SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Dexalot
Date: January 19th, 2023

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Dexalot
Approved By	Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Type	ERC20 token; on-chain exchange
Platform	Avalanche
Language	Solidity
Methodology	Link
Website	https://dexalot.com
Changelog	23.11.2022 - Initial Review 27.12.2022 - Second Review 30.12.2022 - Third Review 10.01.2023 - Fourth Review 19.01.2023 - Fifth Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Dexalot (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project consists of a review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/Dexalot/contracts
Commit	c9c806c909e0d4d2cf723914206c889e59ec2caf
Whitepaper	Link
Functional Requirements	-
Technical Requirements	Link
Contracts Addresses	-
Contracts	File: ./contracts/bridgeApps/LzApp.sol SHA3: 2fca0e10f2d8ebf34a551a83808884d87985d398d745e61cf0c7a1dbee66c315 File: ./contracts/DexalotSubnetBalances.sol SHA3: a839e1f3ceda7e4060bd58c66c03c10542e9f7fef656d954207e2b7df9c1f9a2 File: ./contracts/Exchange.sol SHA3: 550ebf91e6f22fb2313f23067f88376766454a6dc3af6d70e31d0b1c88db1f9c File: ./contracts/ExchangeMain.sol SHA3: 178e2a31337b9360ece798c1ed5c11fd3663c8e0e1f0aa6e16f5aeb5dc0af5fa File: ./contracts/ExchangeSub.sol SHA3: 3a2ee57626eda5025a86d9f4c0995c8af38522202486832425b2511fbd8e61fc File: ./contracts/GasStation.sol SHA3: 5c5e302ed3d68cbd8bc6d3784e29757328d391995945bd7311f20b2637ec757c File: ./contracts/interfaces/IGasStation.sol SHA3: ee7e2a9630531863f69655fcc81423b2d7fd77d6afb59d4ef1d692e26742c89f File: ./contracts/interfaces/INativeMinter.sol SHA3: c908468fcfb75537ab31e2e9228de06858521fd4788d6fd63934743acec3f028

	File: ./contracts/interfaces/IPortfolio.sol SHA3: f20f67f62f6e31ff36c9f58301083bc97da89db1036cd0e79940724e662daf02 File: ./contracts/interfaces/IPortfolioBridge.sol SHA3: 9eb762559817b418c2b34c052653ac94b297dfcef06d4c03a216be29d6ec9b6d File: ./contracts/interfaces/IPortfolioMinter.sol SHA3: cf98e4c727d3b316a11d403fa03d5fb47805a4f1cc796722a694cfadadc9f341 File: ./contracts/interfaces/ITradePairs.sol SHA3: e6b68dc6db499079377de1b049f38a8364dbc30eeabe2459fb4e503b0bf72d6a File: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol SHA3: 371010d107f33cb95c89bdcb146a4926460a5d0e5e293ed8eb686dc382661f46 File: ./contracts/interfaces/layerZero/ILayerZeroReceiver.sol SHA3: 14ac55f1f84ae31204768bb17398725ab29f24a7e507630e7da65db5b43dc584 File: ./contracts/interfaces/layerZero/ILayerZeroUserApplicationConfig.sol SHA3: 7a75ffa592bcb4902a573d5b3ddf8faef41650d68f7d92990eacf942a70f92a4 File: ./contracts/library/Bytes32LinkedListLibrary.sol SHA3: 2e4119c6a6f159755c7323324b5e9b2a64d00316267fca1d14f3748a1af881ca File: ./contracts/library/RBTLibrary.sol SHA3: 5ed8a152c203b10a0892e32d7c32263035c03b366da444775c3a80c11eaf70bf File: ./contracts/library/UtilsLibrary.sol SHA3: c9d01f25d73b36ee1ba28a173ad21073f3c584d2b234b4502de56b7fc9c78dcc File: ./contracts/OrderBooks.sol SHA3: bc7fba1377bc6816f522de7b5ccc9ddb120ffc5064cdf5302f6dbf4ff582b24c File: ./contracts/Portfolio.sol SHA3: 55b546b5687a104ef99acdeeb7ff48dc1928c92ac9fa0ee178fc3756feb8db8d File: ./contracts/PortfolioBridge.sol SHA3: e071e1cce38ca62a4a9a9fc2b2e42e86cda269138cc0d724fed8af7a1fa984ba File: ./contracts/PortfolioBridgeSub.sol SHA3: 59b3218c2c36e6b139fbcccc17f7bba53ddc0133d174027983c2af3cd0828707 File: ./contracts/PortfolioMain.sol

	SHA3: 2ac70f5132b90608cd4923c5482047ac06c1d312c89a955b6cb563b31c1b8e59 File: ./contracts/PortfolioMinter.sol SHA3: 002ea8ea4a596a619d23fb73588dd801f874c7881792adfd845ba039651cba35 File: ./contracts/PortfolioSub.sol SHA3: 13d7ba767c63f9863160bb86c8a320b5aeaafb319c5147b789bcfd47a3fe2232 File: ./contracts/token/IncentiveDistributor.sol SHA3: ba414cb696e9ae9aaa3e3a5f138855f8747f811b8202315c94021c94a07f7932 File: ./contracts/TradePairs.sol SHA3: 64caa52f99b2b5cbce16e346045b0e8c7f5c9b09be650cce2c105dd3add00c13

