Puffer Finance

May 1st, 2024

Made with ❤ by the following Creed authors: Valentin Quelquejay and Dominik Muhs

Table of Contents

Executive Summary

This report presents the results of our engagement with Puffer Finance to review the pufETH and Puffer Pool smart contract systems. The review was conducted over three weeks, from April 10, 2024 to May 1, 2024 by Valentin Quelquejay and Dominik Muhs. A total of 25 person-days were spent.

In total, one critical and two major issues have been identified, two of which have been already addressed during the audit's time frame. The project's access control is centralized through the OpenZeppelin `AccessManager` contract. Thus, the configuration of access control permissions for the majority of functions occurs within the deployment script. Therefore, it is crucial to carefuly review the scripts, and ensure that the permissions are set correctly before deploying the protocol. Additionally, the off-chain guardians play a critical role in securing the system. It is important that the majority of guardians are trustworthy. The secure signer operating within the enclave should also be carefully reviewed.

Follow-up

The Puffer Finance team requested Creed to review PR-248. The team conducted a review of the changes on May 16, 2024. This pull request introduces the capability for the Puffer DAO to trigger arbitrary external calls through the `RestakingOperator` contract. This feature is implemented to address the current lack of standardization in AVS interfaces. To mitigate potential abuse of this mechanism, the Puffer Finance team introduced a new immutable `AVSContractsRegistry` contract. This contract maintains an allowlist of external AVS contracts that can be called through the `RestakingOperator`. Creed recommended authorizing specific function selectors in addition to the contract addresses. The addresses of AVS contracts, along with their related selectors, must be authorized by the DAO before they can be invoked. A 1-day timelock is imposed for authorizing or de-authorizing new addresses or selectors on the `AVSContractsRegistry`.

Scope and Objectives

Our initial review focused on two repositories:

- `pufETH` at commit hash `5db7863db529e007a43bacd88aa7809332027fae`, and
- `PufferPool` at commit hash `d9e7948ef18f7b03c9b98999e01eeb967597879b`.

In the follow-up engagement, we reviewed:

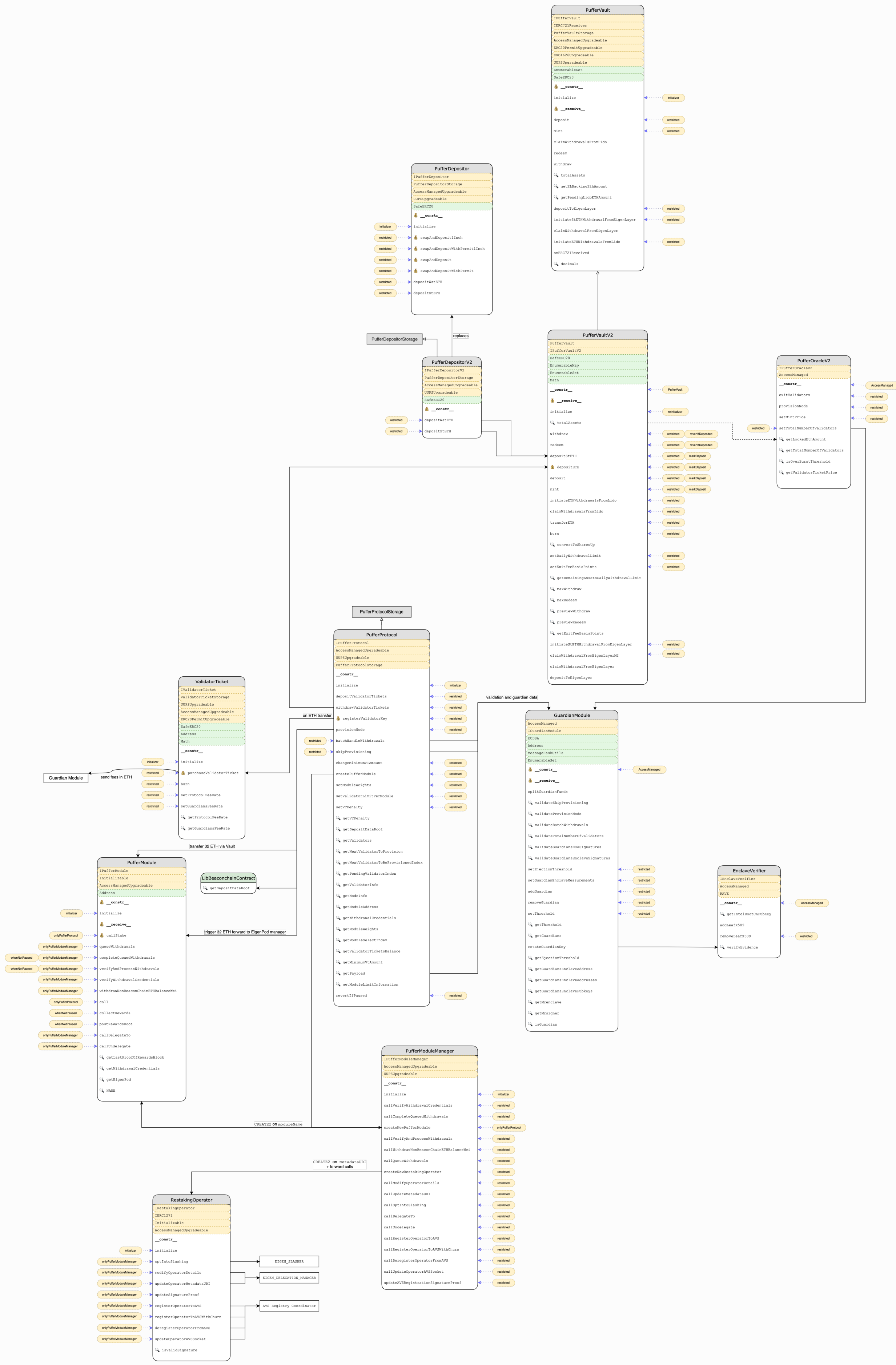
- `PufferPool` at commit hash `063c1d194ab72957b676d3546bc0934c45234ca0`.

