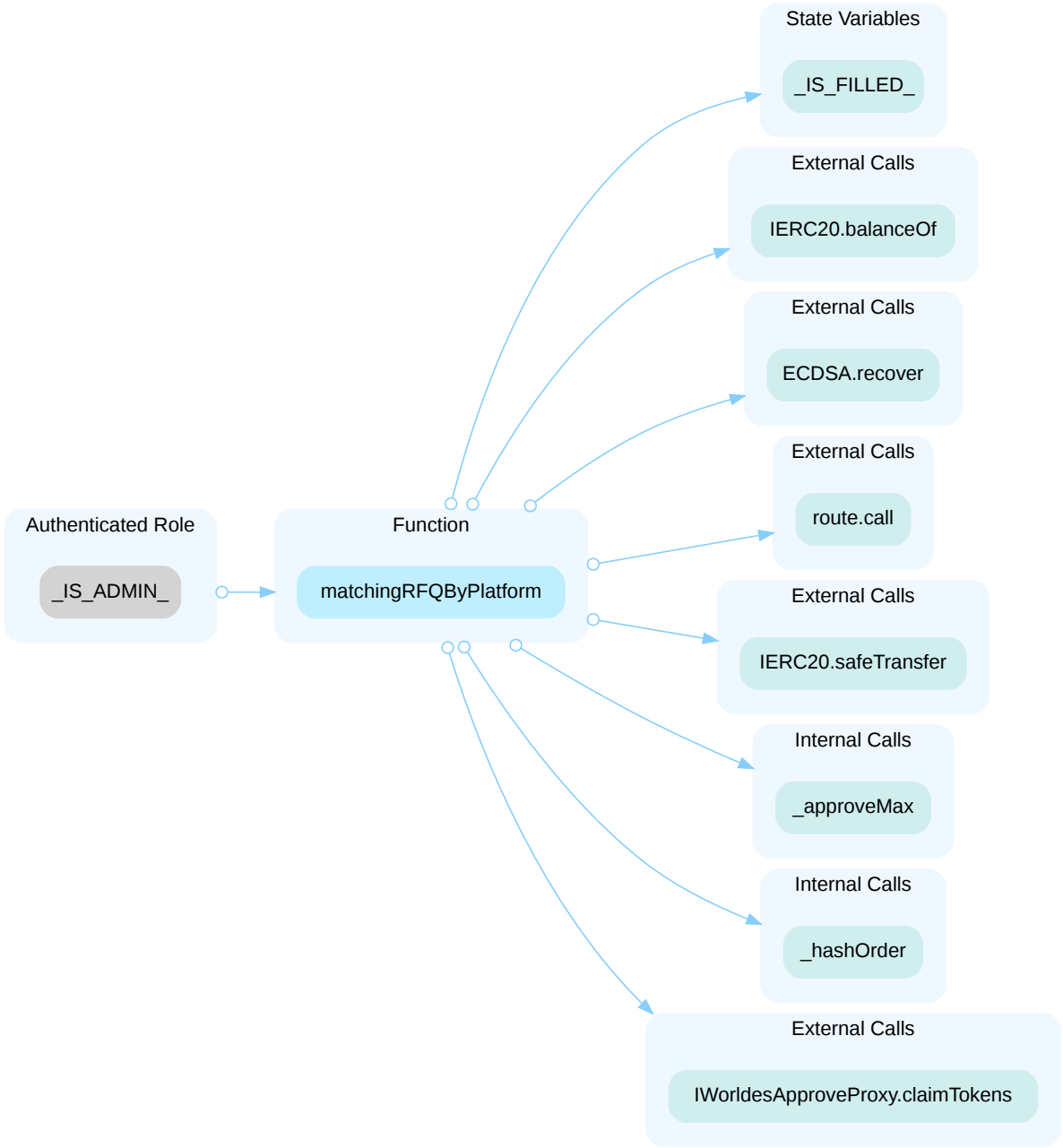
In the contract `InitializableOwnable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set `_NEW_OWNER_`.



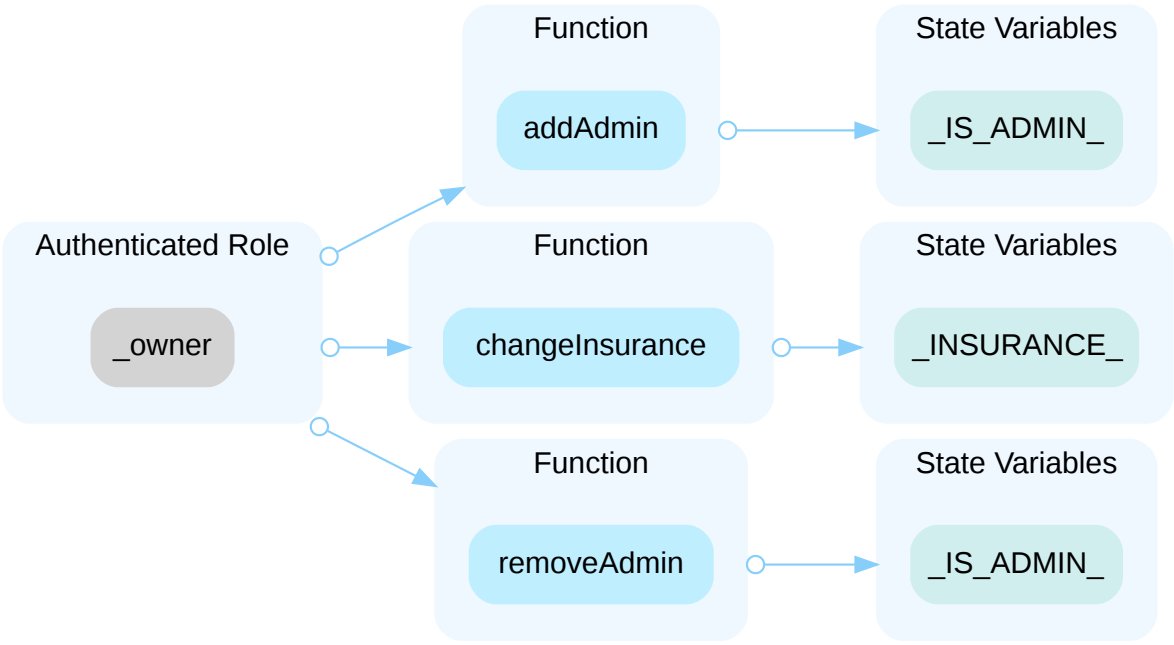
In the contract `Ownable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set `_NEW_OWNER_`.



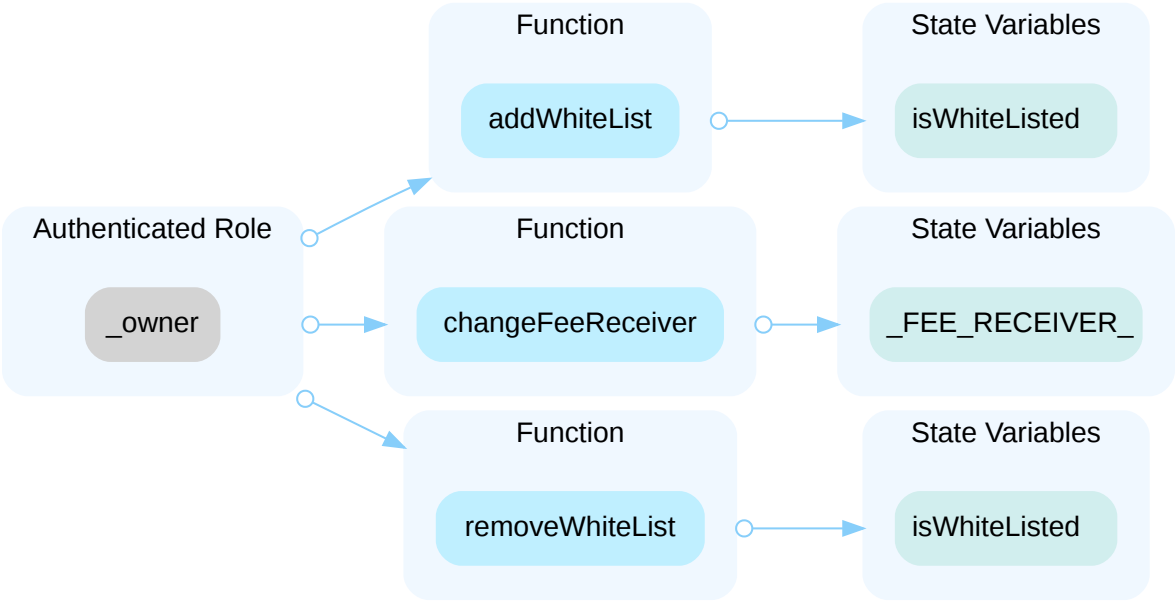
In the contract `WorldesGaslessTrading` the role `_IS_ADMIN_` has authority over the functions shown in the diagram below. Any compromise to the `_IS_ADMIN_` account may allow the hacker to take advantage of this authority and update critical variables and perform external calls to other contracts.