Second review scope

Repository	https://github.com/Dexalot/contracts
Commit	807f523
Whitepaper	Link
Functional Requirements	-
Technical Requirements	Link
Contracts Addresses	-
Contracts	File: ./contracts/bridgeApps/LzApp.sol SHA3: 7c48007352827562226669dcd28af63c00c1e4b2dcb870b3d7d16b7f22424ffb7 File: ./contracts/DexalotSubnetBalances.sol SHA3: 6852c53bdeeb8ca8b235cb2191c02a98ec3f5360b84b0e539668b6e24ae8f912 File: ./contracts/Exchange.sol SHA3: 49c02ce260ac3447986004444399ee1745a391d340bbff7217a2a12e14e12e03 File: ./contracts/ExchangeMain.sol SHA3: 8f8eefe1e432e188c13737c616f24fd786b96b83e9ec89a6f422eb9a4dfae675 File: ./contracts/ExchangeSub.sol SHA3: ab9a1cfb3c34c44583fbb1c69b353a1d172be8238f0515b40660b43ac75c43c8 File: ./contracts/GasStation.sol SHA3: 5d2de02da70e116496c57467f4a642c0e4efddc134b05602734903a1b3f1bf42 File: ./contracts/interfaces/IGasStation.sol

	SHA3: dfd84fed1e22f38b1187be7c2969fa73e1f650051dc551265c1d910b005a4ecd File: ./contracts/interfaces/INativeMinter.sol SHA3: 79d381f48d7acc73b37bd5348ce7df0531b9f4832a408a5084362fc94b6d4768 File: ./contracts/interfaces/IPortfolio.sol SHA3: 3b41a3f31ab77a082e5155c68719fe0de56cb827fa247986a5e531058a17fb93 File: ./contracts/interfaces/IPortfolioBridge.sol SHA3: be503016e6e2fd41bcdd0cdc5728260108e5fce32377a806a28dff309c7ec061 File: ./contracts/interfaces/IPortfolioBridgeSub.sol SHA3: 3da0244eb221fa10ba30de79600ef705614def416a28d6884278cd771f507afc File: ./contracts/interfaces/IPortfolioMain.sol SHA3: 5152c35b35b029678c2ec1e3e5b75bfe0f2232a740a0ce2f89501e5f716f5713 File: ./contracts/interfaces/IPortfolioMinter.sol SHA3: a09113c195a341f4b8d4b5d0ebfa3c064a7ba68c35930adb52c650297682ec80 File: ./contracts/interfaces/IPortfolioSub.sol SHA3: b4c8d547c989c61ed7b5ad66f52848b82ed1ceea678af59787b86a80b7929ab0 File: ./contracts/interfaces/ITradePairs.sol SHA3: 326f0067e764d8750fc0c6d98bf52a2ba1298aa242c41b48463e93d7e14b3149 File: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol SHA3: 77bc094bd019313e1aaa48bcd3699d2db4af174e496f1deaf9c99f0d511e570b File: ./contracts/interfaces/layerZero/ILayerZeroReceiver.sol SHA3: 62c575ff041db59d1e1c8eea84ed441cd86dddc9546c4284affed1da9710b5bf File: ./contracts/interfaces/layerZero/ILayerZeroUserApplicationConfig.sol SHA3: 6864f31c800af55316e156a34ecfef7612b75cf5b1d218c6dc11afd81e667c05 File: ./contracts/library/Bytes32LinkedListLibrary.sol SHA3: 8a0c8c402f983a350c6b3f00c3a3c67e558caafda0823d4e7946e1d708fcff89 File: ./contracts/library/RBTLibrary.sol SHA3: 1d75d91f8b4b5ef92a9d69cbbd1706b03fd0e46b1d94f01cbf8eaf1a3249f546 File: ./contracts/library/UtilsLibrary.sol

	SHA3: eece78d6abfa1a7f84841f4fd7cbb028f3635c933ae166842619c6d51f601591 File: ./contracts/OrderBooks.sol SHA3: 90fc3c23394cc9623c407f315015c79bb16c861ca955f93a43e378b48c31b465 File: ./contracts/Portfolio.sol SHA3: 47af12ba5419c7ef10546b3bf1d19aeb221744e573dfbcd3091a4ddff18d2841 File: ./contracts/PortfolioBridge.sol SHA3: fab4148347c428aa74beffcb86a215a524957b7b2345c546d3f30cc07d915f0f File: ./contracts/PortfolioBridgeSub.sol SHA3: cf04dbdbf2e98c5e3064c00569ffe0127e1d853f525e807b1bf16cebbc2c972d File: ./contracts/PortfolioMain.sol SHA3: 43faea5999cf5b6cd70a15515bea3a0889ff1a8e72a8035a839abf54ef994f2b File: ./contracts/PortfolioMinter.sol SHA3: ecb526ba2f6875ced91da0af4713ea11f2a2e12807822cffe7433dae38ee5686 File: ./contracts/PortfolioSub.sol SHA3: 0a374a319d83ac6fd0e6ead905b627f65aaf1d0e9cd83d9fefc862d6af81846a File: ./contracts/token/IncentiveDistributor.sol SHA3: 3f56d96740e2426b04368a9f1621db55af41b84f5b79ffbb4467e8221661a869 File: ./contracts/TradePairs.sol SHA3: 13627aa758449f59716c2fbce6f9654d570c455acbe0fae087780f8b53b91f92