Together with the Puffer Finance team, we identified the following priorities for our review:

- Review the security of the validator lifecycle and check for invalid/unwanted transitions,
- Review the EigenLayer integration, specifically regarding changes of the M2 upgrade,
- Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases,
- Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Security Field Guide, and the ones outlined in the EEA EthTrust Security Levels Specification.

Audit Artifacts

Audit Artifacts



Findings

Critical Malicious SGX Operator Withdrawal Credentials Frontrunning

Fixed

The Puffer Finance team provided the following statement:

> A fix has been added to Secure Signer to only allow initial deposit message signing and prevent any subsequence signing of the deposit message: https://github.com/PufferFinance/secure-signer/blob/e016932eab9ea204c9fb20baec119a06dacbc600/src/enclave/shared/mod.rs#L22-L28 Removed the bypass of the depositRootHash from the enclave node operators. Now all deposits are checked to prevent any frontrunning opportunities on the deposits: https://github.com/PufferFinance/PufferPool/pull/245/files

The `provisionNode()` function in the `PufferProtocol` contract provisions new validators. If the operator relies on an SGX enclave to run the signer, the function does not check that the `depositRootHash` matches the `depositRootHash` stored in the beacon deposit contract. Thus, even without controlling the BLS private key, which should only be accessible to the software run in the secure enclave, if the secure signer running in the enclave allows signing arbitrary messages with the BLS key, a malicious operator could sign a deposit message with different withdrawal credentials, and front-run the legitimate deposit transaction, effectively stealing 32ETH from the protocol.

Recommendation

Make sure that the signer software stored in the SGX does not allow signing arbitrary deposit messages

Major Malicious Guardian Addition/Removal DoS

Acknowledged

The Puffer Finance team provided the following statement:

> Given that all the guardians are trusted in this phase of the project, we acknowledge this issue, and fix this through the onboarding process checks. We added one step to the guardian signup which is to get them guardians to sign a message with their address, hence making sure the guardian address added is an EOA and not a smart contract to enable this attack. In the future, we might move to the pull strategy for guardians as well to mitigate this issue fully.

In `GuardianModule.sol`, the function `splitGuardianFunds()` transfers ETH to the guardians using the `sendValue()` function. This function will revert if the ETH transfer fails. The comment suggests that guardians are expected to be EOAS. Yet, it is unclear how this expectation is enforced, as there is no check at the smart contract level.