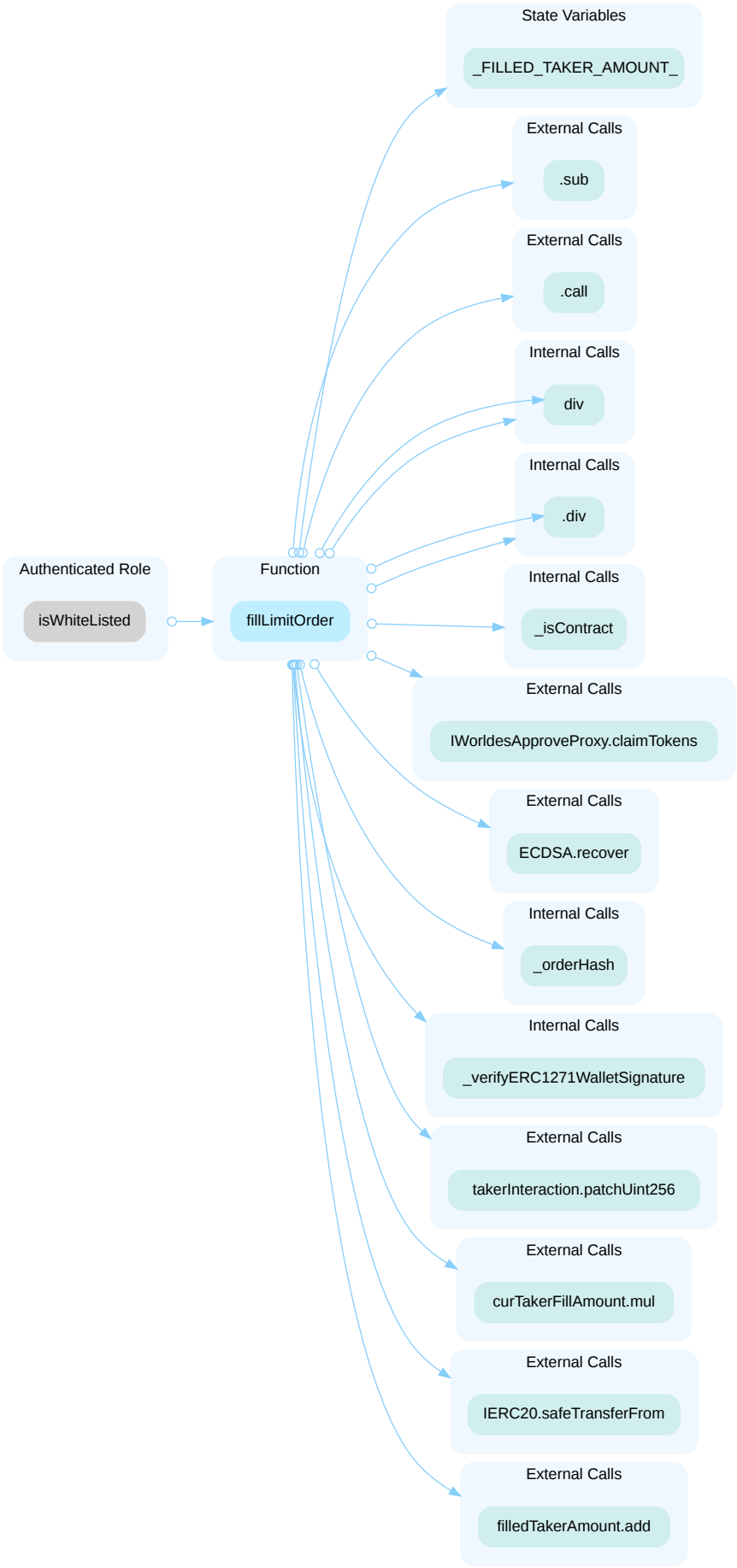
In the contract `WorldesGaslessTrading` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove `_IS_ADMIN_`, set `_INSURANCE_`.



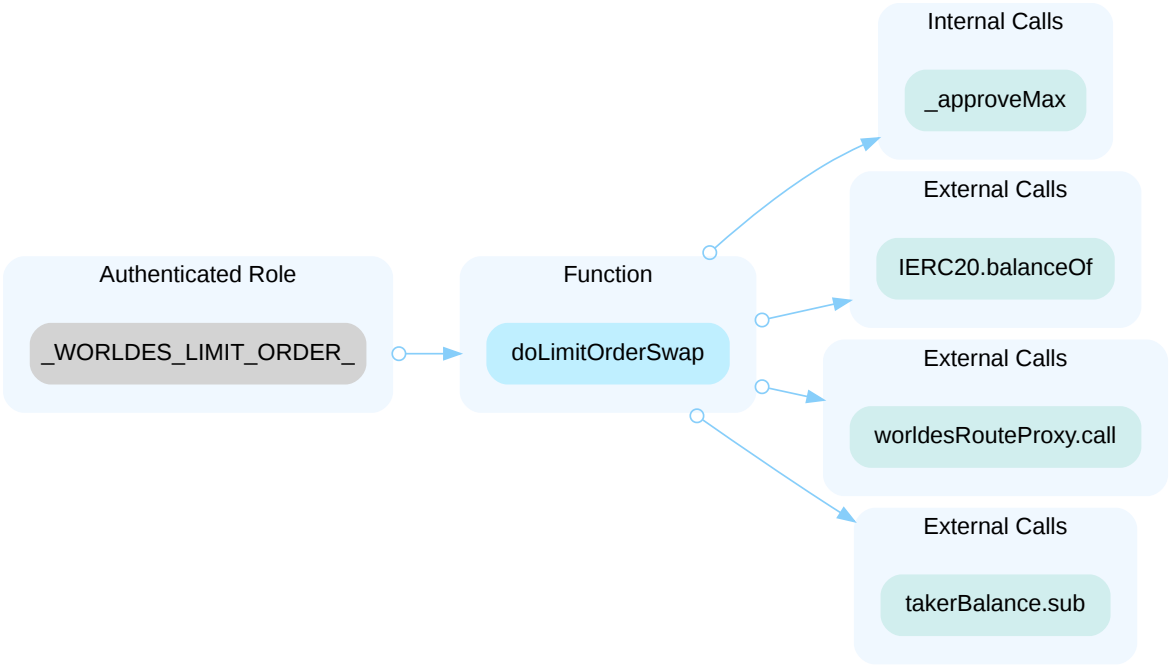
In the contract `WorldesLimitOrder` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove `isWhiteListed`, change `_FEE_RECEIVER_`.



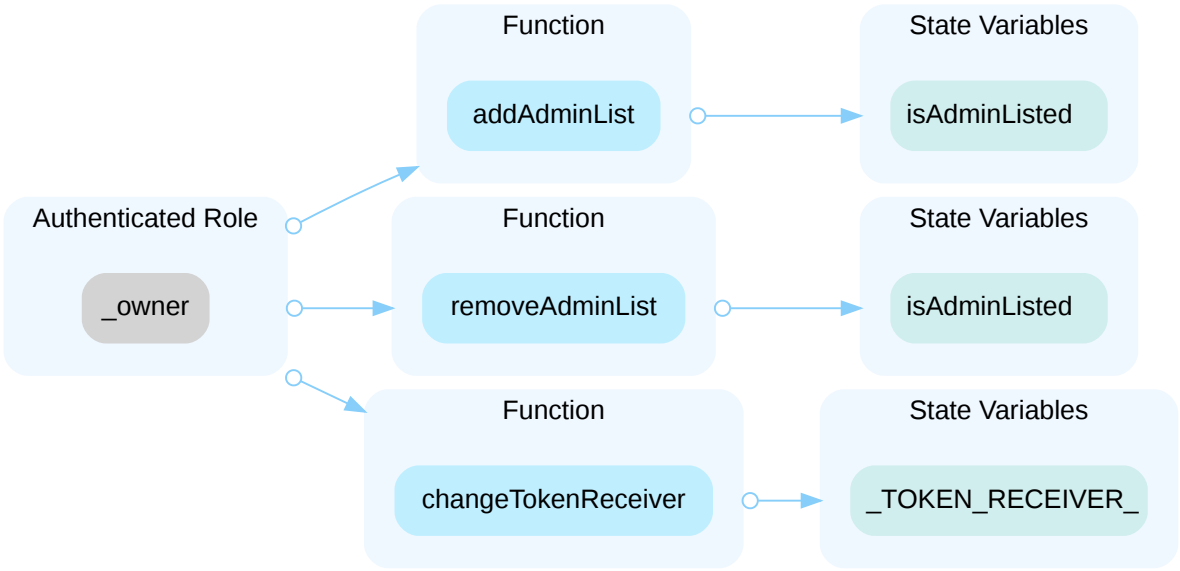
In the contract `WorldesLimitOrder` the role `isWhiteListed` has authority over the functions shown in the diagram below. Any compromise to the `isWhiteListed` account may allow the hacker to take advantage of this authority and call the function `fillLimitOrder()`.