Third review scope

Repository	https://github.com/Dexalot/contracts
Commit	89d40a8aa3072dd9f21a13d41b6743a6fceec9dc
Whitepaper	Link
Functional Requirements	Link
Technical Requirements	Link
Contracts Addresses	-
Contracts	File: ./contracts/bridgeApps/LzApp.sol SHA3: 07f92cdcd8e2ac0c2b663c94979168a79d716d94e7c8778d80ad1ce0072db1a9 File: ./contracts/Exchange.sol

	SHA3: 08a7a8918d272cdb0e2ff7b5eb025e24921aaa438144b6a8e838cfc3d09e613f File: ./contracts/ExchangeMain.sol SHA3: c63bc138e1dd103bff1afcadfe9c3640c9bef888838c361642795f6fc67657fa File: ./contracts/ExchangeSub.sol SHA3: 7857feca5814330d72f3f10b36c3e3904d79f41a1753641bd337879f2add367b File: ./contracts/GasStation.sol SHA3: 3f11290468a8f7165c8eafabf5e502c59940cd1efd2038205c41514c03d68b00 File: ./contracts/interfaces/IGasStation.sol SHA3: dfd84fed1e22f38b1187be7c2969fa73e1f650051dc551265c1d910b005a4ecd File: ./contracts/interfaces/INativeMinter.sol SHA3: 79d381f48d7acc73b37bd5348ce7df0531b9f4832a408a5084362fc94b6d4768 File: ./contracts/interfaces/IPortfolio.sol SHA3: def6219eaa017baf74b444246765e4c3898925ecc09cce7dac28a4bb01c3009b File: ./contracts/interfaces/IPortfolioBridge.sol SHA3: 7cb1538d2bb8af1313962e3ef619a11a46cb8e838c6909c88102d7db87473489 File: ./contracts/interfaces/IPortfolioBridgeSub.sol SHA3: 3da0244eb221fa10ba30de79600ef705614def416a28d6884278cd771f507afc File: ./contracts/interfaces/IPortfolioMain.sol SHA3: 5152c35b35b029678c2ec1e3e5b75bfe0f2232a740a0ce2f89501e5f716f5713 File: ./contracts/interfaces/IPortfolioMinter.sol SHA3: a09113c195a341f4b8d4b5d0ebfa3c064a7ba68c35930adb52c650297682ec80 File: ./contracts/interfaces/IPortfolioSub.sol SHA3: 243c1d8b4b0cde2c3d1ea5e4ecc43701e264f8d4dc268786d85f13607268ec23 File: ./contracts/interfaces/ITradePairs.sol SHA3: 25f04da14144a0f7ce759479761fe0141e00f782cc056954845723eacc2bb9f8 File: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol SHA3: 77bc094bd019313e1aaa48bcd3699d2db4af174e496f1deaf9c99f0d511e570b File: ./contracts/interfaces/layerZero/ILayerZeroReceiver.sol SHA3: 62c575ff041db59d1e1c8eea84ed441cd86dddc9546c4284affed1da9710b5bf

	File: ./contracts/interfaces/layerZero/ILayerZeroUserApplicationConfig.sol SHA3: 6864f31c800af55316e156a34ecfef7612b75cf5b1d218c6dc11afd81e667c05 File: ./contracts/OrderBooks.sol SHA3: 660f3cdf1d404b5825d818fb32471f182ff05a9c4d88ac7a7fee453f4adec3d7 File: ./contracts/others/Multicall2.sol SHA3: dcc2461ccbdf4dac26d35aa2cc6edaf7749a3836ecef5eadbc07177cd76fcac4 File: ./contracts/Portfolio.sol SHA3: 2eede1b67e16bc98865f1250d376f16de5354f008424098a8bb7b354e5e3e90a File: ./contracts/PortfolioBridge.sol SHA3: f495fcb2ff5fe549043f705e51e5e319b8931892d51d39244325b877987c2967 File: ./contracts/PortfolioBridgeSub.sol SHA3: 2b4311a0dc7fa43e0069a6c775a8d7f7e144af357fc92ee52e07a9e28f7096fb File: ./contracts/PortfolioMain.sol SHA3: 3d56add857a1b15fbd4d6e2e2d0b42d07a431b9cf158bb9f46d3198b9371a052 File: ./contracts/PortfolioMinter.sol SHA3: ecb526ba2f6875ced91da0af4713ea11f2a2e12807822cffe7433dae38ee5686 File: ./contracts/PortfolioSub.sol SHA3: 9fc47fc7c39b02e87a8e3e4474597d869624bd79ebf5753e639697e8ffc081dd File: ./contracts/token/IncentiveDistributor.sol SHA3: 3f56d96740e2426b04368a9f1621db55af41b84f5b79ffbb4467e8221661a869 File: ./contracts/TradePairs.sol SHA3: 554d8ce21754e6952449f468fc4d71f9fbc7b3849a680d383c0544c1964b2066

