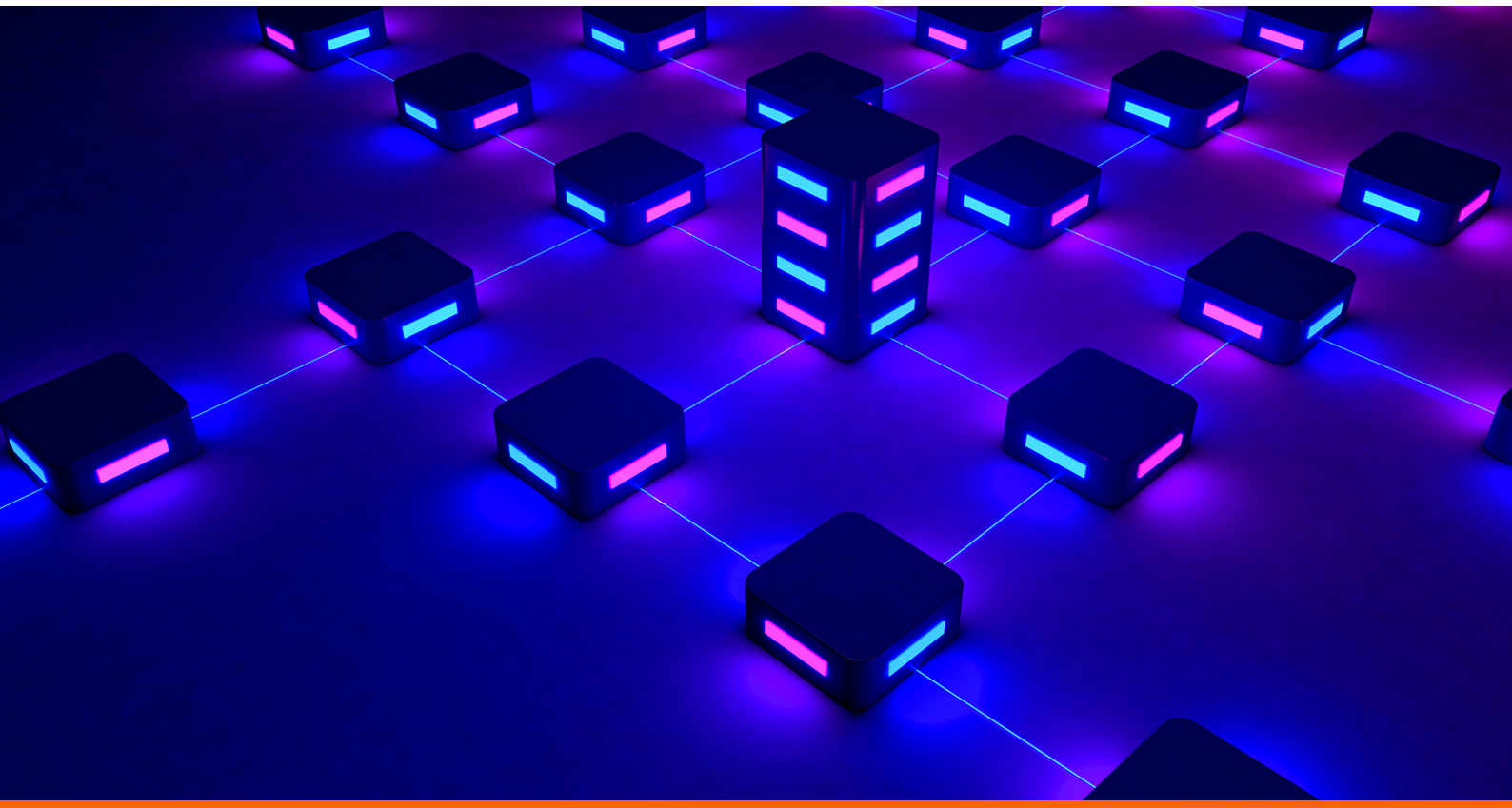
info@apriorit.com
www.apriorit.com



Case study

Evaluating Smart Contract Security for Decentralized Finance (DeFi)

Plenty, a decentralized yield farm, approached Apriorit with a request to check their smart contract security for a decentralized finance project.