State Variables
_FILLED_TAKER_AMOUNT_
External Calls
.sub
External Calls
.call
Internal Calls
div
Internal Calls
.div
Authenticated Role
isWhiteListed
Function
fillLimitOrder
Internal Calls
_isContract
External Calls
IWorldesApproveProxy.claimTokens
External Calls
ECDSA.recover
Internal Calls
_orderHash
Internal Calls
_verifyERC1271WalletSignature
External Calls
takerInteraction.patchUint256
External Calls
curTakerFillAmount.mul
External Calls
IERC20.safeTransferFrom
External Calls
filledTakerAmount.add

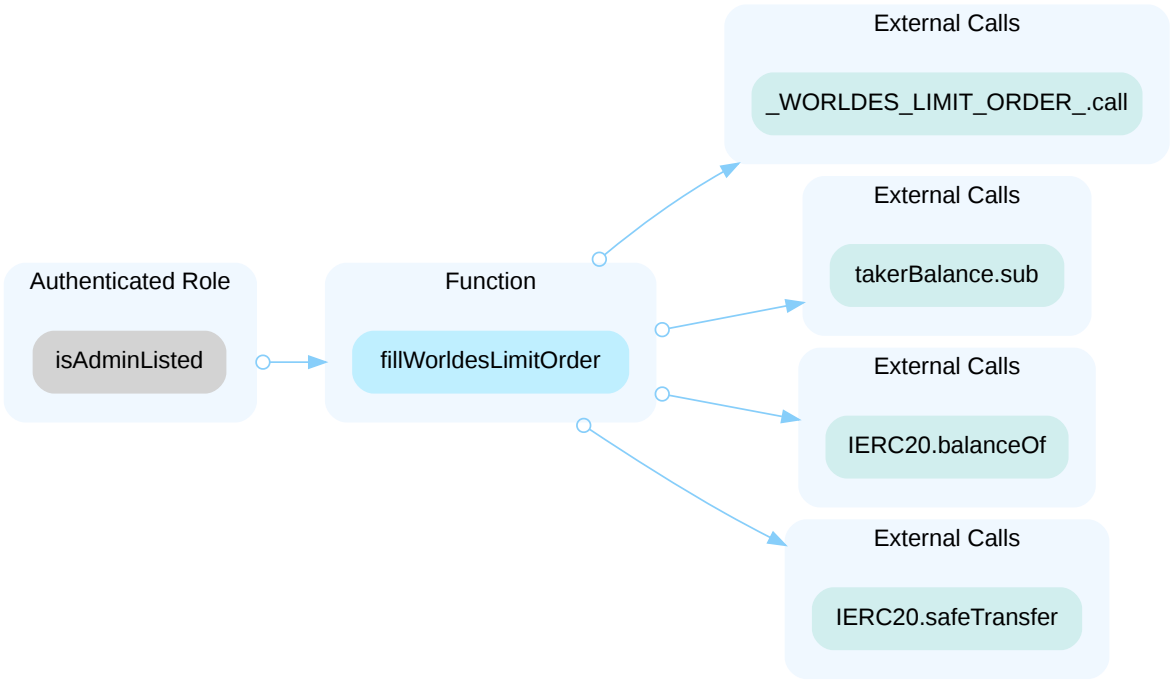
In the contract `WorldesLimitOrderBot` the role `_WORLDES_LIMIT_ORDER_` has authority over the functions shown in the diagram below. Any compromise to the `_WORLDES_LIMIT_ORDER_` account may allow the hacker to take advantage of this authority and call the function `doLimitOrderSwap()`.



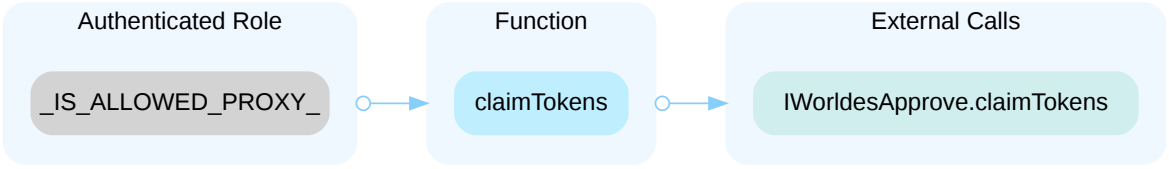
In the contract `WorldesLimitOrderBot` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove `isAdminListed`, change `_TOKEN_RECEIVER_`.



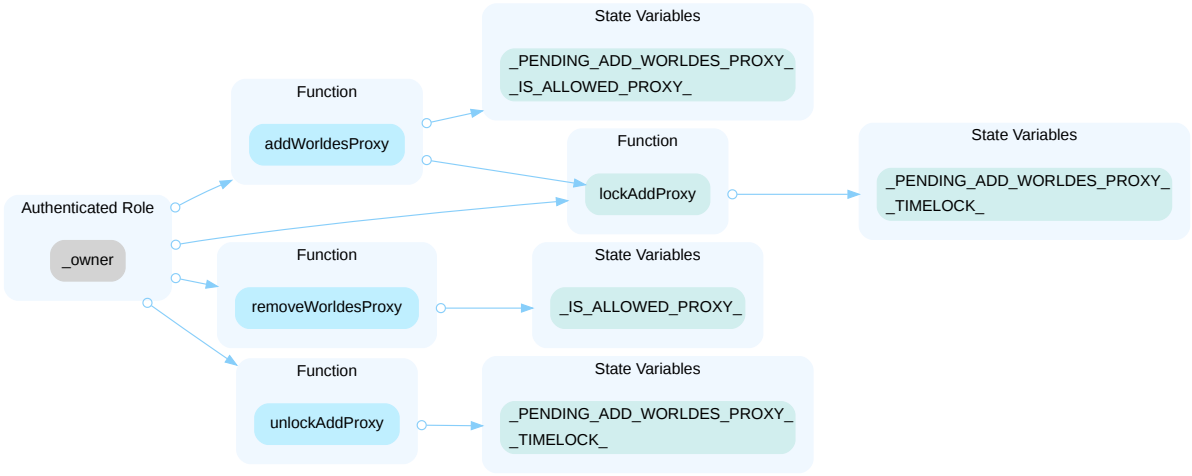
In the contract `WorldesLimitOrderBot` the role `isAdminListed` has authority over the functions shown in the diagram below. Any compromise to the `isAdminListed` account may allow the hacker to take advantage of this authority and call the function `fillWorldesLimitOrder()`.



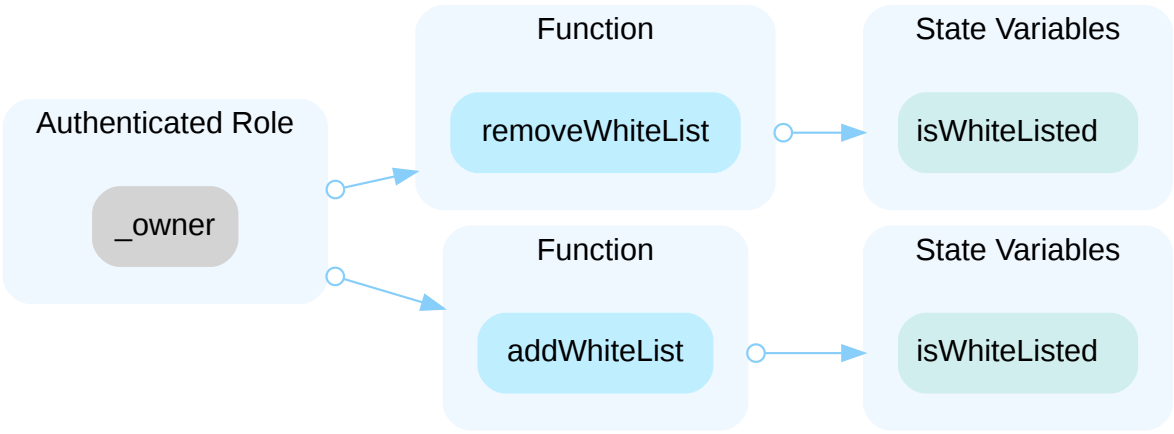
In the contract `WorldesApproveProxy` the role `_IS_ALLOWED_PROXY_` has authority over the function shown in the diagram below. Any compromise to the `_IS_ALLOWED_PROXY_` account may allow the hacker to take advantage of this authority and call the function `claimTokens()`.



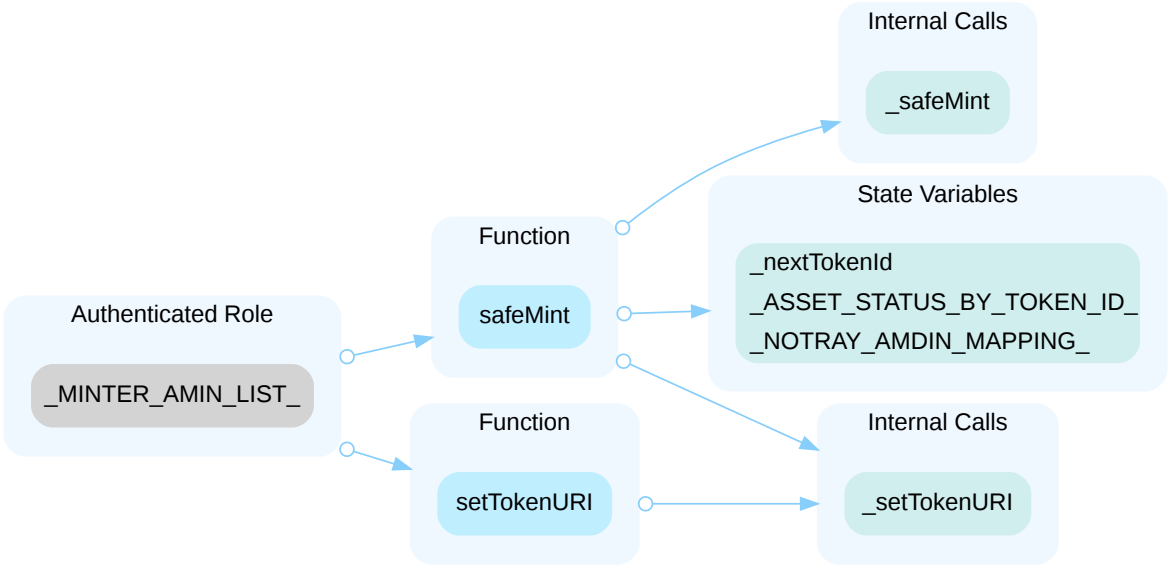
In the contract `WorldesApproveProxy` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set `_TIMELOCK_`, `_PENDING_ADD_WORLDES_PROXY_`, `_IS_ALLOWED_PROXY_`.