PufferPool/src/GuardianModule.sol

```
for (uint256 i = 0; i < numGuardians; ++i) {
    // slither-disable-start reentrancy-unlimited-gas
    // slither-disable-next-line calls-loop
    payable(_guardians.at(i)).sendValue(amountPerGuardian);
    // slither-disable-end reentrancy-unlimited-gas
}
```

Moreover, this approach would exclude the use of smart-contract wallets, such as multi-sigs, potentially posing a problem in the future. Assuming guardians could be smart-contract wallets, a malicious guardian could intentionally revert on all ETH transfers.

PufferPool/src/GuardianModule.sol

```
function addGuardian(address newGuardian) external restricted {
    splitGuardianFunds();
    _addGuardian(newGuardian);
}
```

```
PufferPool/src/GuardianModule.sol

function removeGuardian(address guardian) external restricted {
    splitGuardianFunds();

    (bool success) = _guardians.remove(guardian);
```

This would cause both the `removeGuardian()` and `addGuardian()` functions to revert, preventing any guardian from being added or removed from the system and effectively bricking the guardian module. Note that this would also prevent rewards from being distributed to other guardians.

Recommendation

We recommend using a 'pull' over a 'push' strategy for ETH transfers.

Major Frontrunnable Protocol Initialization

Fixed

Note: During the audit, the development team added a separate Mainnet deploy script and fixed the issue independently: PufferPool@50fd1bc1

In the Puffer deployment script, the `PufferProtocol` contract has circular dependencies with other system components. To resolve this issue, the `NoImplementation` placeholder contract is deployed first and then upgraded to the new `PufferProtocol` implementation.

In the `upgradeToAndCall` admin call to the proxy, no call data is given, however, and a separate call to `initialize` is performed after.

PufferPool/script/DeployPuffer.s.sol

```
pufferProtocol = PufferProtocol(payable(address(proxy)));

NoImplementation(payable(address(proxy))).upgradeToAndCall(address(pufferProtoco…

// Initialize the Pool
pufferProtocol.initialize({ accessManager: address(accessManager) });
```

This call can be frontrun by an attacker to initialize the protocol contract with a malicious `AccessManager` instance.

Recommendation

We recommend performing the upgrade and initialization atomically to avoid frontrunning attacks.

Medium `postRewardsRoot()` might lead to a potential loss of rewards

Acknowledged

The Puffer Finance team provided the following statement:

> The rewards won't be enabled in this upgrade as it's still unclear for EigenLayer rewards, etc. The issue will be fixed when more details are defined.

The external function `postRewardsRoot()` in `PufferModule` is used to post the root of the rewards Merkle Tree for the given module and block number. The function checks the guardian signatures to ensure the root and the corresponding block number are valid. Additionally, it requires the block number of the posted root to be greater than the block number of the last posted root. Thus, it disallows posting rewards roots out of order.

PufferPool/src/PufferModule.sol

```
if (blockNumber <= $.lastProofOfRewardsBlockNumber) {
    revert InvalidBlockNumber(blockNumber);
}
```

This means that if rewards are posted for block `x`, and then `x+j`, it is impossible to post rewards for blocks `x+i, 0<i<j`. This could cause problems if transactions are reordered, for instance, as it might prevent posting rewards for certain blocks.

Recommendation

Ideally, one should allow reward roots to be posted out-of-order. Alternatively, a sub-optimal solution would be to implement strict off-chain validation at the guardian level to ensure no roots can be signed before the previous one has been confirmed on-chain.

Minor Prefer OpenZeppelin's Address library instead of Solidity low-level calls

Fixed PR 248

The Puffer Finance team fixed the issue in the following PR

The function `customCalldataCall()` in the `RestakingOperator` contract utilizes a low-level `call` statement to invoke the target contract with an arbitrary payload. This function leaves the responsibility of checking the return value of the call to the caller. While this approach is functional, it would be preferable to rely on OpenZeppelin's Address library `functionCall()` function instead. Indeed, it offers two benefits over relying on Solidity low-level `call` statement: 1. It reverts if the target contract account is empty while the low-level `call` statement silently succeeds. Although this shouldn't cause issues in the current codebase, it could potentially lead to problems in the future. 2. It propagates custom errors thrown by the callee when the call reverts, which can be useful for debugging failed calls.

