SharkTeam

Move Language Security Analysis and Contract Audit — Contract Upgrade Vulnerability

Jan 4, 2023

SharkTeam, a leading blockchain security service team, offers smart contract audit services for developers. To satisfy the demands of different clients, thesmart contract audit services provide both manual auditing and automated auditing.

We implement almost 200 auditing contents that cover four aspects: high-level language layer, virtual machine layer, blockchain layer, and business logiclayer, ensuring that smart contracts are completely guaranteed and Safe.

In the previous “Top 10 Smart Contracts Security Threats” series, SharkTeam summarized and analyzed the top 10 vulnerabilities in the smart contract space based on historical smart contract security incidents. These vulnerabilities were usually found in Solidity smart contracts before, so will they be the same for Move smart contracts?

The SharkTeam [Move Language Security Analysis and Contract Audit Essentials] course series will take you step-by-step into the content, including permission vulnerabilities, re-entry vulnerabilities, logical checksum vulnerabilities, function malicious initialization, fallback attacks, proposal attacks, contract escalation vulnerabilities, manipulation of the prophecy machine, sandwich attacks, and replay attacks. The content of this chapter [contract upgrade vulnerability].

1. Brief description of contract escalation

Blockchain is tamper-proof, that is, the data saved on the blockchain cannot be changed. Smart contracts are essentially executable code data saved on the blockchain. Once a smart contract is deployed to the blockchain and becomes data on the blockchain, it has the characteristic of being tamper-proof. Even if there are bugs in the smart contract that need to be fixed or business logic changes, it cannot be directly modified on the original contract and then redeployed. This is one of the biggest differences between smart contracts and traditional applications. Therefore, smart contracts cannot be upgraded iteratively like other traditional applications, and different blockchains have different upgrade modes.

In Ethernet, there are multiple upgrade modes for smart contracts, the most important of which is the agent upgrade mode, which includes the inherited storage mode, unstructured storage mode, permanent storage mode and so on. The common point of these upgrade modes is the use of proxy contracts and logical contracts to achieve the separation mode of storing data and business

logic. The separated storage data is accessed by a proxy calling a function in the logical contract.

For the proxy upgrade mode, the conflict and overwriting of stored data should be avoided when the contract is upgraded, i.e., the new data storage structure cannot conflict with the original data storage structure, let alone overwrite the original data, otherwise, the historical data will be lost. This is also the biggest possible vulnerability in the process of contract upgrade.

2. Contract upgrade vulnerability security incidents

2.1 Uranium Finance Security Incident

On April 28, 2021, Uranium Finance, a blockchain project on the Coin Smart Chain, was attacked during the liquidity migration process, involving $50 million in funds, with the attack contract address:

0x2b528a28451e9853F51616f3B0f6D82Af8bEA6Ae.

Analysis found that the vulnerability in the Uranium project contract appears in the swap function in the UraniumPair.sol contract, a vulnerability that causes anyone to be able to transfer the digital assets in the contract at will, at the cost of a small amount. You can see that the swap ends up being a comparison between a 10^8 and a 10^6, which is an almost constant judgment, which means that all the digital assets in the contract can be emptied if the swap function is executed continuously according to a certain set of rules.

```
// this low-level function should be called from a contract which performs important safety checks
function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) external lock {
    require(amount0Out > 0 || amount1Out > 0, 'UraniumSwap: INSUFFICIENT_OUTPUT_AMOUNT');
    (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
    require(amount0Out < _reserve0 && amount1Out < _reserve1, 'UraniumSwap: INSUFFICIENT_LIQUIDITY');
    uint balance0;
    uint balance1;
    { // scope for _token{0,1}, avoids stack too deep errors
    address _token0 = token0;
    address _token1 = token1;
    require(to != _token0 && to != _token1, 'UraniumSwap: INVALID_TO');
    if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimistically transfer tokens
    if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimistically transfer tokens
    if (data.length > 0) IUraniumCallee(to).pancakeCall(msg.sender, amount0Out, amount1Out, data);
    balance0 = IERC20(_token0).balanceOf(address(this));
    balance1 = IERC20(_token1).balanceOf(address(this));
    }
    uint amount0In = balance0 > _reserve0 - amount0Out ? balance0 - (_reserve0 - amount0Out) : 0;
    uint amount1In = balance1 > _reserve1 - amount1Out ? balance1 - (_reserve1 - amount1Out) : 0;
    require(amount0In > 0 || amount1In > 0, 'UraniumSwap: INSUFFICIENT_INPUT_AMOUNT');
    { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
    uint balance0Adjusted = balance0.mul(10000).sub(amount0In.mul(16));
    uint balance1Adjusted = balance1.mul(10000).sub(amount1In.mul(16));
    require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul(1000**2), 'UraniumSwap: K');
    }

    _update(balance0, balance1, _reserve0, _reserve1);
    emit Swap(msg.sender, amount0In, amount1In, amount0Out, amount1Out, to);
}
```

We see that UniswapV2Pair.sol is written the same way in the contract, but it is a comparison of two 10^6's.

```
    uint balance0Adjusted = balance0.mul(1000).sub(amount0In.mul(3));
    uint balance1Adjusted = balance1.mul(1000).sub(amount1In.mul(3));
    require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul(1000**2), 'UniswapV2: K');
    }
```

So, the cause of this incident should be that when the project owner updated and upgraded this contract, he forgot to change the last 1000^2 to 10000^2.