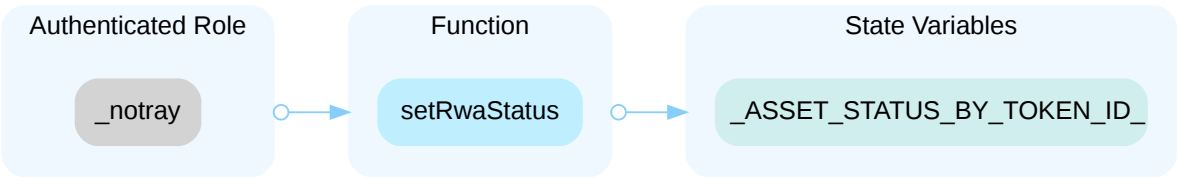
In the contract `WorldesDvmProxy` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove `isWhiteListed`.



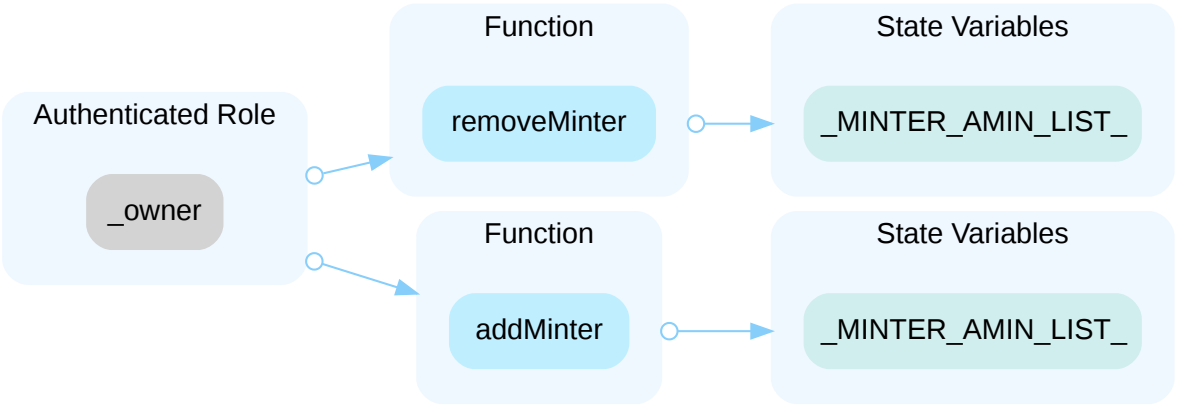
In the contract `WorldesPropertyRights` the role `_MINTER_AMIN_LIST_` has authority over the functions shown in the diagram below. Any compromise to the `_MINTER_AMIN_LIST_` account may allow the hacker to take advantage of this authority and mint a token to any account and set any active token's uri.



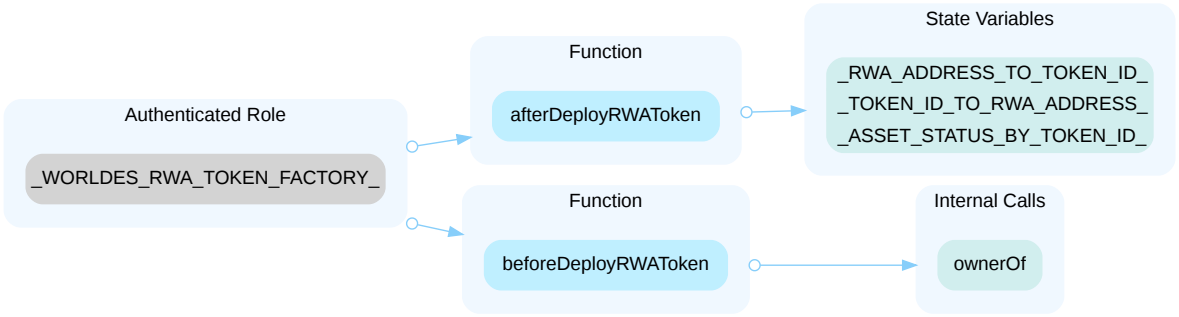
In the contract `WorldesPropertyRights` the token's `_notray` address has authority over the function shown in the diagram below. Any compromise to the `_notray` account may allow the hacker to take advantage of this authority and modify a token's asset status.



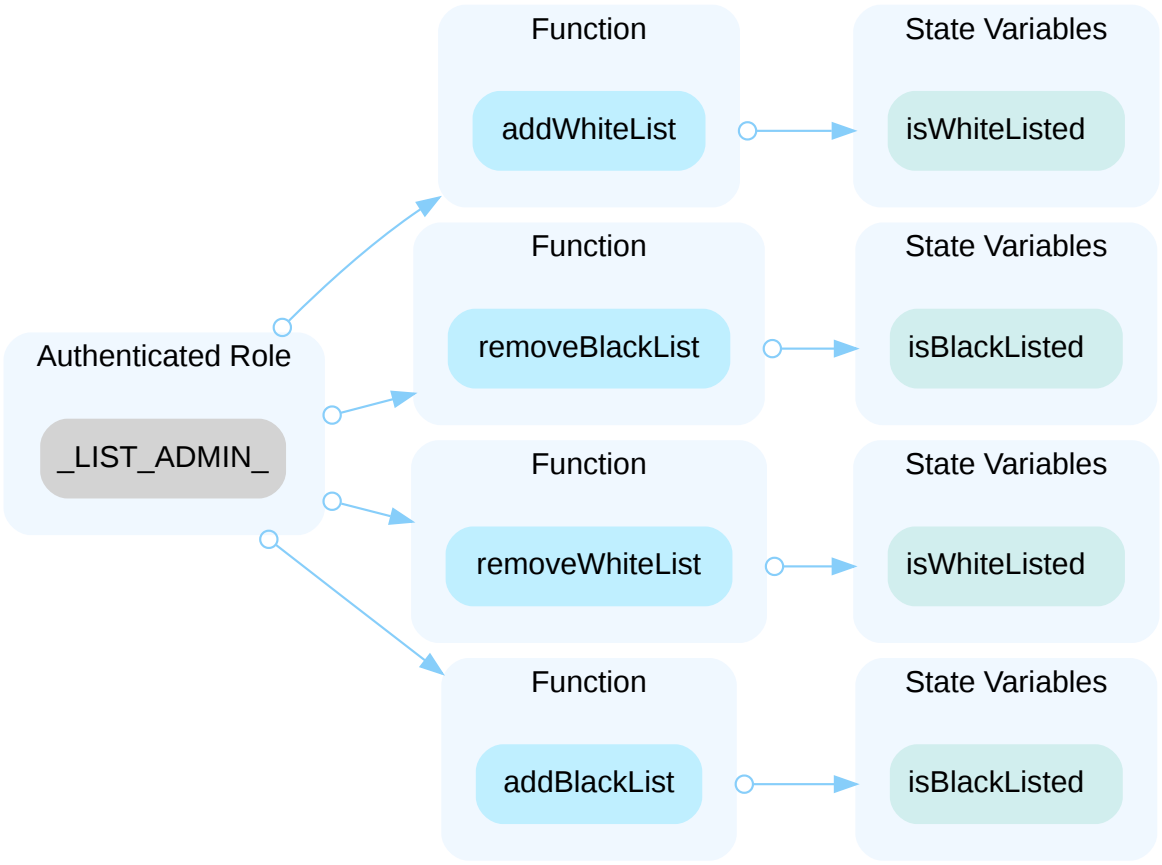
In the contract `WorldesPropertyRights` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add an account to or remove an account from the `_MINTER_AMIN_LIST_`.



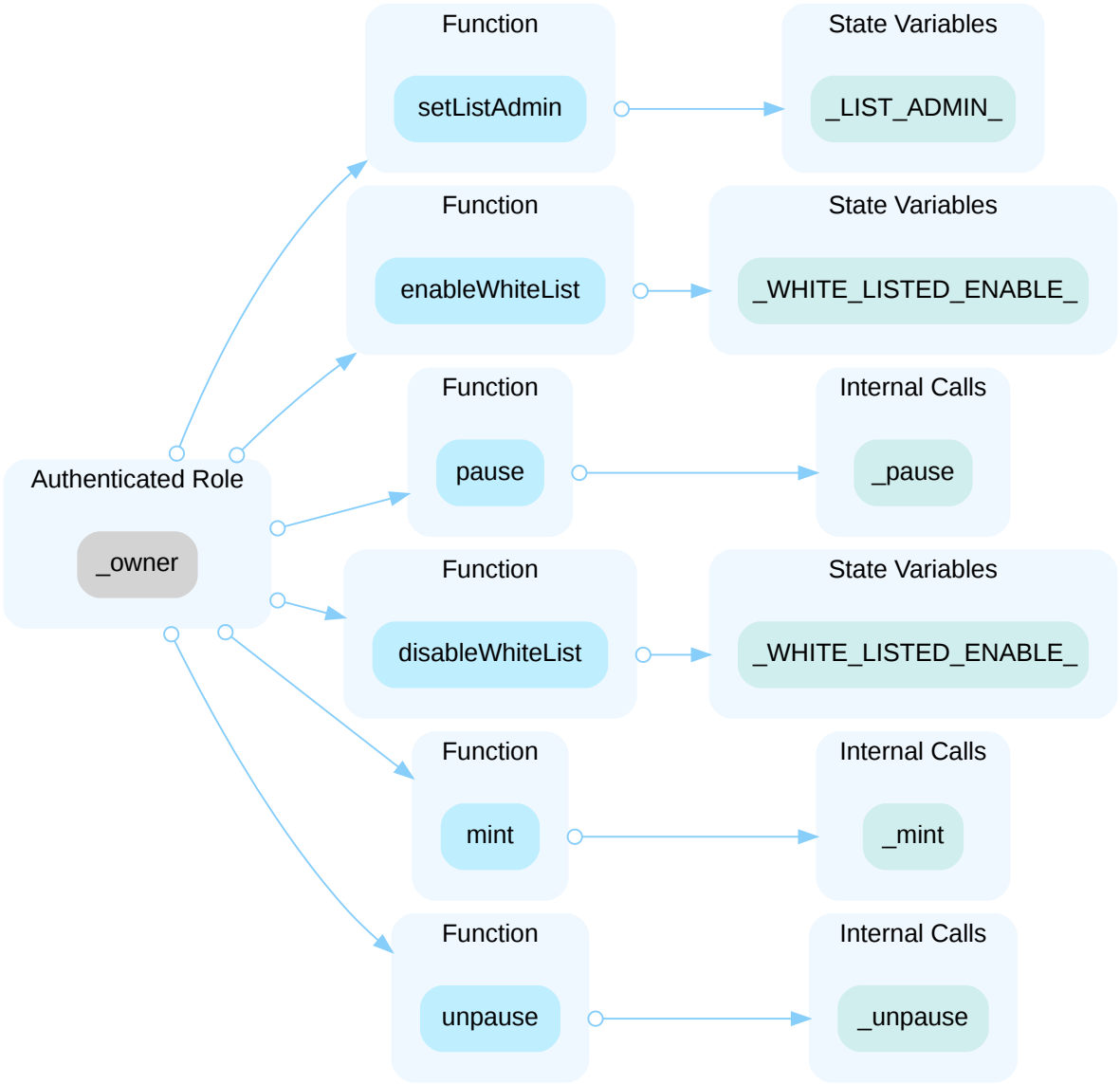
In the contract `WorldesPropertyRights` the role `_WORLDES_RWA_TOKEN_FACTORY_` has authority over the functions shown in the diagram below. Any compromise to the `_WORLDES_RWA_TOKEN_FACTORY_` account may allow the hacker to take advantage of this authority and verify if a `WorldesRWAToken` can be deployed and modify related settings after the deployement.