code_pr248/src/RestakingOperator.sol

```
function customCalldataCall(address target, bytes calldata customCalldata)
    external
    virtual
    onlyPufferModuleManager
    returns (bool success, bytes memory response)
{
    return target.call(customCalldata);
}
```

Minor

setAvsRegistryCoordinator() in AVSContractsRegistry should revert on idempotent operations

Fixed PR 248

The Puffer Finance team fixed the issue in the following PR

The function `setAvsRegistryCoordinator()` in the contract `AVSContractRegistry` does not revert when attempting to allow (resp. disallow) the specified selector `selector` of the target contract `avsRegistryCoordinator`, while it is already allowed (resp. disallowed). As a result, the function emits a `AvsRegistryCoordinatorSet` event even if the function does not modify the state of the contract.

code_pr248/src/AVSContractsRegistry.sol

```
function setAvsRegistryCoordinator(address avsRegistryCoordinator, bytes4 select…
    external
    restricted
{
    _avsRegistryCoordinators[avsRegistryCoordinator][selector] = isAllowed;
    emit AvsRegistryCoordinatorSet(avsRegistryCoordinator, selector, isAllowed);
}
```

Recommendation:

Consider reverting if the function does not modify the contract's state. This will prevent emitting misleading events.

Minor Events emitted when no state change is performed

Fixed

The Puffer Finance team provided the following statement:

> Fixed: by making sure there are no empty calls to the functions and revert if empty: https://github.com/PufferFinance/pufETH/pull/77/files

In the `PufferVaultV2` contract, the operations multisig has access to the `initiateETHWithdrawalsFromLido` and `claimWithdrawalsFromLido` functions. These functions take calldata arrays which are not checked for emptiness.

pufETH/src/PufferVaultV2.sol

```
for (uint256 i = 0; i < requestIds.length; ++i) {
    $.lidoWithdrawalAmounts.set(requestIds[i], amounts[i]);
}
emit RequestedWithdrawals(requestIds);
```

pufETH/src/PufferVaultV2.sol

```
for (uint256 i = 0; i < requestIds.length; ++i) {
    // .get reverts if requestId is not present
    expectedWithdrawal += $.lidoWithdrawalAmounts.get(requestIds[i]);

    // slither-disable-next-line calls-loop
    _LIDO_WITHDRAWAL_QUEUE.claimWithdrawal(requestIds[i]);
}

// ETH balance after the claim
uint256 balanceAfter = address(this).balance;
uint256 actualWithdrawal = balanceAfter - balanceBefore;
// Deduct from the locked amount the expected amount
$.lidoLockedETH -= expectedWithdrawal;

emit ClaimedWithdrawals(requestIds);
```

Thus, the functions' logic can be skipped and a rogue event can be emitted. A similar issue affects the `Timelock.cancelTransaction` function, which permits the cancelation of non-existent operation IDs:

pufETH/src/Timelock.sol

```
function cancelTransaction(address target, bytes memory callData, uint256 operat…
    // Community multisig can call this by via executeTransaction
    if (msg.sender != OPERATIONS_MULTISIG && msg.sender != address(this)) {
        revert Unauthorized();
    }

    bytes32 txHash = keccak256(abi.encode(target, callData, operationId));
    queue[txHash] = 0;

    emit TransactionCanceled(txHash, target, callData, operationId);
}
```

Another instance of this issue is the `EnclaveVerifier.removeLeafX509` function where non-existent leaves can be deleted and an event is emitted:

PufferPool/src/EnclaveVerifier.sol

```
function removeLeafX509(bytes32 hashedCert) external restricted {
    delete _validLeafX509s[hashedCert].modulus;
    delete _validLeafX509s[hashedCert].exponent;
    emit RemovedPubKey(hashedCert);
}
```

Minor Duplicate stETH Withdrawal Root Submissions

Acknowledged

In the `PufferVaultV2` contract, users can initiate a withdrawal of their stETH funds from the respective EigenLayer strategy. This will queue the withdrawal through the EigenLayer delegation manager and return a withdrawal root.