Fourth review scope

Repository	https://github.com/Dexalot/contracts
Commit	4c8e033e01a3f3b1589739ef906f49047c3966b4
Whitepaper	Link
Functional Requirements	-
Technical Requirements	Link

Contracts Addresses	-
Contracts	File: ./contracts/bridgeApps/LzApp.sol SHA3: 07f92cdcd8e2ac0c2b663c94979168a79d716d94e7c8778d80ad1ce0072db1a9 File: ./contracts/Exchange.sol SHA3: 08a7a8918d272cdb0e2ff7b5eb025e24921aaa438144b6a8e838cfc3d09e613f File: ./contracts/ExchangeMain.sol SHA3: c63bc138e1dd103bff1afcadfe9c3640c9bef888838c361642795f6fc67657fa File: ./contracts/ExchangeSub.sol SHA3: 7857feca5814330d72f3f10b36c3e3904d79f41a1753641bd337879f2add367b File: ./contracts/GasStation.sol SHA3: 3f11290468a8f7165c8eafabf5e502c59940cd1efd2038205c41514c03d68b00 File: ./contracts/interfaces/IGasStation.sol SHA3: dfd84fed1e22f38b1187be7c2969fa73e1f650051dc551265c1d910b005a4ecd File: ./contracts/interfaces/INativeMinter.sol SHA3: 79d381f48d7acc73b37bd5348ce7df0531b9f4832a408a5084362fc94b6d4768 File: ./contracts/interfaces/IPortfolio.sol SHA3: b0712cf2171fc305bff339304cbc944001452192431fc3cbb467625f8123c6cb File: ./contracts/interfaces/IPortfolioBridge.sol SHA3: 7cb1538d2bb8af1313962e3ef619a11a46cb8e838c6909c88102d7db87473489 File: ./contracts/interfaces/IPortfolioBridgeSub.sol SHA3: 3da0244eb221fa10ba30de79600ef705614def416a28d6884278cd771f507afc File: ./contracts/interfaces/IPortfolioMain.sol SHA3: 5152c35b35b029678c2ec1e3e5b75bfe0f2232a740a0ce2f89501e5f716f5713 File: ./contracts/interfaces/IPortfolioMinter.sol SHA3: a09113c195a341f4b8d4b5d0ebfa3c064a7ba68c35930adb52c650297682ec80 File: ./contracts/interfaces/IPortfolioSub.sol SHA3: 243c1d8b4b0cde2c3d1ea5e4ecc43701e264f8d4dc268786d85f13607268ec23 File: ./contracts/interfaces/ITradePairs.sol SHA3: 25f04da14144a0f7ce759479761fe0141e00f782cc056954845723eacc2bb9f8 File: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol

	SHA3: 77bc094bd019313e1aaa48bcd3699d2db4af174e496f1deaf9c99f0d511e570b File: ./contracts/library/Bytes32LinkedListLibrary.sol SHA3: e798302aecc7346b0e2a5cbc7632b3fa11a9ea5263012951639e5453cf738eec File: ./contracts/library/RBTLibrary.sol SHA3: 800295acffd379ca0497ff58e6fe37e7b07f778d9b25e623bdd298e29496a082 File: ./contracts/library/UtilsLibrary.sol SHA3: 02fa557c9158493d4fe5f991d61834ee3cf63f9980aaac1ebc19598b3cfb687c File: ./contracts/OrderBooks.sol SHA3: 871bedf29e3b81e8bcba65eb9550b243f16e0275759b93e5abd75ec6d98e8eb1 File: ./contracts/Portfolio.sol SHA3: 2eede1b67e16bc98865f1250d376f16de5354f008424098a8bb7b354e5e3e90a File: ./contracts/PortfolioBridge.sol SHA3: f495fcb2ff5fe549043f705e51e5e319b8931892d51d39244325b877987c2967 File: ./contracts/PortfolioBridgeSub.sol SHA3: eb66a3acbb15132d48173b15a02486f45f8a339f7bff49b52344d3d3453d45a7 File: ./contracts/PortfolioMain.sol SHA3: 3d56add857a1b15fbd4d6e2e2d0b42d07a431b9cf158bb9f46d3198b9371a052 File: ./contracts/PortfolioMinter.sol SHA3: ecb526ba2f6875ced91da0af4713ea11f2a2e12807822cffe7433dae38ee5686 File: ./contracts/PortfolioSub.sol SHA3: 1e626f3cda3618a87773ace3fd66f8bd7a1d186d1dd21b4f4130332df7b610c8 File: ./contracts/token/IncentiveDistributor.sol SHA3: 3f56d96740e2426b04368a9f1621db55af41b84f5b79ffbb4467e8221661a869 File: ./contracts/TradePairs.sol SHA3: 6630bad7915e0676670d8bfa1be4cc70be7b4b3897974207659c2448aef1586c

Fifth review scope

Repository	https://github.com/Dexalot/contracts
Commit	91b2b26ffe8cdc9b5411836621465db6d4299ed0
Whitepaper	Link