As a company that works with blockchain and cryptocurrency technologies, our client has to ensure flawless security of all infrastructure elements so they can maintain a reliable reputation and gain customers' trust.

The Apriorit team performed a comprehensive security audit for the DeFi project. We detected several vulnerabilities in smart contracts and offered best practices to address them.

The customer

Our client, **Plenty**, is a platform for creating liquidity and trading FA 1.2 and FA 2 tokens on the **Tezos blockchain**. Unlike centralized exchanges, Plenty allows users to trade directly from their wallets and does not control their funds. This is why it's essential for Plenty to ensure top-notch security and performance of their smart contracts.

The challenge

Plenty approached us with a request to perform a security audit for a decentralized finance smart contract implementation. Since they work in the DeFi sector, our client needs to make sure their smart contracts operate smoothly and that users' funds are safe and sound.

Plenty wanted to receive an unbiased evaluation of their smart contract security and discover possible vulnerabilities.

The result

After performing a swap and DeFi smart contract security audit, we discovered 15 vulnerabilities:

- Two high-risk vulnerabilities related to bypassing system norms
- Six medium-risk vulnerabilities related to excessive admin rights that could possibly cause unexpected behavior
- Seven low-risk vulnerabilities that had little chance of breaking the contract's execution flow

Thanks to Apriorit's quick feedback on smart contract vulnerabilities and recommendations on how to address them, our client was able to quickly and efficiently fix the most critical issues.

Our approach

To help our client ensure the security of their product, we gathered a team of blockchain developers with experience auditing smart contracts. The team analyzed the project, defined the scope of work, planned their actions, and received our client's confirmation of the plan.

Our security audit consisted of checking the security of two types of smart contracts:

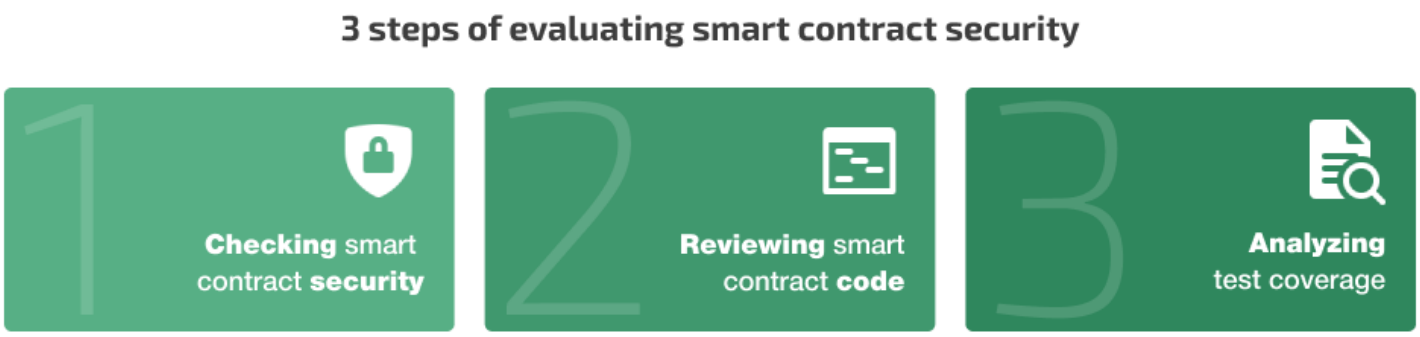
- DeFi smart contracts
- Swap smart contracts

To check the security of the client's smart contracts, we used the following methods:

- Behavioral analysis of smart contract source code
- Smart contract checks against our database of vulnerabilities and manual attacks
- Symbolic analysis of potentially vulnerable areas
- Manual code review and evaluation of code quality
- Unit test coverage analysis
- Gas usage analysis

The three major steps of our security evaluation included:

1. Checking smart contract security
2. Reviewing smart contract code
3. Analyzing test coverage



Let's start with finding vulnerabilities in smart contracts.

Smart contract security checks

First, we focused our attention on finding possible vulnerabilities in the client's smart contracts. To do that, the Apriorit team applied our security checklist that covers the main types of security risks, including unprotected functions and storage allocation exploits.