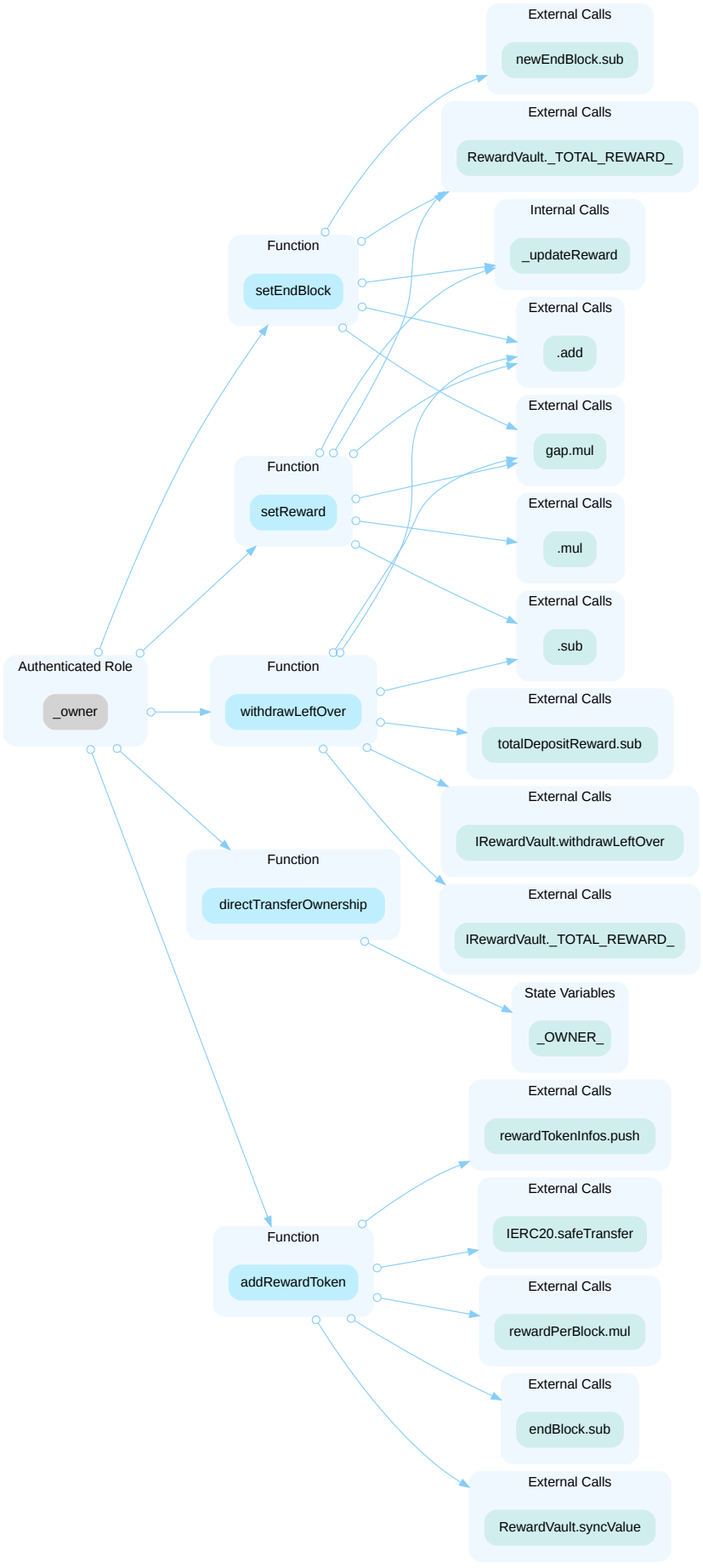
In the contract `WorldesRWAToken` the role `_LIST_ADMIN_` has authority over the functions shown in the diagram below. Any compromise to the `_LIST_ADMIN_` account may allow the hacker to take advantage of this authority and add/remove an account from the whitelist and add/remove an account from the blacklist.



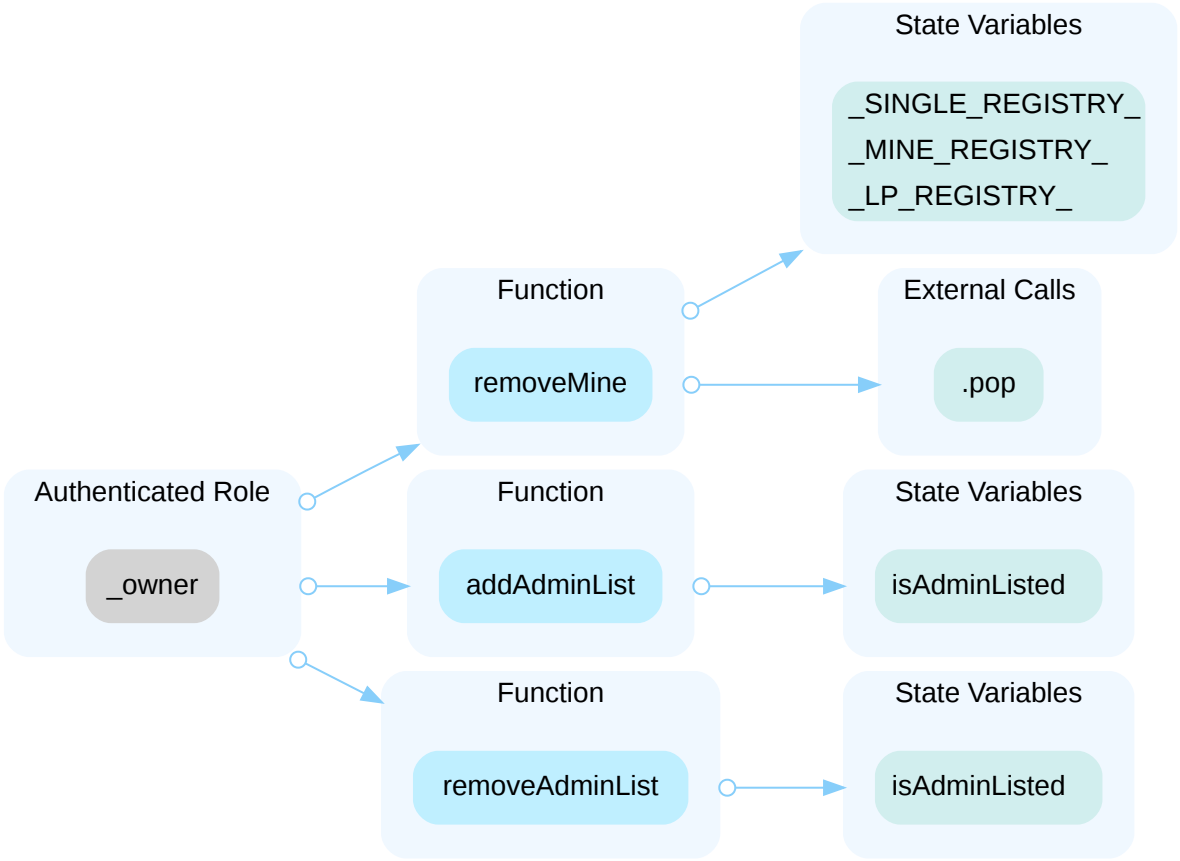
In the contract `WorldesRWAToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the `_LIST_ADMIN_` role, toggle whitelist/blacklist model, pause/unpause token transfers and mint tokens.