Functional Requirements	-
Technical Requirements	Link
Contracts Addresses	-
Contracts	File: ./contracts/bridgeApps/LzApp.sol SHA3: 07f92cdcd8e2ac0c2b663c94979168a79d716d94e7c8778d80ad1ce0072db1a9 File: ./contracts/Exchange.sol SHA3: 08a7a8918d272cdb0e2ff7b5eb025e24921aaa438144b6a8e838cfc3d09e613f File: ./contracts/ExchangeMain.sol SHA3: c63bc138e1dd103bff1afcadfe9c3640c9bef888838c361642795f6fc67657fa File: ./contracts/ExchangeSub.sol SHA3: be80796088b9e50708dbd8758f1a86bfd4b368848f7d41525b036897b76998c6 File: ./contracts/GasStation.sol SHA3: 3f11290468a8f7165c8eafabf5e502c59940cd1efd2038205c41514c03d68b00 File: ./contracts/interfaces/IGasStation.sol SHA3: dfd84fed1e22f38b1187be7c2969fa73e1f650051dc551265c1d910b005a4ecd File: ./contracts/interfaces/INativeMinter.sol SHA3: 79d381f48d7acc73b37bd5348ce7df0531b9f4832a408a5084362fc94b6d4768 File: ./contracts/interfaces/IPortfolio.sol SHA3: b0712cf2171fc305bff339304cbc944001452192431fc3cbb467625f8123c6cb File: ./contracts/interfaces/IPortfolioBridge.sol SHA3: 7cb1538d2bb8af1313962e3ef619a11a46cb8e838c6909c88102d7db87473489 File: ./contracts/interfaces/IPortfolioBridgeSub.sol SHA3: 3da0244eb221fa10ba30de79600ef705614def416a28d6884278cd771f507afc File: ./contracts/interfaces/IPortfolioMain.sol SHA3: 5152c35b35b029678c2ec1e3e5b75bfe0f2232a740a0ce2f89501e5f716f5713 File: ./contracts/interfaces/IPortfolioMinter.sol SHA3: a09113c195a341f4b8d4b5d0ebfa3c064a7ba68c35930adb52c650297682ec80 File: ./contracts/interfaces/IPortfolioSub.sol SHA3: 243c1d8b4b0cde2c3d1ea5e4ecc43701e264f8d4dc268786d85f13607268ec23 File: ./contracts/interfaces/ITradePairs.sol

	SHA3: 5f4317d54e41348435360680d3019e575f90479e41c53121612a9971022bc8a8 File: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol SHA3: 77bc094bd019313e1aaa48bcd3699d2db4af174e496f1deaf9c99f0d511e570b File: ./contracts/library/Bytes32LinkedListLibrary.sol SHA3: e798302aecc7346b0e2a5cbc7632b3fa11a9ea5263012951639e5453cf738eec File: ./contracts/library/RBTLibrary.sol SHA3: 800295acffd379ca0497ff58e6fe37e7b07f778d9b25e623bdd298e29496a082 File: ./contracts/library/UtilsLibrary.sol SHA3: 02fa557c9158493d4fe5f991d61834ee3cf63f9980aaac1ebc19598b3cfb687c File: ./contracts/OrderBooks.sol SHA3: 871bedf29e3b81e8bcba65eb9550b243f16e0275759b93e5abd75ec6d98e8eb1 File: ./contracts/Portfolio.sol SHA3: 2eede1b67e16bc98865f1250d376f16de5354f008424098a8bb7b354e5e3e90a File: ./contracts/PortfolioBridge.sol SHA3: f495fcb2ff5fe549043f705e51e5e319b8931892d51d39244325b877987c2967 File: ./contracts/PortfolioBridgeSub.sol SHA3: eb66a3acbb15132d48173b15a02486f45f8a339f7bff49b52344d3d3453d45a7 File: ./contracts/PortfolioMain.sol SHA3: 3d56add857a1b15fbd4d6e2e2d0b42d07a431b9cf158bb9f46d3198b9371a052 File: ./contracts/PortfolioMinter.sol SHA3: ecb526ba2f6875ced91da0af4713ea11f2a2e12807822cffe7433dae38ee5686 File: ./contracts/PortfolioSub.sol SHA3: aab6f2864dd9b802c6ca0cd2c352b7c98eb1bbc13b3a9776d9bd838b24c5c063 File: ./contracts/TradePairs.sol SHA3: 444abc620fc54433fb58e46dbeacc1cfa84eb3c9ac437c5b85e0ba978a0cb869

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are fully provided.
- Technical descriptions of the contracts are sufficient.

Code quality

The total Code Quality score is **9** out of **10**.

- The development environment is well configured, and the code is well documented and covered by tests.
- The code readability suffers due to contract size limitations that limit potential abstraction.

Test coverage

Test coverage of the project is **95.57%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Some test branches are missing in the Bytes32LinkedListLibrary, TradePairs and most Portfolio* contracts.

Security score

As a result of the audit, the code contains no issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.6**.