To conduct a security audit for swap and DeFi smart contracts, we used manual code analysis. Once we discovered potential vulnerabilities, we performed manual attacks to check if these vulnerabilities were exploitable. During the security audit, our team detected:

1. Four vulnerabilities in swap smart contracts: two high-risk, one medium-risk, and one low-risk. Below, we describe the most significant vulnerabilities:

Swap smart contract vulnerabilities

RISK LEVEL	VULNERABILITY	DESCRIPTION	RECOMMENDATIONS
High	A fraudulent voter can block their removal	Fraudulent voters can block all attempts to remove them from the voters list and continue to perform malicious actions	Change the architecture of the voting procedure
High	A fraudulent voter can cancel voting against them	Fraudulent voters can reset the ongoing voting process against them	Change the architecture of the voting procedure
Medium	Control over a smart contract can be lost	If the admin changes their address to an inauthentic one, control over the smart contract will be lost	When changing the admin, set a new admin address with a pending status; to finalize the operation, the new admin should accept their new role

2. Eleven vulnerabilities in DeFi smart contracts: five medium-risk and the rest low-risk. Below are the most significant:

DeFi smart contract vulnerabilities

RISK LEVEL	VULNERABILITY	DESCRIPTION	RECOMMENDATIONS
Medium	Admin has full control over distributed tokens	Admin can transfer reward tokens from any account without the owner's approval	Limit admin access to users' balances
Medium	Admin can set any parameter after voting is done	Voting doesn't verify operation parameters	Implement pending parameters so voters see upcoming changes
Medium	Admin has excessive access rights to staking tokens	Admin can transfer any tokens from a contract balance, including those that are staked by users	Limit admin access to staking tokens so that user balances can be secured and are not accessible without users' permission
Medium	Process of token minting can be skipped	Participants won't receive part of the reward, since the minting function is called after the rewards parameter is modified	Execute the minting function before modifying the rewards parameters to make sure participants receive their rewards in full
Medium	Admin can remove accounts at any time	Voting protection of operations may be compromised	Revise an admin's rights for user accounts and add voting protection

High-risk vulnerabilities can potentially allow fraudulent voters to block attempts to remove them from the voter list and reset the ongoing voting process against them. To fix these issues, we suggested a way to change the architecture of the voting procedure.

Medium-risk vulnerabilities are concerned with excessive administrator access rights that can lead to transferring reward tokens without the owner's approval, skipping the minting process, compromising the voting protection of operations, etc.

To fix these issues, we recommended our client take various actions to limit admin access rights to users' balances and staking tokens. These actions included implementing pending parameters, revising an admin's rights over user accounts, and adding voting protections.

