07/09/2022

FunPass smart contract

Security audit report

GETSECURE

—WORLD—

AUTHOR : oualid_pro
WEBSITE : https://getsecureworld.com

Summary

Introduction

I was reached by dendenmoshi to conduct a security audit on the FunPass smart contract in order to determine the security issues that could lead to some serious financial loses. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attack against it with the goals of:

- Identifying if a remote attacker could exploit a vulnerability in the contract to steal funds
- Determining the impact of a security issue on the business
- Check for any business logic vulnerabilities

Oualid has performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation, automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices.

Scope

The security audit was mainly performed on the following smart contract:

- FunPassAlpha.sol

However, a quick check was performed on the other libraries and smart contract used by FunPassAlpha.sol to ensure no malicious code was introduced in the source code:

- ReentrancyGuard.sol
- MerkleProof.sol
- ERC1155Supply.sol
- Ownable.sol
- ERC1155.sol
- ERC165.sol
- Context.sol
- Address.sol
- IERC1155MetadataURI.sol
- IERC1155Receiver.sol
- IERC1155.sol
- IERC165.sol

All the discussed source code could be found here:

https://etherscan.io/address/0x9d3a390c898ecf778288b69ec1f8c6db543c97e7#code

Vulnerabilities

Title	Hardcoded address (Medium)
Description	The smart contract withdraw() function use a hardcoded address to transfer the collected funds. If the destination address become unusable, the fund would be stuck on the smart contract and no money would be extracted.
Impact	Ether lost
Severity	Medium

Code location	Here is the portion of source code that creates this vulnerability:
	`function withdraw() public onlyOwner {` infinite gas
	`(bool FunPassVault,) = 0x387e664a41C146ba42DAeE0466aaF75Cda017BE5.call{`
	`value : address(this).balance`
	`}("");`
	
	`require(`
	`FunPassVault,`
	`"funds were not sent properly"`
	`);`
	`}`

How to fix it	To fix this issue you should: 1) Store that address in a variable 2) Create a function with onlyOwner modifier that set that variable.
Note	According to the owner of the smart contract the withdraw function will be used one time and it is up to the Owner to check if the address is still working correctly before sending the funds. Therefore, the severity of this vulnerability was downgraded from High to Medium.

Title	No upgradability mechanism (Medium)
Description	The smart contract does not have any upgradability mechanisms. Therefore, if you want to add mùore functionalities to it in the future, you would have to deploy a new smart contract with a new address and then perform a social migration.
Impact	The impact of this vulnerability depends on the roadmap of the project.
Severity	Medium
Code location	None
How to fix it	The short answer is to use proxies. If you want more detailed with code examples, I higly encourage you to check the link bellow of a blog post that I have written for this subject.
References	https://www.getsecureworld.com/blog/how-to-fix-a-smart-contract-all-you-need/

Title	Floating pragma (Low)
Description	Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to 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.
Impact	The impact of this vulnerability depends on the future vulnerabilities that might be introduced by the used compiler.
Severity	Low
Code location	None
How to fix it	Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.
References	https://github.com/ethereum/solidity/releases