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
23 November 2022	9	8	5	1
26 December 2022	1	2	0	0
30 December 2022	0	0	0	0
10 January 2023	0	0	1	1
19 January 2023	0	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and more specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed

Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed

Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Not Relevant
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

System Overview

Dexalot is an on-chain exchange system that manages built-in order book implementation by supporting different ERC20 tokens. The system relies on the following contracts:

- *DexalotSubnetBalances* — a contract that stores the balances as Merkle roots.
- *Exchange* — an abstract management-wrapper contract that offers varying access privileges. It is inherited by the ExchangeMain and ExchangeSub contracts.
- *ExchangeMain* — a contract that pulls the price feeds from the provided oracle contract address. It has a flipping coin implementation that benefits from the oracle price feed result.
- *ExchangeSub* — a contract that allows the owner to manage the trade pairs and order books.
- *GasStation* — a contract that is used by the PortfolioSub contract to swap ERC20 tokens with the subnet's native coin to send users native coins for Gas.
- *OrderBooks* — a contract implementing limit order books with time and price priority.
- *Portfolio* - an abstract contract that manages the bridge provider and its features. It is used by both PortfolioSub and PortfolioMain.
- *PortfolioMain* — a contract for users to deposit native or ERC20 tokens. Deposited amounts are sent to the portfolio bridge contract.
- *PortfolioSub* - a contract that helps native token deposits and withdrawals and execution of orders between traders.
- *TradePairs* - Implements data structures and functions for trade pairs (e.g. AVAX/USDT).
- LzApp - generic Layer Zero app copied from the LayerZero example contracts.
- *Bytes32LinkedListLibrary* - heavily modified circular FIFO LinkedList implementation.
- *RBTLibrary* - Red-Black Tree binary search library to store and access a sorted list of unsigned integer data. An algorithm rebalances the binary tree, resulting in O(log n) insert, remove and search times. Modified with Gas optimizations.
- *UtilsLibrary* - common utility functions used in all Dexalot contracts
- *IncentiveDistributor* - distributor for Dexalot incentives that distributes 200k $ALOT tokens monthly and other tokens based on off-chain calculated usage reports.

Privileged roles

- The admin of the LzApp contract can:
  - set the Layer Zero Endpoint address
  - set the send/receive message version

  - set the Layer Zero trusted remote address
  - force resumes the stuck bridge
  - retry the stuck message in the bridge
- The owner of the IncentiveDistributor contract can:
  - add a reward token
  - withdraw reward token balances of the contract
  - pause/unpause the contract
- The DEFAULT_ADMIN_ROLE of the Exchange contract can:
  - set the portfolio address
  - add/remove multiple default admin roles
  - add/remove AUCTION_ADMIN_ROLE roles
  - pause/unpause the portfolio contract
  - add/remove trusted contract addresses to/from the portfolio contract
- The AUCTION_ADMIN_ROLE of the Exchange contract can:
  - add new trade token addresses to the portfolio contract
- The ONBEHALF_OF role of the TradePairs contract can:
  - add Orders instead of the _trader
- The DEFAULT_ADMIN_ROLE role of the TradePairs contract can:
  - add trade pairs
  - pause/unpause the contract
  - pause the trade activity for a specific token pair
  - pause adding a new order
  - Set a trade pair as only post. (No matching)
  - set the auction mode
  - set the min trade amount
  - define order types for the trade pairs
  - set the display decimals of the base or the quote asset
  - set the maker and taker rates
  - set the allowed slippage percent
  - match auction orders
  - cancel orders by starting from the less profit order in the orderbook.
- The admin of the GasStation contract can:
  - set the Gas amount
  - pause/unpause the contract
  - withdraw the native balance.
- The admin of the Portfolio contract can:
  - set the portfolio bridge contract address
  - enable/disable the bridge provider’s activity
  - force resume receive action, wiping the existing message
  - retry the stuck message in the LZ bridge
  - revoke admin and bridge roles
  - pause/unpause the contract

    - set bridge fees and swap amounts
    - add and remove trusted contracts
    - add and remove tokens
- The admin of the PortfolioMain contract can:
    - recover the LZ payload
    - collect bridge fees (native and ERC20)
- The admin of the PortfolioSub contract shares most functions with the admin of the Portfolio, but can:
    - set up the portfolio bridge contract
    - change bridge providers
- The PORTFOLIO_BRIDGE_ROLE of the PortfolioSub contract can:
    - process DEPOSIT messages since they are the only ones being sent to portfolio sub
- The PORTFOLIO_BRIDGE_ROLE of the PortfolioMain contract can:
    - process WITHDRAW messages as it is the only message that can be sent to the portfolio main
- The admin of the PortfolioMinter contract can pause/unpause the contract. The minter role can mint native tokens and this privilege must be the Portfolio contract.
- The WRITER_ROLE of the DexalotSubnetBalances can set the Merkle root hashes and public IPFS links that store the user balances in a Merkle tree.
- The ExchangeMain contract is the admin of PortfolioMain.
- The ExchangeSub contract is the admin of PortfolioSub and TradePairs.
- The Exchange contract acts as the AuctionManager using AUCTION_ADMIN_ROLE.
- The admin of the OrderBooks contract can set the trade pair addresses
- All the functions pertaining to Auction can be called directly in TradePairs and Portfolio using DEFAULT_ADMIN_ROLE but not recommended because certain actions require a synchronised update to both Portfolio and TradePairs contracts.
- ExchangeSub needs to have a DEFAULT_ADMIN_ROLE on TradePairs. TradePairs should have an EXECUTOR_ROLE on OrderBooks

Risks

- The code contains **out-of-scope** 3rd party contracts that could not be verified within the scope of the audit. (LayerZeroEndpoint)
- Due to the dual-chain nature of the system, unexpected behavior could arise from the **off-chain out-of-scope** parts of the system responsible for bridging. These could lead to user fund manipulation and losses.
- The system relies on requests received from the bridge. The **reliability and security of the bridge are not verifiable** by the current audit.