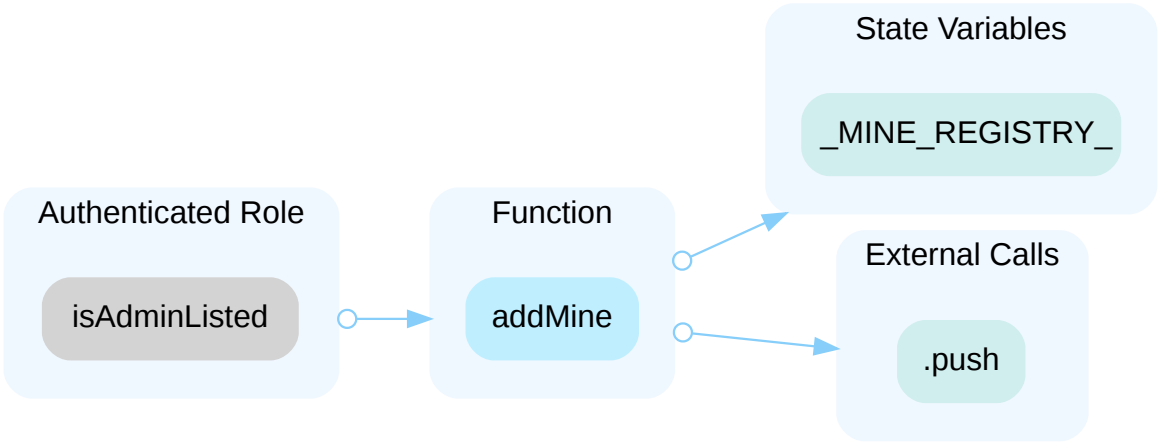
In the contract `BaseMine` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add a reward token, set the reward end block, set the reward per block, withdraw reward tokens, and transfer the ownership.


External Calls
newEndBlock.sub
External Calls
RewardVault._TOTAL_REWARD_
Internal Calls
_updateReward
Function
setEndBlock
External Calls
.add
External Calls
gap.mul
Function
setReward
External Calls
.mul
External Calls
.sub
Authenticated Role
_owner
Function
withdrawLeftOver
External Calls
totalDepositReward.sub
External Calls
IRewardVault.withdrawLeftOver
Function
directTransferOwnership
External Calls
IRewardVault._TOTAL_REWARD_
State Variables
OWNER
External Calls
rewardTokenInfos.push
External Calls
IERC20.safeTransfer
Function
addRewardToken
External Calls
rewardPerBlock.mul
External Calls
endBlock.sub
External Calls
RewardVault.syncValue

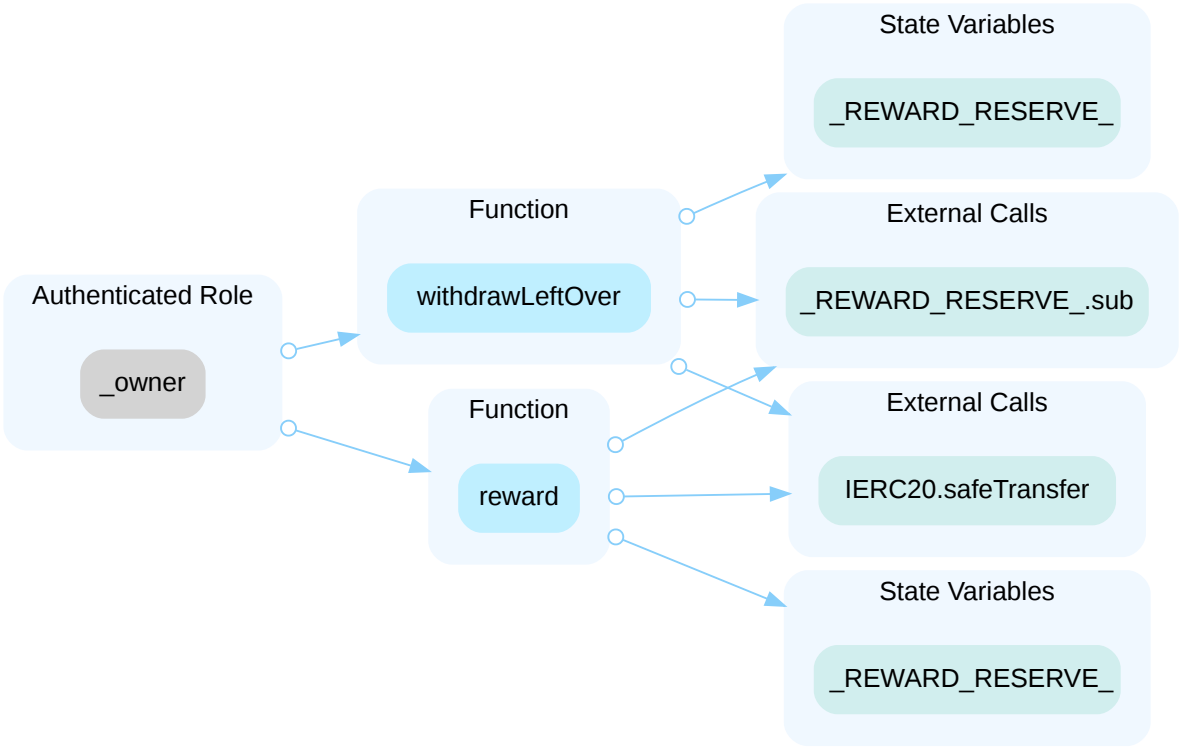
In the contract `WorldesMineRegistry` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and remove a mine from `_MINE_REGISTRY_`, and grant/revoke the `isAdminListed` role from an account.



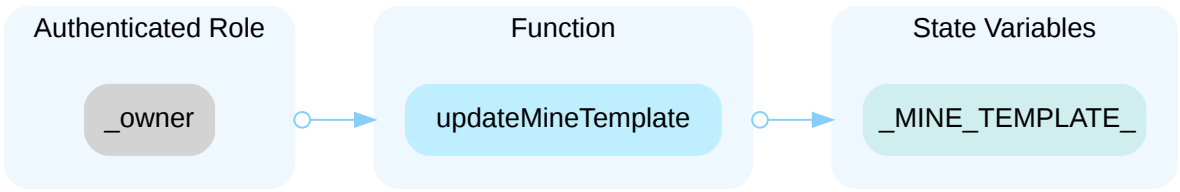
In the contract `WorldesMineRegistry` the role `isAdminListed` has authority over the function shown in the diagram below. Any compromise to the `isAdminListed` account may allow the hacker to take advantage of this authority and add a mine to `_MINE_REGISTRY_`.



In the contract `RewardVault` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and transfer tokens out from the contract.



In the contract `WorldesMineProxy` the role `_owner` has authority over the function shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and update the template contract of the mines.



Modifications in Commit d4199a7d833053f78de6ae87d8aa416d9393ff58

- In the contract `WorldesRWAToken`, a cap `_MAX_SUPPLY_` was added to the total supply, which can be increased by the `_WPR_ADDRESS_` address via the newly added `setMaxSupply()` function.
- In the contract `WorldesPropertyRights`, the `_notray` was renamed to `notary`, and the following two functions were added and can only be called by the token's `_notary` address:
  - `clearRwaRelation()`: Removes the relationship between `tokenId` and RWA Token address.
  - `increaseMaxSupply()`: Increases the RWA Token's `_MAX_SUPPLY`.

Modifications in https://arbiscan.io/address/0xeb8172a596fda7b8c9992dc97ba21db8dfc9ce2d#code

- In the contract `WorldesPropertyRights`, a function `setWorldesRwaTokenFactory` was added to change the `_WORLDES_RWA_TOKEN_FACTORY_` by the owner.

Modifications in https://arbiscan.io/address/0x4Ef31B45919aE1874840B9563D46FCD57E2Ae0b7#code

- In the contract `WorldesRWATokenFactory`, a function `setRwaTokenTemplate` was added to change the `_RWA_TOKEN_TEMPLATE_` by the owner.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.

  OR

- Remove the risky functionality.

Alleviation

[Worldes Team, 03/17/2025]: The team acknowledged the issue and adopted the multisign solution to ensure the private key management process at the current stage.

The WorldesPropertyRights contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x0a7f36ddeb9dceb4061dd088853f18d49971d4a61d6f719d69ca7444809e7e4c

The WorldesApprove contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0xd06e82060230d84b9b05a163a853741ddc11e171571682e46dd635dd077f457d

The DVMFactory contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x0323a39b6350458463ea4b840b14e15addb6d46d34027bbeebd26f8da4517b4f

The DSPFactory contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x9fa80d3cb39ee10f2575754d003d0bf06502f4069dfd4a4a92aeb7baaba61567

The WorldesDvmProxy contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x58edf3693d495afc9adb602e7734c1769d15cc41aff241a2a32657ca4ad1061c

The WorldesRWATokenFactory contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:

https://arbiscan.io/tx/0xd307544840dc7c7633074948db74bd1a729c32d7952ed34da671d5b583a34a97

The WorldesMineProxy contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe: https://arbiscan.io/tx/0xf167fce751ea0fd0d30c25dc6acda1a3a0e9a148a56ac9b7d67f06fbcfb85f79

The WorldesRWAToken contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe: https://arbiscan.io/tx/0x43a27958c9880ff8b82bd0ed4cef95e856b64565f3195f238ac051f1403f3578
- The 3 multisign addresses:
    1. EOA:0x4B1302f82cb0A96DbDeeb52F23f3848EFB2B7f7c
    2. EOA:0xA3627C24FBF5d4e2640948A4cB0e0451B6F3d324
    3. EOA:0xb52c9B17f9696c4A907786F4979021f45323f058

[CertiK, 03/17/2025]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team to periodically revisit the private key security management of all above-listed addresses.