2.2 Audius Security Incident

On July 24, 2022, hackers used the contract upgrade vulnerability to transfer 18 million AUDIO tokens from the music streaming protocol Audius.

By analyzing the whole attack process, we found that the attacker was able to attack successfully, the fundamental reason is that the initialization function was called multiple times through the proxy contract, while the initialization function should have been called only once. Take the initialization function in the Governance contract as an example, its code is as follows.

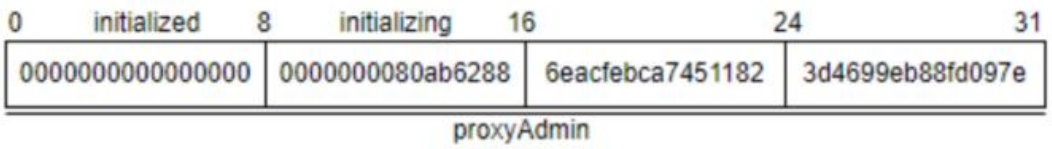
```
5434    function initialize(
5435        address _registryAddress,
5436        uint256 _votingPeriod,
5437        uint256 _executionDelay,
5438        uint256 _votingQuorumPercent,
5439        uint16 _maxInProgressProposals,
5440        address _guardianAddress
5441    ) public initializer {
5442        require(_registryAddress != address(0x00), ERROR_INVALID_REGISTRY);
5443        registry = Registry(_registryAddress);
5444
5445        require(_votingPeriod > 0, ERROR_INVALID_VOTING_PERIOD);
5446        votingPeriod = _votingPeriod;
5447
```

The initializer in Openzeppelin is used here, but in reality, the initializer does not work because the delegate call in the proxy contract implements two bool-type state variables, initialized and initializing, which are defined in the contract. The first 16 bytes of slot0 are occupied by the two bool state variables defined in the implementation contract. The first 8 bytes are initialized and the second 8 bytes are initialized.

Since the proxy contract itself defines an address type state variable proxyAdmin with the value 0x80ab62886eacfebca74511823d4699eb88fd097e, it also occupies the storage slot slot0.

```
212 contract AudiusAdminUpgradeabilityProxy is UpgradeabilityProxy {
213     address private proxyAdmin;
214     string private constant ERROR_ONLY_ADMIN = (
215         "AudiusAdminUpgradeabilityProxy: Caller must be current proxy admin"
216     );
217
```

Thus, the initialized and initializing in the implementation contract and the proxyAdmin in the proxy contract occupies the storage slot slot0 at the same time, thus causing a storage conflict. The data distribution in slot0 is as follows.



When the initialization function executes to the initializer, it reads initialized from the first 8 bytes of storage slot0, whose value is 0, i.e., false, and reads

initializing from the second 8 bytes of storage slot0, whose value is 0x80ab6288 > 0, i.e., true. therefore, the initializer does not do anything at all.

3. Move contract upgrade

Move ecology is diverse, and different Move ecological public chains have different support for contract upgrades and upgrade modes. For example, SUI currently does not support contract upgrade. Starcoin and Aptos, on the other hand, support contract upgrade, but with different upgrade strategies.

3.1 Starcoin Contract Upgrade

Starcoin supports contract upgrades, and it is more standardized and convenient than the agent contract upgrade in Solidity. This is because Starcoin has been designed with contract upgrades in mind, and has made a lot of exploration.

(1) Starcoin's account model is designed to support contract upgrades.

(2) Starcoin's standard library, Stdlib, has a variety of built-in contract upgrade policies (which also include a policy to prohibit upgrades) for users to freely choose from.

(3) Starcoin's standard library Stdlib contains a complete DAO on-chain governance function, which can be easily combined with contract upgrade policies to constrain contract upgrades through DAOs.

(4) Support contract upgrade account model. starcoin has a ModuleId data structure, which stores the address and Identifier (module name) of the account, and then does a hash calculation on the ModuleId (i.e. ModuleId hash), and uses it as a unique index to map to the real contract code. Starcoin's contracts and other resources (Tokens, NFT …) are stored in the account address, so when calling a contract you need to find it by the owner address + module. If the contract is upgraded, it will not affect the address and module name of the calling contract.