Recommendations

- The DEFAULT_ADMIN_ROLE role is central to the system. It is recommended that this address be a multisig with at least ⅗ signatures required where an EOA is set for this role.

Findings

■■■■ Critical

1. Weak Source of Randomness

In the ExchangeMain contract, a coinflip is generated based on the AVAX/USDT market data. The purpose of this is unclear and not used anywhere within the scope.

Using random values that users can affect or predict is not secure because the impact can be used for profit.

Path: ./contracts/bridgeApps/ExchangeMain.sol

Recommendation: The random values should be obtained from external, provably random sources.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The issue is mitigated since it is not used in the project. However it is used in the out-of-scope off-chain code.)

2. Incorrect Calculations

When a deposit transaction is done in processXFerPayload function and if the trader’s account needs Gas token to be sent and the trader has enough amount in asset[trader][native].available, Gas amount is not deducted from the tokenTotals although it’s deducted from the asset.available and asset.total.

Since there always will be accumulated leftover amounts in tokenTotals, the condition of “tokenTotals[_symbol] == 0” on line 774 is never going to be matched. This will lead owner to not be able to remove the tokens.

Path: ./contracts/PortfolioSub.sol: processXFerPayload()

Recommendation: Do deduct the gasAmount that is sent to user from the tokenTotals.

Status: Mitigated (Revised commit: 91b2b26ffe8cdc9b5411836621465db6d4299ed0) (The Customer stated that it’s not an issue as it’s a subject only for the native token. Native tokens will never be removed.)

■■■ High

3. Non-Finalized Code

On line 95, *version* number is hardcoded and there are no version options other than “1”. Therefore, it has no effect on the implementation.

Path: ./contracts/bridgeApps/LzApp.sol: lzEstimateFees()

Recommendation: Remove the redundant variable or allow the possibility of other versions.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The customer provided documentation in the code that LayerZero contract needs v1 in adapterParams to specify a higher gas for the destination to receive a transaction. However, it should be noted here that the LayerZero contract is not verified.)

4. Denial of Service Vulnerability

If the number of tokens reaches a large enough size, it can cause the *withdrawFees* function to fail due to excessive Gas.

Path: ./contracts/PortfolioSub.sol: withdrawFees

Recommendation: Instead of iterating over all tokens, follow a pull-over-push pattern and do the withdrawal according to the given token address parameter or limit the number of tokens.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

5. Highly Permissive Role Access

Owners should not have access to funds that belong to users.

The refund functionality can be used to siphon user funds without providing bridge proof.

Path: ./contracts/PortfolioBridge.sol: refundTokens, refundNative

Recommendation: Remove the highly permissive withdrawal functionality.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The Customer clarified that the bridge contract is going to be funded by only the owners.)

6. Highly Permissive Role Access

The admin of the TradePairs contract can cancel pending auction orders without user permission.

This may cause manipulations of orders by the admin privilege.

Path: ./contracts/TradePairs.sol: unsolicitedCancel()

Recommendation: Do not allow the admin to cancel pending orders.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

7. EIP Standard Violation

Hashing structured data is non-trivial, and errors result in the loss of the security properties of the system.

Chain ID and the contract address are not used as part of the signature. It may result in double-spending if the contract is deployed on multiple networks.

Path: ./contracts/token/IncentiveDistributor.sol: _checkClaim()

Recommendation: Use the EIP-712 standard for hashing and signing of typed structured data.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

07f523b40089ab4aa55cd98398f7eabd963e819)

8. **Data inconsistency**

When the admin alters the auction mode of a trade pair from the ExchangeSub contract, auction mode of the related base token is automatically being updated. However, if the admin does it from the TradePairs contract base token's auction mode will not be updated.

This may lead conflicts when the changes are done with TradePairs contract.

Path: ./contracts/ExchangeSub.sol: setAuctionMode()

./contracts/TradePairs.sol: setAuctionMode()

Recommendation: Do not allow updating the auction mode of base token from two different implementations/contracts.

Status: Fixed (Revised commit: 91b2b26ffe8cdc9b5411836621465db6d4299ed0)

■■ Medium

1. **Unfinalized Code**

The implementation contains a test oracle address on line 36 and a comment on line 34 that specifies that the address is going to be updated later.

This makes the code look unfinished.

Path: ./contracts/contracts/ExchangeMain.sol : function()

Recommendation: Finalise its implementation with the main oracle address.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

2. **Missing Check Validation**

When adding a new pair, the base display decimal is not checked against the base decimal.

In case of an event where the base display decimal is greater than the base decimal, the *decimalsOk* function will always revert, and adding an order will not be possible.

The same check statement is missing in the *addLimitOrder* function for quote decimals.

Path: ./contracts/TradePairs.sol: addTradePair(), addLimitOrder()

Recommendation: In the addTradePair and addLimitOrder functions, put a require statement that checks the display decimals against the base/quote decimals.

Status: Fixed

3. **Data Inconsistency**

Token details with the same symbol are recorded twice in two different contracts; Portfolio and PortfolioBridge. In the PortfolioBridge contract, different token information from the one in the Portfolio contract can be saved.