WIO-02 INITIAL DEPOSITOR CAN SET QUOTE TARGET TO ZERO TO AFFECT POOL FUNCTIONALITY

Category	Severity	Location	Status
Logical Issue	Major	libraries/DecimalMath.sol (pre): 23~25; stablePool/implements/DSPFunding.sol (pre): 24~68; stablePool/implements/DSPVault.sol (pre): 199	Resolved

Description

During the initial deposit, there is a potential issue where the quote target can be set to zero, affecting all subsequent liquidity providers (LPs). Even if later LPs deposit significant amounts, the quote target remains zero due to multiplication with the initial zero value. This zero value in the `QUOTE_TARGET` will adversely impact the swaps facilitated by the pool. The quote target is established as follows during the first deposit:

```
if (totalSupply == 0) {
    // case 1. initial supply
    require(quoteBalance > 0, "ZERO_QUOTE_AMOUNT");
    shares = quoteBalance < DecimalMath.mulFloor(baseBalance, _I_)
        ? DecimalMath.divFloor(quoteBalance, _I_)
        : baseBalance;
    _BASE_TARGET_ = uint112(shares);
    _QUOTE_TARGET_ = uint112(DecimalMath.mulFloor(shares, _I_));
    require(shares > 2001, "MINT_AMOUNT_NOT_ENOUGH");
    _mint(address(0), 1001);
    shares -= 1001;
}
```

In this scenario, the 'shares' value can be as low as 1e3, as indicated in the below code snippet.

```
function _mint(address user, uint256 value) internal {
    require(value > 1000, "MINT_AMOUNT_NOT_ENOUGH");
    _SHARES_[user] = _SHARES_[user].add(value);
    totalSupply = totalSupply.add(value);
    emit Mint(user, value);
    emit Transfer(address(0), user, value);
}
```

This means that if an individual deposits very small amounts of quote and base tokens, they can set the QUOTE_TARGET to zero because the `mulFloor()` function uses a scaling factor of 1e18:

```
function mulFloor(uint256 target, uint256 d) internal pure returns (uint256) {
        return target.mul(d) / (10**18);
    }
```

Overall, the initial depositor can manipulate the QUOTE_TARGET to zero, thereby impacting the pool's functionality and subsequent LPs.

Scenario

1. The attacker deposits very small amounts of quote and base tokens, such as 10**5 if the tokens' decimals are 18. The `QUOTE_TARGET` is manipulated to 0.
2. A user deposits regular amounts of tokens, such as 1000 * 10**18. The value of `QUOTE_TARGET` remains 0.

Recommendation

It is recommended to add the following require check before the `_mint()` function call when `totalSupply == 0`:

```
require(_QUOTE_TARGET_ > 0, "QUOTE_TARGET_IS_ZERO");
```

Note that DODOV2 has resolved the issue in commit d055deeace98871c05a0ec30537e631d58c60224.

Alleviation

[Worldes Team, 05/31/2024]: The team heeded the advice and resolved the issue in commit: db5c3736013f44e2b9118f59da55927ed177bf1a.

WWA-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Major	asset/WorldesRWAToken.sol (pre2): 10	Pending

Description

The WorldesRWAToken contract inherits upgradeable contracts, indicating that it is part of an upgradeable system. Upgradeable contracts often pair with a proxy contract that is responsible for managing contract upgrades. The `admin` of the proxy often have the authority to update the implementation contract.

Any compromise of the privileged account could allow a hacker to exploit this authority, potentially altering the implementation contract pointed to by the proxy and thus executing malicious functionality within the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the privileged roles of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.

- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

- Renounce the ownership and never claim back the privileged role;
  OR
- Remove the risky functionality.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Worldes Team, 03/17/2025]: The team acknowledged the issue and adopted the multisign solution to ensure the private key management process at the current stage. The WorldesRWAToken contract has transferred the ownership to a Gnosis Safe contract with 2/3 signers in the sensitive function signing process.

- Grant Role transaction hash for Gnosis Safe:
  https://arbiscan.io/tx/0x43a27958c9880ff8b82bd0ed4cef95e856b64565f3195f238ac051f1403f3578
- The 3 multisign addresses:

1. EOA:0x4B1302f82cb0A96DbDeeb52F23f3848EFB2B7f7c
2. EOA:0xA3627C24FBF5d4e2640948A4cB0e0451B6F3d324
3. EOA:0xb52c9B17f9696c4A907786F4979021f45323f058

[CertiK, 03/17/2025]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team to periodically revisit the private key security management of all above-listed addresses.

BMM-01 BLOCK.TIMESTAMP SHOULD BE USED INSTEAD OF BLOCK.NUMBER ON ARBITRUM

Category	Severity	Location	Status
Logical Issue	Minor	contracts/mining/BaseMine.sol (ERC20Mine): 27, 56	Pending

Description

The reward distribution should be a variable of time instead of a block number, as a block is minted in arbitrum every 0.26 seconds, while a block is minted in Ethereum every 12 seconds.

Recommendation

We advise the team to replace the usage of `block.number` with `block.timestamp`.

WPR-01 | INVALID USE OF ACCESS CONTROL MODIFIER

Category	Severity	Location	Status
Logical Issue	Minor	asset/WorldesPropertyRights.sol (pre): 76~77	Resolved

Description

The functions marked as 'view' or 'pure' are unnecessarily restricted by the 'restrict' modifier. These functions are designed to be read-only, meaning they do not modify the state on the blockchain. However, they are restricted so that only a specific address can call them.

It's important to note that even private state variables can be read off-chain, rendering the access restriction on these functions ineffective.

Additionally, this function only contains two `require` statements, without any return values.

Recommendation

We recommend that access restrictions are not used on `view` or `pure` functions, as they do not improve security for read-only operations. Instead, these getter functions should be made public to allow transparency and follow best practice. If there is sensitive information that should not be disclosed, the way in which this data is managed and stored should be reconsidered, as restricting access in this way does not provide effective security.

Alleviation

[Worldes Team, 05/31/2024]: The team heeded the advice and resolved the issue in commit: f47273d4b9ee2ca22c2081bc0494d470ae82fb43.

WPW-01 LACK OF EXISTENCE CHECK IN `setTokenURI()` FUNCTION

Category	Severity	Location	Status
Logical Issue	Minor	asset/WorldesPropertyRights.sol (pre2): 120	Acknowledged

Description

The `setTokenURI()` function allows setting the URI for a token identified by `tokenId`. However, it does not check if the token with the given `tokenId` has been minted. This omission can lead to the URI being set for a non-existing token, which can cause inconsistencies and potential errors in the contract's behavior.

```
120     function setTokenURI(uint256 tokenId, string memory uri) external
 onlyMinter {
121         require(_ASSET_STATUS_BY_TOKEN_ID_[tokenId] != AssetStatus.Voided,
"WPR: token is voided.");
122         _setTokenURI(tokenId, uri);
123     }
```

Recommendation

Recommend using the `_exists()` function to check if the token exists.

Alleviation

[Worldes Team, 06/11/2024]: The team acknowledged the finding and decided not to change the current codebase.

FORMAL VERIFICATION | WORLDES

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of ERC-20 Compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply`, which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
erc20-approve-revert-zero	`approve` Prevents Approvals For the Zero Address
erc20-transferfrom-revert-zero-argument	`transferFrom` Fails for Transfers with Zero Address Arguments
erc20-transfer-revert-zero	`transfer` Prevents Transfers to the Zero Address
erc20-approve-never-return-false	`approve` Never Returns `false`
erc20-approve-succeed-normal	`approve` Succeeds for Valid Inputs
erc20-approve-false	If `approve` Returns `false`, the Contract's State Is Unchanged
erc20-allowance-succeed-always	`allowance` Always Succeeds
erc20-transferfrom-never-return-false	`transferFrom` Never Returns `false`
erc20-totalsupply-correct-value	`totalSupply` Returns the Value of the Corresponding State Variable
erc20-allowance-correct-value	`allowance` Returns Correct Value
erc20-balanceof-correct-value	`balanceOf` Returns the Correct Value

Property Name	Title
erc20-balanceof-succeed-always	`balanceOf` Always Succeeds
erc20-totalsupply-succeed-always	`totalSupply` Always Succeeds
erc20-approve-correct-amount	`approve` Updates the Approval Mapping Correctly
erc20-transfer-never-return-false	`transfer` Never Returns `false`
erc20-transfer-recipient-overflow	`transfer` Prevents Overflows in the Recipient's Balance
erc20-transferfrom-false	If `transferFrom` Returns `false`, the Contract's State Is Unchanged
erc20-transferfrom-fail-exceed-balance	`transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-fail-exceed-allowance	`transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transfer-false	If `transfer` Returns `false`, the Contract State Is Not Changed
erc20-transferfrom-fail-recipient-overflow	`transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transfer-exceed-balance	`transfer` Fails if Requested Amount Exceeds Available Balance
erc20-totalsupply-change-state	`totalSupply` Does Not Change the Contract's State
erc20-allowance-change-state	`allowance` Does Not Change the Contract's State
erc20-balanceof-change-state	`balanceOf` Does Not Change the Contract's State
erc20-transferfrom-correct-allowance	`transferFrom` Updated the Allowance Correctly
erc20-transfer-correct-amount	`transfer` Transfers the Correct Amount in Transfers
erc20-transferfrom-correct-amount	`transferFrom` Transfers the Correct Amount in Transfers

Verification Results

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.
- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.
- Inapplicable: The property does not apply to the project.