This withdrawal root is stored internally for later claiming. However, when adding the root, the return value is left unchecked, allowing the function call to succeed even if the given root is already in the set:

pufETH/src/PufferVaultV2.sol

```
461  $.eigenLayerWithdrawals.add(withdrawalRoot);
```

Minor DAO Lacks Exit Fee Setting Permissions

Fixed

Note: This issue has been fixed in the following revision: https://github.com/PufferFinance/pufETH/commit/175b3fb93da366c071f7018a6a7e53c8ec8cbd38

In `PufferVaultV2`, the Puffer Finance DAO should be given the ability to call `setExitFeeBasisPoints` to set the vault's exit fee:

pufETH/src/PufferVaultV2.sol

```
function setExitFeeBasisPoints(uint256 newExitFeeBasisPoints) external restricte…
    _setExitFeeBasisPoints(newExitFeeBasisPoints);
}
```

However, in the access initialization script the permission is never granted since the selector has been omitted:

pufETH/script/GenerateAccessManagerCallData.sol

```
function _getDaoSelectorsCalldataCalldata(address pufferVaultProxy) internal pur…
    // DAO selectors
    bytes4[] memory daoSelectors = new bytes4[](1);
    daoSelectors[0] = PufferVaultV2.setDailyWithdrawalLimit.selector;

    return abi.encodeWithSelector(
        AccessManager.setTargetFunctionRole.selector, pufferVaultProxy, daoSelec…
    );
}
```

Recommendation

We recommend adding the respective selector to the list. The assignment of roles in a separate script instead of in the code itself can result in hard-to-detect permission errors. We recommend reviewing all access restrictions before deployment to make sure all roles are correctly assigned.

None Guardians messages are not bound to a specific domain

Acknowledged

The Guardians' message hashes are not using the EIP-712 domain separator. Thus, the messages are not bound to a specific chain. This could mean that the messages might be replayable if the protocol is deployed on a different chain in the future with the same guardian setup. Ideally, it would be beneficial to bind the messages to a specific domain to make the protocol more robust and prevent any future issues if the protocol were eventually deployed on other chains.

PufferPool/src/LibGuardianMessages.sol

```
function _getSkipProvisioningMessage(bytes32 moduleName, uint256 index) internal…
    // All guardians use the same nonce
    return keccak256(abi.encode(moduleName, index)).toEthSignedMessageHash();
}
```

PufferPool/src/LibGuardianMessages.sol

```
function _getSetNumberOfValidatorsMessage(uint256 numberOfValidators, uint256 ep…
    internal
    pure
    returns (bytes32)
{
    return keccak256(abi.encode(numberOfValidators, epochNumber)).toEthSignedMes…
}
```

None Separate Test and Production Code

Acknowledged

The `PufferVaultV2Tests` contract is located in the source directory. To keep the code base clean and separate the test from the production code, it should be moved to the `test` directory.