Path: ./contracts/PortfolioBridge.sol: addToken()

Recommendation:

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

4. **Unused Enum Variable**

The BridgeProvider type CELER is not validated anywhere in the project, and there is not any implementation for it.

Path: ./contracts/PortfolioBridge.sol

Recommendation: Consider CELER status or remove it from the project.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The Customer did not remove it on purpose to run their test cases to make sure that a message sent to a disabled bridge fails as expected.)

5. **Code Duplication**

Two different functions are detected for the same operation, setting the LZ trusted remote address. Moreover, one records the path without encoding while the other one is encoding.

Path: ./contracts/bridgeApps/LzApp.sol: setLZTrustedRemote, setLZTrustedRemoteAddress

Recommendation: Remove one of the functions.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

6. Redundant Function Implementation

The applied abstraction strategy for inheritance in the Portfolio contracts includes unnecessary interface components.

This can lead to users calling unimplemented functions on a contract, and result in a loss of funds or data corruption.

Path: ./contracts/PortfolioSub.sol : addIERC20, getToken, depositToken, depositTokenFromContract

Recommendation: Improve the Portfolio abstraction or implement reverts on the empty functions.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

7. Best Practice Violation

The implementation of the OrderBooks mechanism is unusual. For each pair e.g. AVAX/USDT there can be a few combinations of order books: AVAX/USDT-BUYBOOK, AVAX/USDT-SELLBOOK, USDT/AVAX-BUYBOOK and USDT/AVAX-SELLBOOK.

This introduces unnecessary complexity into the order fulfilment process.

Path: ./contracts/TradePairs.sol: addTradePair()

Recommendation: Make the system match 1-to-1 with the way TradePairs are set up.

Status: Fixed (Revised commit: 89d40a8aa3072dd9f21a13d41b6743a6fceec9dc)

Low

1. Typos in the Code

On line 55, the quoteSymbol parameter is misspelt.

Path: ./contracts/interfaces/ITradePairs.sol

Recommendation: Fix the syntax issue.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

2. Redundant Require Statement

Trusted contracts are already checked in the *depositToken* function. Therefore line 164 is redundant in the *depositTokenFromContract* contract.

Redundant code decreases code readability and causes higher consumption of Gas.

Path: ./contracts/PortfolioMain.sol: depositTokenFromContract()

Recommendation: Remove the redundant require statement.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The issue is mitigated due to the Customer's design preference)

3. **Redundant Comparison**

   To revert the function in an unintended scenario, use *revert()* instead of *require(1 == 0)*.

   Path: ./contracts/PortfolioBridge.sol: sendXChainMessageInternal

   Recommendation: Use the revert function instead of the require statement.

   Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

4. **Outdated Solidity Version**

   Using an outdated compiler version can be problematic, especially if publicly disclosed bugs and issues affect the current compiler version.

   Paths: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol, ./contracts/interfaces/layerZero/ILayerZeroReceiver.sol, ./contracts/interfaces/layerZero/ILayerZeroUserApplicationConfig.sol,

   Recommendation: Use a modern compiler version.

   Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

5. **Floating Pragma**

   Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

   Paths: ./contracts/interfaces/layerZero/ILayerZeroEndpoint.sol, ./contracts/interfaces/layerZero/ILayerZeroReceiver.sol, ./contracts/interfaces/layerZero/ILayerZeroUserApplicationConfig.sol,

   Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

   Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

6. **Unused Variables**

   A PBRIDGE_ROLE constant is declared but never used.

Redundant declarations decrease code readability and consume unnecessary Gas.

Path: ./contracts/Portfolio.sol

Recommendation: Remove the redundant role declaration.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

7. Unchecked Return Value

The software does not check the return value of a method or function, which can prevent it from detecting unexpected states and conditions.

An unexpected return value could place the system in a state that could lead to a crash or other unintended behaviors.

Paths: ./contracts/OrderBooks.sol : lines 467 and 440, /contracts/Portfolio.sol : lines 416 and 434, /contracts/PortfolioBridge.sol : lines 237 and 278, /contracts/TradePairs.sol : lines 360, 372 and 128

Recommendation: Ensure that all possible return values from the function are taken into account.

Status: Mitigated (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819) (The issue is mitigated based on the Customer’s design preferences.)

8. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Paths: ./contracts/GasStation.sol : requestGas(), /contracts/IncentiveDistributor.sol : initialize(), /contracts/PortfolioMain.sol : processXFerPayload()

Recommendation: Implement zero address checks.

Status: Fixed (Revised commit: 807f523b40089ab4aa55cd98398f7eabd963e819)

9. Redundant comparison

On line 384, checking the *_from.balance* twice is redundant. When the “*_from.balance >= gasStation.gasAmount() * 2 + msg.value*” condition is matched, “*_from.balance > msg.value*” will be matched already.

Paths: ./contracts/PortfolioSub.sol: depositNative()

Recommendation: Remove the first comparison of the require statement.

Status: Fixed (Revised commit: 89d40a8aa3072dd9f21a13d41b6743a6fceec9dc)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on the best industry practices at the time of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, Consultant cannot guarantee the explicit security of the audited smart contracts.