Detailed Results For Contract DVMTrader (contracts/vendingMachine/implements/DVMTrader.sol) In Commit d4199a7d833053f78de6ae87d8aa416d9393ff58

Verification of ERC-20 Compliance

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	Inconclusive	
erc20-approve-never-return-false	True	
erc20-approve-succeed-normal	True	
erc20-approve-false	True	
erc20-approve-correct-amount	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	Inconclusive	
erc20-transfer-never-return-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-change-state	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-correct-value	True	
erc20-balanceof-succeed-always	True	
erc20-balanceof-change-state	True	

Detailed Results For Contract DVMVault (contracts/vendingMachine/implements/DVMVault.sol) In Commit d4199a7d833053f78de6ae87d8aa416d9393ff58

Verification of ERC-20 Compliance

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-exceed-balance	True	
erc20-transfer-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	
erc20-transfer-never-return-false	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-fail-exceed-balance	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-change-state	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-succeed-always	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-change-state	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-succeed-always	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	Inconclusive	
erc20-approve-never-return-false	True	
erc20-approve-false	True	
erc20-approve-succeed-normal	True	
erc20-approve-correct-amount	True	

Detailed Results For Contract DSPVault (contracts/stablePool/implements/DSPVault.sol) In Commit d4199a7d833053f78de6ae87d8aa416d9393ff58

Verification of ERC-20 Compliance

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-false	True	
erc20-approve-never-return-false	True	
erc20-approve-correct-amount	True	
erc20-approve-revert-zero	Inconclusive	
erc20-approve-succeed-normal	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-change-state	True	
erc20-totalsupply-correct-value	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-false	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	Inconclusive	
erc20-transferfrom-revert-zero-argument	Inconclusive	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-false	True	
erc20-transfer-never-return-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	

Detailed Results For Contract DSPTrader (contracts/stablePool/implements/DSPTrader.sol) In Commit d4199a7d833053f78de6ae87d8aa416d9393ff58

Verification of ERC-20 Compliance

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	
erc20-balanceof-succeed-always	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-never-return-false	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	True	
erc20-transfer-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	Inconclusive	
erc20-approve-succeed-normal	True	
erc20-approve-never-return-false	True	
erc20-approve-false	True	
erc20-approve-correct-amount	True	

Detailed Results For Contract DVMTrader (contracts/vendingMachine/implements/DVMTrader.sol) In Commit ca7d926e9387f304ccea4ab72860e882242109af

Verification of ERC-20 Compliance

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-correct-amount	True	
erc20-approve-revert-zero	Inconclusive	
erc20-approve-never-return-false	True	
erc20-approve-false	True	
erc20-approve-succeed-normal	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-never-return-false	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-never-return-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-change-state	True	
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	

Detailed Results For Contract DSPVault (contracts/stablePool/implements/DSPVault.sol) In Commit ca7d926e9387f304ccea4ab72860e882242109af

Verification of ERC-20 Compliance

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	
erc20-transfer-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-never-return-false	True	
erc20-transfer-recipient-overflow	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-fail-exceed-allowance	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	Inconclusive	
erc20-approve-false	True	
erc20-approve-never-return-false	True	
erc20-approve-succeed-normal	True	
erc20-approve-correct-amount	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-correct-value	True	
erc20-balanceof-succeed-always	True	
erc20-balanceof-change-state	True	

Detailed Results For Contract DVMVault (contracts/vendingMachine/implements/DVMVault.sol) In Commit ca7d926e9387f304ccea4ab72860e882242109af

Verification of ERC-20 Compliance

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-never-return-false	True	
erc20-approve-false	True	
erc20-approve-succeed-normal	True	
erc20-approve-correct-amount	True	
erc20-approve-revert-zero	Inconclusive	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-correct-value	True	
erc20-allowance-succeed-always	True	
erc20-allowance-change-state	True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-false	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	

Detailed Results For Contract DSPTrader (contracts/stablePool/implements/DSPTrader.sol) In Commit ca7d926e9387f304ccea4ab72860e882242109af

Verification of ERC-20 Compliance

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-false	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-revert-zero-argument	Inconclusive	
erc20-transferfrom-never-return-false	True	

Detailed Results for Function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	
erc20-totalsupply-succeed-always	True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	True	
erc20-transfer-false	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	Inconclusive	

Detailed Results for Function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-change-state	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-succeed-always	True	

Detailed Results for Function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-change-state	True	
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	Inconclusive	
erc20-approve-succeed-normal	True	
erc20-approve-never-return-false	True	
erc20-approve-correct-amount	True	
erc20-approve-false	True	

APPENDIX | WORLDES

Finding Categories

Categories	Description
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed ERC-20 Properties

Properties related to function `approve`

erc20-approve-correct-amount

All non-reverting calls of the form `approve(spender, amount)` that return `true` must correctly update the allowance mapping according to the address `msg.sender` and the values of `spender` and `amount`.

Specification:

```
requires spender != address(0);
ensures \result ==> allowance(msg.sender, \old(spender)) == \old(amount);
```

erc20-approve-false

If function `approve` returns `false` to signal a failure, it must undo all state changes that it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-approve-never-return-false

The function `approve` must never returns `false`.

Specification:

```
ensures \result;
```

erc20-approve-revert-zero

All calls of the form `approve(spender, amount)` must fail if the address in `spender` is the zero address.

Specification:

```
ensures \old(spender) == address(0) ==> !\result;
```

erc20-approve-succeed-normal

All calls of the form `approve(spender, amount)` must succeed, if

- the address in `spender` is not the zero address and
- the execution does not run out of gas.

Specification:

```
requires spender != address(0);
ensures \result;
reverts_only_when false;
```

Properties related to function `transferFrom`

erc20-transferfrom-correct-allowance

All non-reverting invocations of `transferFrom(from, dest, amount)` that return `true` must decrease the allowance for address `msg.sender` over address `from` by the value in `amount`.

Specification:

```
ensures \result ==> allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) - \old(amount)
                  || (allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) && \old(allowance(sender, msg.sender)) == type(uint256).max);
```

erc20-transferfrom-correct-amount

All invocations of `transferFrom(from, dest, amount)` that succeed and that return `true` subtract the value in `amount` from the balance of address `from` and add the same value to the balance of address `dest`.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient) +
amount)
                  && balanceOf(\old(sender)) == \old(balanceOf(sender) - amount);
  also
requires recipient == sender;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient));
```

erc20-transferfrom-fail-exceed-allowance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the allowance of address `msg.sender` must fail.

Specification:

```
requires msg.sender != sender;
requires amount > allowance(sender, msg.sender);
ensures !\result;
```

erc20-transferfrom-fail-exceed-balance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the balance of address `from` must fail.

Specification:

```
requires amount > balanceOf(sender);
ensures !\result;
```

erc20-transferfrom-fail-recipient-overflow

Any call of `transferFrom(from, dest, amount)` with a value in `amount` whose transfer would cause an overflow of the balance of address `dest` must fail.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transferfrom-false

If `transferFrom` returns `false` to signal a failure, it must undo all incurred state changes before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transferfrom-never-return-false

The `transferFrom` function must never return `false` .

Specification:

```
ensures \result;
```

erc20-transferfrom-revert-zero-argument

All calls of the form `transferFrom(from, dest, amount)` must fail for transfers from or to the zero address.

Specification:

```
ensures \old(sender) == address(0) ==> !\result;
also
ensures \old(recipient) == address(0) ==> !\result;
```

Properties related to function `transfer`

erc20-transfer-correct-amount

All non-reverting invocations of `transfer(recipient, amount)` that return `true` must subtract the value in `amount` from the balance of `msg.sender` and add the same value to the balance of the `recipient` address.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(recipient) == \old(balanceOf(recipient) + amount)
&& balanceOf(msg.sender) == \old(balanceOf(msg.sender) - amount);
  also
requires recipient == msg.sender;
ensures \result ==> balanceOf(msg.sender) == \old(balanceOf(msg.sender));
```

erc20-transfer-exceed-balance

Any transfer of an amount of tokens that exceeds the balance of `msg.sender` must fail.

Specification:

```
requires amount > balanceOf(msg.sender);
ensures !\result;
```

erc20-transfer-false

If the `transfer` function in contract `DVMTrader` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `DVMVault` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `DSPVault` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `DSPTrader` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-never-return-false

The transfer function must never return `false` to signal a failure.

Specification:

```
ensures \result;
```

erc20-transfer-recipient-overflow

Any invocation of `transfer(recipient, amount)` must fail if it causes the balance of the `recipient` address to overflow.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transfer-revert-zero

Any call of the form `transfer(recipient, amount)` must fail if the recipient address is the zero address.

Specification:

```
ensures \old(recipient) == address(0) ==> !\result;
```

Properties related to function `allowance`

erc20-allowance-change-state

Function `allowance` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-allowance-correct-value

Invocations of `allowance(owner, spender)` must return the allowance that address `spender` has over tokens held by address `owner`.

Specification:

```
ensures \result == allowance(\old(owner), \old(spender));
```

erc20-allowance-succeed-always

Function `allowance` must always succeed, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `totalSupply`

erc20-totalsupply-change-state

The `totalSupply` function in contract DVMTrader must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract DVMVault must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract DSPVault must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract DSPTrader must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract DVMTrader.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract DVMVault.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract DSPTrader.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract DSPVault.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-succeed-always

The function `totalSupply` must always succeeds, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `balanceOf`

erc20-balanceof-change-state

Function `balanceOf` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-balanceof-correct-value

Invocations of `balanceOf(owner)` must return the value that is held in the contract's balance mapping for address `owner`.

Specification:

```
ensures \result == balanceOf(\old(account));
```

erc20-balanceof-succeed-always

Function `balanceOf` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.