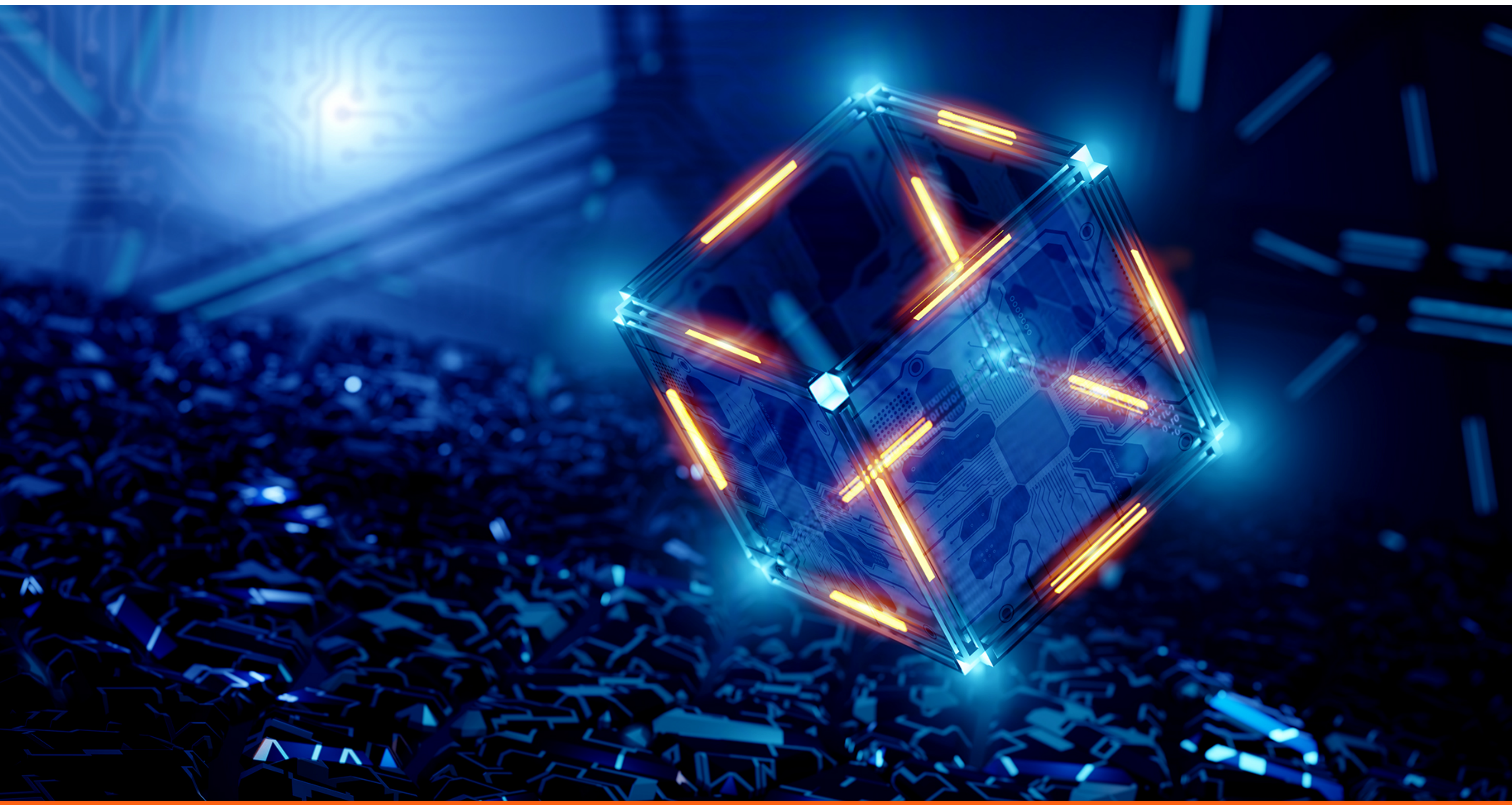
info@apriorit.com
www.apriorit.com



Case study

Security audit of blockchain wallet

Cryptonomic, a US-based company providing tools and services for blockchain-based financial products, approached Apriorit with a request to audit their Tezos blockchain solutions. It's vital to find and fix critical security issues in blockchain-based financial products before their release. Otherwise, the creators of such products risk losing not only time and money but also their reputations and customer trust.