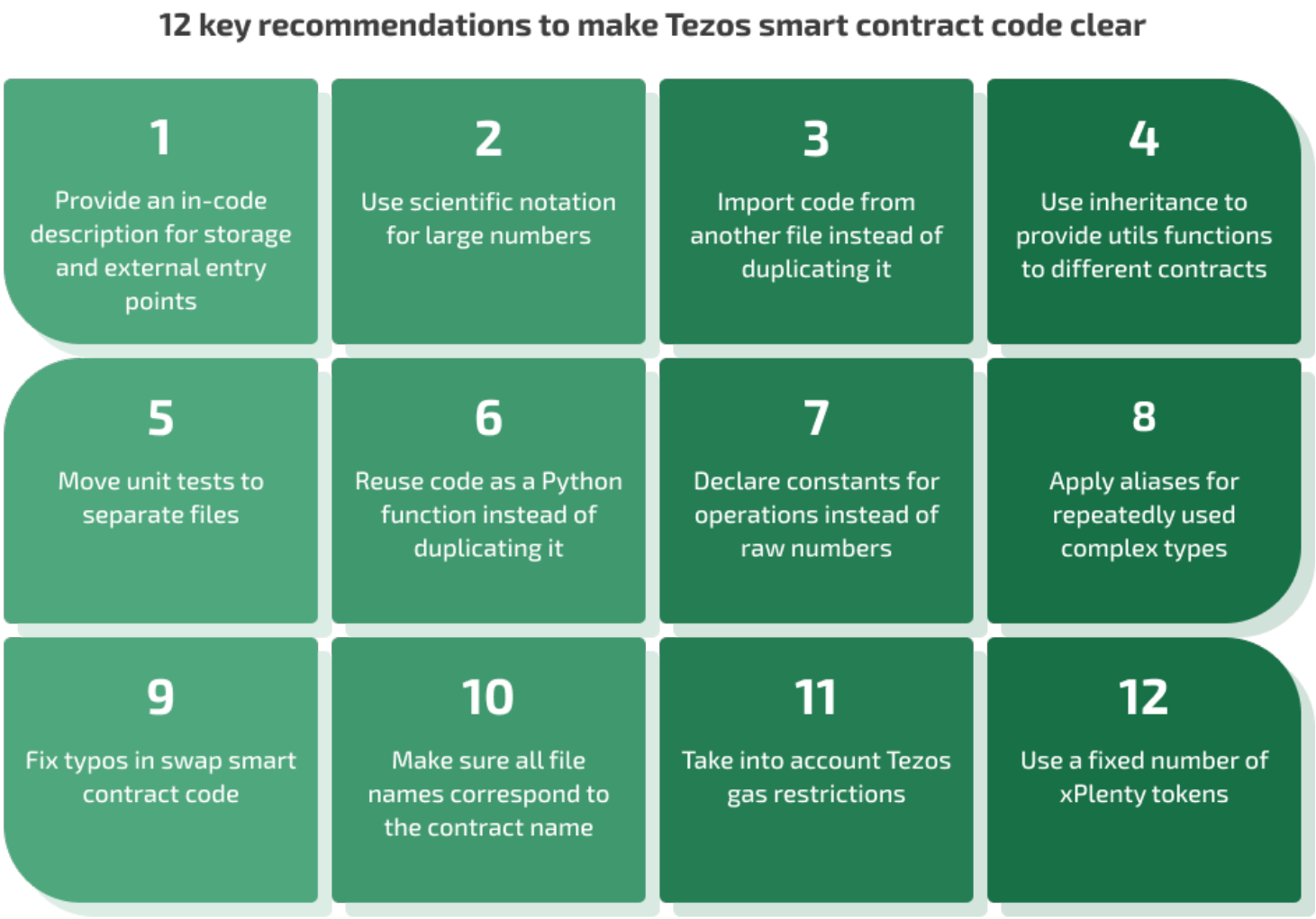
Our team put effort into establishing clear and efficient communication with our client by sending them timely updates so they could fix the most critical vulnerabilities right off the bat. This was especially vital for high-risk vulnerabilities, since fixing them could have a large impact on the entire solution architecture and therefore require time and effort.

By the end of the security audit, our client managed to fix the most critical high- and medium-risk vulnerabilities and significantly improved the security posture of their smart contracts.

Smart contract code review

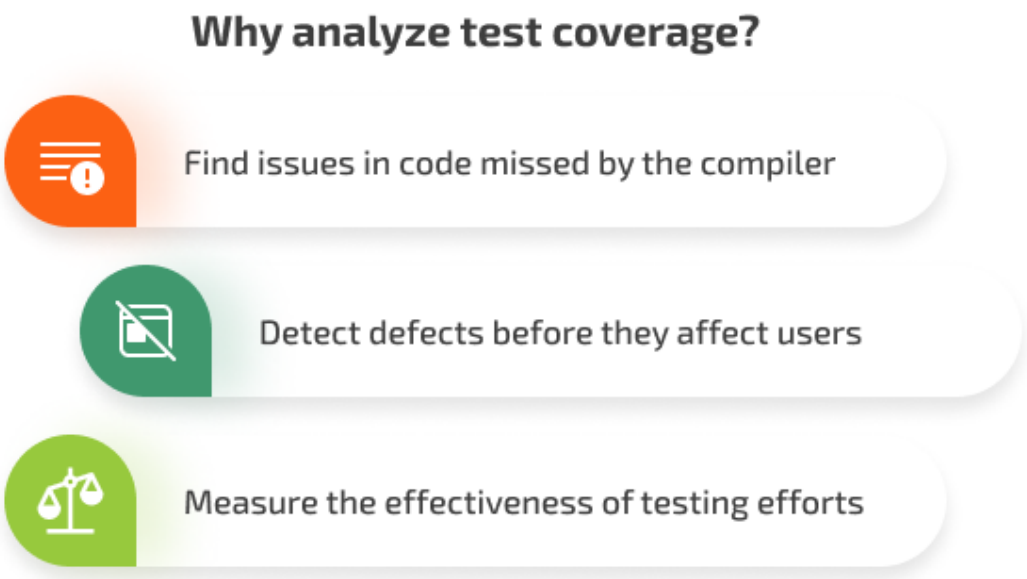
Reviewing smart contract source code is essential, since it helps you avoid coding antipatterns, make code understandable for all team members, and eliminate potential bugs. And since it's impossible to make changes to a deployed smart contract, we have to make sure that contracts are secure before deployment.