File Hashes

- ./code/PufferPool/script/DeployNewPufferModule.s.sol
  - 388148afd36a9e4ec7c1dfa6198e5f7cb17ef9c434cc7e4281c73ac7e438974e
- ./code/PufferPool/script/DeployPufferOracle.s.sol
  - 03357b3ad85f7ef5f9a969f8ba6f0601b425653adfcbfc566e2fc4034111c46f
- ./code/PufferPool/script/DeployEverything.s.sol
  - 273fa0db33d71fd6dd58782d3bb071662367512f9ae64119c5740344669006d8
- ./code/PufferPool/script/DeployGuardians.s.sol
  - 7b820d88574c7479c9deb346c062094c678db46f4435c7e4cfc9015f41ce8f3e
- ./code/PufferPool/script/AddLeafX509.s.sol
  - a17758826e2e1c36521b405ccf4a9567b3a5b0f3b61f4167841051611382ba6c
- ./code/PufferPool/script/SetGuardianEnclaveMeasurements.s.sol
  - 374e1aa57e93a769115f6aaf8fa892964c42a8d90dab0d2ed4adda69c878b5d4
- ./code/PufferPool/script/DeployPuffer.s.sol
  - c05ca8bfab214c2a9285c430af826b2ba39d8c47b0bff6068db991700ec55107
- ./code/PufferPool/script/DeploymentStructs.sol
  - 78402b887d58bd37dd318607396a341e6ed3e90b8eca3a2afbae05ffa519911c
- ./code/PufferPool/script/DepositETH.s.sol
  - fe72ffe2e8764f90a2e4827bb933ca6f135f6d8f670f6f9f35136939ab708b34
- ./code/PufferPool/script/BaseScript.s.sol
  - 1e8b74745c33756faecaf77d9a282d8d49d0daca5d09d4a94b53777d4d1f1017
- ./code/PufferPool/script/SetupAccess.s.sol
  - c46e9da2913d971fc5baf101510b58df6e9bcc9c102e440f60de425221d98f44
- ./code/PufferPool/script/ReadValidators.s.sol
  - 0620a9b6202263e5b06debf0c26f1c92e7f9e5482797ee728ba285c27a24e204
- ./code/PufferPool/script/DelegateTo.s.sol
  - 0a45884c303c9a76af52b218d532a3c70f40244aa5e762f79772fb0866dac614
- ./code/PufferPool/script/DeployPufferProtocolImplementation.s.sol
  - b291f38313c923f85cf86b822e97fa28dd607e726d8c3e2a462a21b952cc386e
- ./code/PufferPool/src/LibGuardianMessages.sol
  - 20ef275b8ce80f698d5528f5a74e05fd23282b08f90629a768bf62f6f2167905
- ./code/PufferPool/src/PufferModuleManager.sol
  - 194d569ec86d80588d6a651479ef9f1ca438ad0c151fbf7d8dee1e57407d9ebe
- ./code/PufferPool/src/interface/IEnclaveVerifier.sol
  - 1de9b4493b02e6b8a6b030bc88c096972e2cd529e040e4416a63ff54479f54f9
- ./code/PufferPool/src/interface/IRegistryCoordinatorExtended.sol
  - fdb233a5a6367d41bd3503700fa830738220d3c5032855ab080313e14d72cf44

- ./code/PufferPool/src/interface/IValidatorTicket.sol
  - ac9f076e2dbd9cabd64c406e392645c8536805be62578242d742968b952dc8df
- ./code/PufferPool/src/interface/IBeaconDepositContract.sol
  - 99aac3de1a8691783984f798f429eee6d6a202fd56d36121ff0218e5ec06f028
- ./code/PufferPool/src/interface/IRestakingOperator.sol
  - ccd5b4cdbddc206e1ea402ddd7d2d36f42c2b48163d6138f8f9efbc577275f7b
- ./code/PufferPool/src/interface/IWETH.sol
  - 6e0fd67e25bdacbc7bbbade2fdc4da68b9695310f16c95ed12be6f1475e42aae
- ./code/PufferPool/src/interface/IPufferModule.sol
  - 6d143e0ff4deb48e25e2ee8ab8f92accbb310264dab06c50d74eb256ca9d9695
- ./code/PufferPool/src/interface/IPufferModuleManager.sol
  - b8f7fbc71b2aead6827e2504ebd627ca6f8a34f86b7629be729b7d88fe32641b
- ./code/PufferPool/src/interface/IPufferProtocol.sol
  - 7dd40738810465456b1e2b06681d578015ebb44728d9be90de87aa3846604a5c
- ./code/PufferPool/src/interface/IGuardianModule.sol
  - 3b48a7c0e5718cf4e323eb39937f15b19fd0a3aa96f0221c1ec3b72e6d9c6d42
- ./code/PufferPool/src/PufferProtocolStorage.sol
  - 183d055a78020383ee7b30c31d5762a922fab442237475d3c9b38700c7157df2
- ./code/PufferPool/src/Errors.sol
  - effe1495e01156217299f03f9fcda120f477211c4b70e809da68530f3b571fe3
- ./code/PufferPool/src/EnclaveVerifier.sol
  - 8039a84892027611 1a8e5c3e47048b1ed813c06d8d788fd7949d221df27caa54
- ./code/PufferPool/src/ValidatorTicket.sol
  - 7efebb165b08a346a3d01846ad87c4593b37887d0eda35419733c0f24272eabd
- ./code/PufferPool/src/PufferOracle.sol
  - 0713072cf05d62cba4740be18b1b2953c9278aaa0be2f59a880e92dedf2145bd
- ./code/PufferPool/src/struct/StoppedValidatorInfo.sol
  - 02e462da7ecda05d6ecc76cc2da8fcd78fa97e0f328e6705fbbaad17803bba41
- ./code/PufferPool/src/struct/Validator.sol
  - 3733090b908f559bcf50175767e066e87ab68572e0cd7655b0ee15d19aa7fc8c
- ./code/PufferPool/src/struct/NodeInfo.sol
  - f748dd5ecfeae58bcc80f9d5be4ed38afaee461cc18bac13ee75da89275b0dc0
- ./code/PufferPool/src/struct/ProtocolStorage.sol
  - 2a9754ca50ba1effedafae14522d560b6eb48ada9b17aa14a642b6fdbaef5a51
- ./code/PufferPool/src/struct/Status.sol
  - 14598e49a5badf71a9b41bc99003c79faa90df99cd928dedaf10d3a50b54a251
- ./code/PufferPool/src/struct/RaveEvidence.sol
  - 23b24221ad6611d71d834b9faed2bafe8029a44045493c1f194ad1b95364fe48
- ./code/PufferPool/src/struct/ValidatorKeyData.sol
  - 797d77d62cba82668ae1eb38096317cc95d40964b71271b6e216b05695f04f41
- ./code/PufferPool/src/integrations/BalancerRateProvider.sol
  - adb715cde09cd9f183e7db4ebca74bd27dceb40f4e0dc6c552e48ce17627cc91