Our team successfully audited a Tezos wallet and a decentralized application (dApp) that consisted of smart contracts, a bot application, and a frontend library. After the audit, we provided the client with a detailed report on discovered vulnerabilities and easy-to-implement recommendations on how to fix them.

We also offered several non-security-related recommendations aimed at improving the performance of the client's dApp and reducing the cost of smart contract operations.

The customer

Cryptonomic is a US-based company that creates tools, wallets, and smart contracts for blockchain-based financial solutions. They work with several blockchain networks, including Tezos.

The challenge

The client approached us with a request for auditing the security of a crypto wallet application that was recently updated and a new smart contract-based dApp for the Tezos blockchain.

As the wallet code had been updated, the client wanted to make sure that all critical data and financial operations the wallet works with remained secure and well-protected.

They also wanted to test their dApp through and through and make sure their smart contracts were flawless before releasing them to the mainnet.

The result

After a thorough security audit for the crypto wallet and dApp, the Apriorit team discovered several low- and medium-risk vulnerabilities in the wallet and smart contracts. We also found a logical error in the bot application of the client's dApp.

Thanks to timely updates from the Apriorit team, our client was able to eliminate most vulnerabilities shortly after they were discovered. Once we re-checked the fixed code, we provided the client with a detailed report and recommendations on how to address the remaining security issues.

“

The project was successful, helping increase customer confidence in the system's security. For a cost-effective fee, Apriorit worked efficiently and delivered excellent results. They communicated well, ensuring their partners felt like they were on the same page throughout the engagement. Read more »

CTO, Cryptonomic

(Extract from the independent review on Clutch.co)

Our approach

For this project, Apriorit formed a team of competent blockchain developers familiar with the technology stack requested by the client.

THE PROJECT'S MAIN TECHNOLOGIES

Blockchain platforms	Smart contract languages	Other technologies
● Tezos	● Solidity	● React.js
● Ethereum	● SmartPy	● Node.js
		● TypeScript

We also assigned an experienced project manager to this team to ensure efficient and timely cooperation between our specialists and the client.

After getting familiar with the project, we outlined key security risks for the crypto wallet and dApp to focus on during the audit:

EVALUATED RISKS

- User wallet compromise
- Loss of user finances
- Unexpected behavior of the wallet and dApp

For both the wallet and dApp, we outlined four main auditing activities:

1. Automated static analysis
2. Manual code review
3. Unit test coverage analysis
4. Testing for unexpected behavior

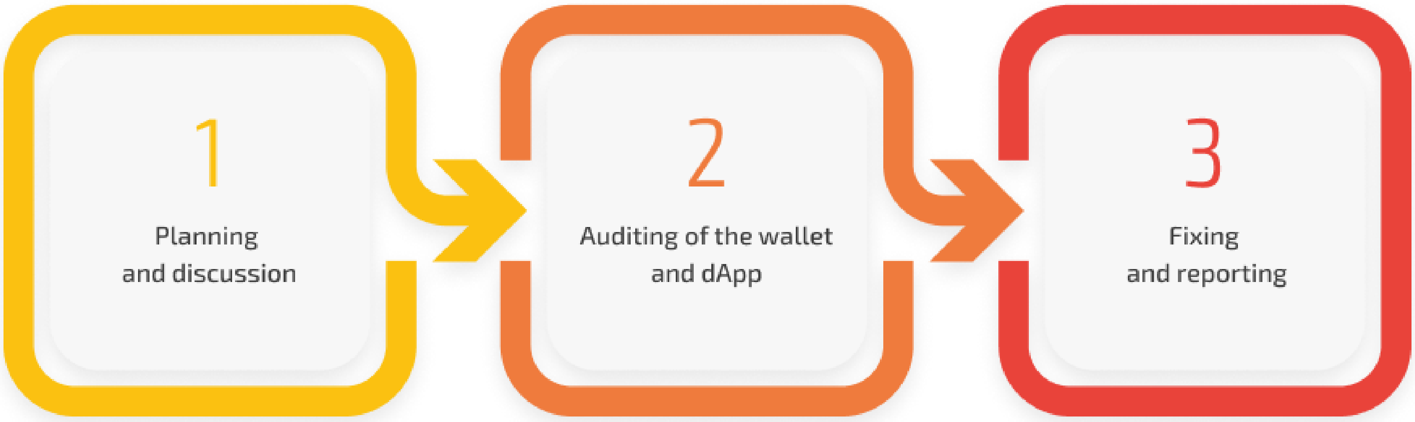
MAIN AUDITING ACTIVITIES

- Manual code review
- Static code analysis
- Unit test coverage analysis
- Testing for unexpected behavior

We also evaluated the overall quality of the source code for each of the audited products.