During the smart contract security assessment, we compared contract code against our list of best practices. By doing so, we identified potential weak spots in smart contract code and formed 12 key recommendations for our client:



Test coverage analysis

Unit tests help developers find issues in code missed by the compiler before deploying the contract to the blockchain. This is why checking unit test coverage was a crucial part of our audit.



During this stage, our team evaluated the percentage of unit test coverage and highlighted uncovered test cases for each of the contracts. Once our client fixed critical cases, we checked test coverage again and saw a significant improvement with two out of three smart contracts.

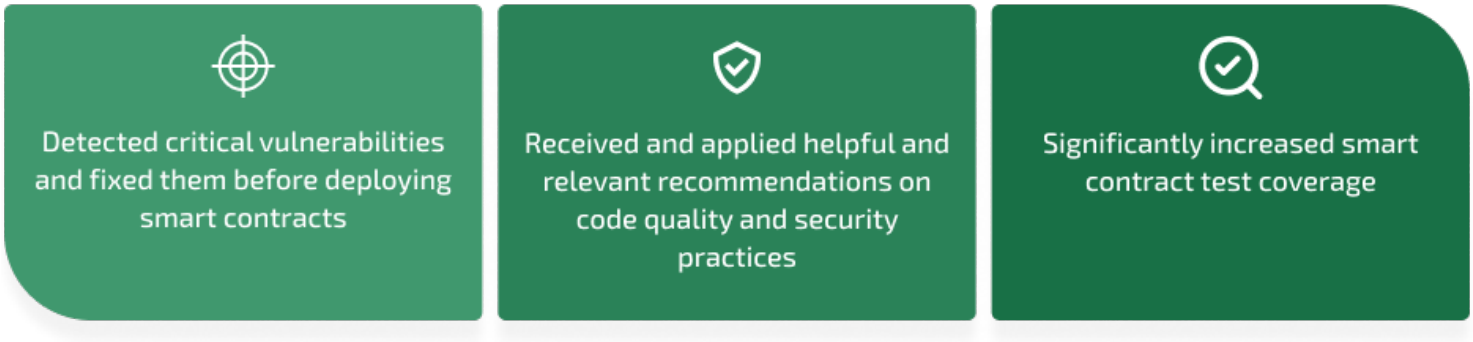
Thanks to Apriorit's work, our client managed to significantly increase test coverage of their smart contracts. For some contracts, they improved test coverage by 20% to 30%.

Increasing test coverage is essential for both measuring the effectiveness of testing efforts and finding issues before they affect users.

The impact

Security audits are essential for decentralized finance products since they allow you to identify vulnerabilities and security loopholes. The swap and DeFi smart contracts audit we performed for our client helped them make certain their project was secure and reliable.

The client's key accomplishments



After receiving comprehensive reports, the client **continues using our recommendations** related to code quality and other security practices. Also, they are planning to address the rest of the open issues mentioned in reports to improve the security posture of their smart contracts.

Overall, Plenty was pleased to know that no vulnerabilities were found during the assessment that could result in a loss of funds and that all of the identified medium- and low-risk vulnerabilities can be fixed. And since they feel certain about their smart contract security, our client can work on further improvements and adding features to current functionality. Not to mention that an **unbiased security audit** can help win users' trust.

Ready to check your product's security? Contact Apriorit to receive a comprehensive security evaluation along with relevant recommendations for improvements.