- ./code/PufferPool/src/ValidatorTicketStorage.sol
  - 92b12047708f23ef1d0843b5056778c11271efad489b681acb8b2ca71cbdb1c5
- ./code/PufferPool/src/PufferOracleV2.sol
  - 97086d1546ed2b7413f77cb98b594e33364e1120d0cfb8928141234588ed9b21
- ./code/PufferPool/src/PufferModule.sol
  - e6d25c5ac05a9be213f2010a9b7b8db4d6aee8b715150f23596455cb74ffd6b7
- ./code/PufferPool/src/RestakingOperator.sol
  - 7289872fd7564dc6ebc9f199ddc3017343590ec087f63a28a10c1704c93f391b
- ./code/PufferPool/src/LibBeaconchainContract.sol
  - 863e2608a2c7fb92ff94f9a2f22870cf5b51e2d7e2884dac576e69a76b9ff2e7
- ./code/PufferPool/src/GuardianModule.sol
  - 60ce70372f40944569e81b8bbb8c845f0e0a6718b6855a719f734ddf3c727048
- ./code/PufferPool/src/PufferProtocol.sol
  - 7fd50a3614a74c9dbce5f99f21f892030efea2474065fdebd292d6f56b625d79
- ./code/pufETH/script/Roles.sol
  - fcdc6d6817d41d67f50b077a18489c488c4ed8e61aef5be684186d6ebdab827e
- ./code/pufETH/script/DeployPufETH.s.sol
  - 323509b219dc4a6038675f2bcdca9daccd72f25cd236f85d3789bde4abc953d4
- ./code/pufETH/script/BaseScript.s.sol
  - 70a660e619e9c3389eb53e62206461f18b167af264697428be93f43c66678e8b
- ./code/pufETH/script/GenerateAccessManagerCallData.sol
  - c714feeb269810894ae6f058134b7554529987c87f6c264db449d3a38879c10e
- ./code/pufETH/script/UpgradePufETH.s.sol
  - 280f297d65fa8ce2f5e4126f51f1a1d53c2ba4f362b626f71080350506d32f7c
- ./code/pufETH/src/interface/EigenLayer/IDelegationManager.sol
  - a45cd5d139300362d48b8835a0f0664e4f8743799e092fff46cc15302b511461
- ./code/pufETH/src/interface/EigenLayer/IEigenLayer.sol
  - 3c21ba0ff050fa43a12e19c417220f31435f109e868c91b0bfe33374db382a71
- ./code/pufETH/src/interface/EigenLayer/IStrategy.sol
  - 44d31de9ccd9c0b6f228829bac6e9707a71b748e0a8bec42abdd940cecaf45b5
- ./code/pufETH/src/interface/Lido/IStETH.sol
  - 85585fd946b70fc135f2d36488c6dec8b2b7e85d7bb99657f7914fed6ae6b450
- ./code/pufETH/src/interface/Lido/IWstETH.sol
  - 1dcf9256e90af3d6d16cb99158dce769a791291a21389ed80d43c601d58706e5
- ./code/pufETH/src/interface/Lido/ILidoWithdrawalQueue.sol
  - d6f52d07e3c3bc300fbeb4ecba96bca27d84a72f8ca6c09f01c7d9ee774307ef
- ./code/pufETH/src/interface/IPufferOracle.sol
  - b697c15e4032b336b5747438f0a8dc17acbc7a768128e884dcc696640bbfdb6f
- ./code/pufETH/src/interface/Other/ISushiRouter.sol
  - 86c7a45893cd124357d1de206a1a09ee4e25b2c5199faeb795d4ab7dbb7c0ae8
- ./code/pufETH/src/interface/Other/IWETH.sol
  - ef73ee560254e14a80a6b043e1ba8cd2a860320d6e278a69d5db648f8c994ba0