(5) Stdlib contract upgrade strategy. starcoin supports 4 contract upgrade strategies, leaving the choice to the user.

(1) STRATEGY_ARBITRARY: random update

(2) STRATEGY_TWO_PHASE: TwoPhaseUpgrade

(3) STRATEGY_NEW_MODULE: Only new Modules can be added, not modified.

(4) STRATEGY_FREEZE: Freeze, not allowed to update the contract

These 4 upgrade strategy restrictions are getting stricter and stricter, allowing only the low strategy to the high strategy settings, not the other way around. The default contract upgrade strategy is STRATEGY_ARBITRARY.

(6) DAO model contract upgrade scheme. starcoin contract upgrade scheme uses DAO decentralized management, the community can decide the deployment of contract upgrade plan by voting operation, etc.. Code submission is a two-stage submission: first submit the upgrade plan, and then submit the updated code. The whole upgrade process is divided into seven stages.

(1) PENDING: After modifying the code and submitting a contract upgrade proposal transaction to the DAO, the whole process enters the PENDING state. Set a period of time to enable the community to discuss and understand the

issue and then move on to the next stage.

(2) ACTIVE: After the PENDING phase, it enters the ACTIVE phase, in which people from the community are required to vote, and then moves to the next phase after reaching the specified time set.

(3) AGREED: After the ACTIVE phase reaches the specified time, the process enters the AGREED phase, in which the voting results will be counted, and if it exceeds the predetermined percentage, the upgrade plan is considered to be allowed by the DAO community, and the next phase can be carried out after the initiation of public notice.

(4) QUEUED: After the launch of public notice in the AGREED stage, the process enters the public notice period, this stage is mainly to display the information of the sponsor and the proposal, etc., when the public notice period has passed, enter the next stage.

(5) EXECTABLE: After the public notice period in the QUEUED phase, the process enters the first phase of the Two-phase (two-phase submission), in which the contract code upgrade plan is submitted, and then enters the next phase.

(6) ETRACTED: After the submission of the contract upgrade plan in the EXECTABLE phase, the process enters the second phase of the Two-phase (two-phase submission) of the upgrade contract, in which the code to repair or upgrade the contract can be submitted, and then can enter the next phase.

(7) Upgrade complete: After the code is submitted in the ETRACTED phase, the entire contract upgrade process is completed when the transaction is confirmed, after which the new contract can be used.

Starcoin contract upgrades do not necessarily have to be managed by DAO, but can be managed directly by the contract owner or developer.

3.2 Aptos Contract Upgrade

The Move contract in Aptos supports upgrades, i.e. the contract owner or developer can publish a new contract code to replace the old one at the same

account address, and Aptos will automatically accept and interact with the latest version of the contract code.

(1) Contract upgrade strategy

Aptos supports two upgrade strategies.

(1) compatible: the upgraded contract must maintain backward compatibility with the storage and API.

For storage, the new contract code must be compatible with all old contract structures (especially resources), which ensures that the new code can correctly interpret the existing storage state. New contracts can add new structure declarations.

Public API, the new contract in the public function must be compatible with the old contract in the public function, that is, with the old contract in the public function of the same signature, in addition to adding new functions, including public functions and entry (entry) function.

(2) unchanging policy immutable: never allow updates to the contract, that is, does not support contract upgrades.

(2) Contract upgrade method

To upgrade the deployed Move contract, simply redeploy the new contract code at the same address where it was previously deployed. Of course, you can also combine DAO to manage the contract upgrade, which will make the contract upgrade more secure.

4. Contract upgrade security analysis

(1) Different chains have different contract upgrade strategies and plans, so you need to select the upgrade plan that best fits the chain and the project according to the actual situation of the chain.

(2) The choice of upgrade strategy is decided by the user himself, and the strategy level should be fully considered. For example, each strategy in Aptos is account level, that is, all modules under the same account share the same

upgrade strategy, and it is not possible to set an independent strategy for each module.

(3) Consider compatibility. In the new contract, modules, resources, public APIs, etc. should have some compatibility with the old contract, otherwise accidents are likely to happen, and even affect the business functions of the project after the contract upgrade is successful.

(4) DAO upgrade scheme should consider the security of DAO to avoid being operated by high authority accounts; if the contract owner or developer manages the contract upgrade alone, the centralization risk must be considered.

About Us

SharkTeam's vision is to fully secure the Web3 world. The team consists of experienced security professionals and senior researchers from around the world, well versed in the underlying theory of blockchain and smart contracts, providing services including smart contract auditing, on-chain analysis, emergency response, and more. We have established long-term partnerships with key players in various areas of the blockchain ecosystem, such as Polkadot, Moonbeam, polygon, OKC, Huobi Global, imToken, ChainIDE, etc.

SharkTeam

In Math, We Trust!

https://sharkteam.org

https://t.me/sharkteamorg

https://twitter.com/sharkteamorg