- `./code/pufETH/src/interface/IPufferDepositorV2.sol`
  - `89731f758e11987292b578c715b5097d2be35cb28ec9451cd6fc4df61c5d6e0c`
- `./code/pufETH/src/interface/IPufferDepositor.sol`
  - `2e2e9cb4780012e415f7a659ffbd910006c77e28be05976e57ac96fee8fac66f`
- `./code/pufETH/src/interface/IPufferOracleV2.sol`
  - `b4d67da62e4fba52b8b8e38b9a1998a958be131ec96bd520414cdddf594d550d`
- `./code/pufETH/src/interface/IPufferVaultV2.sol`
  - `7241e106c22027d40bd0b71b0419e7c286a1c86be7e57aef8b62fa06df43ab7d`
- `./code/pufETH/src/interface/IPufferVault.sol`
  - `eb26fe8c1d19d394721755c9a57940aeeb7ed00fdf41e7fa88b97b52831d9c8c`
- `./code/pufETH/src/NoImplementation.sol`
  - `9029116868f444a7c46a3205b94fac76aed889a0f92fcc6b8b6748e9b2158fee`
- `./code/pufETH/src/Timelock.sol`
  - `8e44bfd64f94c9151f419e52756759d8444bb28e5046d1e9bcbcd168b15f0ec7`
- `./code/pufETH/src/PufferDepositorStorage.sol`
  - `beaf244dcf1228a47aa1aa7235219ca2b16c914a0e261d6884d08f24585a4ff9`
- `./code/pufETH/src/PufferVault.sol`
  - `7ecfe5c361904805e77ee20c25211c3ea3251de8f8097589fbbf2b6cf37a3abb`
- `./code/pufETH/src/echidna/EchidnaPufferVaultV2.sol`
  - `d261fd1c662d71be0aa943702726119e477792f8912a84161aa7442a1174b9bf`
- `./code/pufETH/src/structs/PufferDeployment.sol`
  - `49a95164ca8b5f3f233860f384e113c30cd67b0cd76939dc6445ab0b49eb3685`
- `./code/pufETH/src/structs/Permit.sol`
  - `76fbee25bec43dd516415f9bda1552d8863ec3b201a31ddcaaeb9ac183f0c091`
- `./code/pufETH/src/PufferDepositor.sol`
  - `bc94998113109aead1b414e0dbcc7c799b4cc61ec93c8cb185d63e1f11b21c06`
- `./code/pufETH/src/PufferVaultStorage.sol`
  - `1fcee57b54cfa49eebcff881d25d320c0cfe3b9a20dd0586ec7833847334f2a8`
- `./code/pufETH/src/PufferVaultV2.sol`
  - `2791e47f3209f03ee7a12750550c81b7e8b77969e4b4142a5c891294c77022f8`
- `./code/pufETH/src/PufferDepositorV2.sol`
  - `72e5245d7f284ad8efa0e93435ae72246082555126076 04bb7369ce13c67ddc7`
- `./code/pufETH/src/PufferVaultV2Tests.sol`
  - `79223dcbecd59889a2042d03a8b7b9000f3f4c0b5cc529eb9ee01aa36aa16d9a`

Disclaimer

Creed ("CD") typically receives compensation from one or more clients (the "Clients") for performing the analysis contained in these reports (the "Reports"). The Reports may be distributed through other means, including via Creed publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., "third parties") – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that CD are not responsible for the content or operation of such Web sites, and that CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that CD endorses the content on that Web site or the operator or operations of that site. You are solely