The entire process was split into three stages:

3 STAGES OF THE BLOCKCHAIN WALLET AND DAPP SECURITY AUDIT



Twice a week, we sent the client a short report with up-to-date information on all discovered security vulnerabilities and their statuses. This way, the client could make a timely decision on whether to fix a certain security issue or not and have us re-check the improved code if necessary.

Let's briefly go over each of these stages.

Stage 1. Planning and discussion

We started with analyzing the client's request and the project's specifics, determining the focal points for the audit, and selecting tools and techniques.

For this audit, we outlined several objectives:

- Evaluate the exposure of the wallet and dApp to known security vulnerabilities
- Determine potential attack vectors
- Check if any detected vulnerabilities can be exploited maliciously

Once the preliminary plan and audit schedule were prepared and approved by the customer, we moved to the main stage — running the actual audit.

Stage 2. Audit of the wallet and dApp

First, let’s take a look at some key points of the wallet audit.

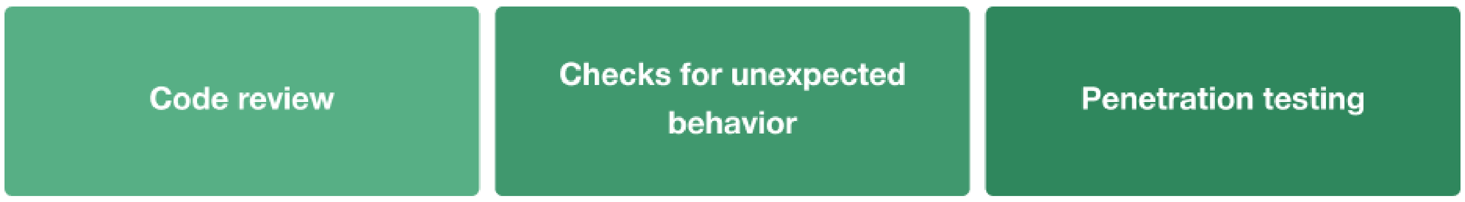
Auditing the blockchain wallet

The client had a desktop wallet application written in TypeScript. Users can use this app to see their balances, send new transactions, and review their operation histories. To connect to the Tezos blockchain network, this wallet uses a custom library.

We assessed the security of the wallet in three steps:

1. Performed a general code review, focusing on the analysis of private key exposure risks, smart contract interactions, and external dependencies
2. Checked the wallet for unexpected behavior
3. Executed manual attacks based on discovered vulnerabilities

KEY STEPS OF THE WALLET SECURITY AUDIT



We audited the wallet application using manual code review and automated static analysis. Once potential vulnerabilities were discovered, we ran manual attacks to check if these vulnerabilities could be exploited by malicious actors.

Auditing the dApp

The client's dApp is meant to perform cross-chain atomic swap cryptocurrency exchanges between two blockchain networks: Tezos and Ethereum.

The application has three main components:

- Ethereum and Tezos smart contracts containing the app's main logic
- A frontend library that provides an interface for initializing token exchange
- A bot application written in JavaScript that's responsible for finishing the exchange procedure

To ensure maximum security of the entire dApp, we assessed the security of each component separately:

KEY STEPS OF THE DAPP SECURITY AUDIT

SMART CONTRACT	FRONTEND LIBRARY	BOT APPLICATION
Manual code review and code quality evaluation	Manual code review and code quality evaluation	Manual code review and code quality evaluation
Behavioral analysis of smart contract source code		Static analysis
Penetration testing	Analysis of library interaction with blockchain	Analysis of interactions with other resources
Unit test coverage analysis	Private user data handling analysis	Filesystem and local resources usage analysis
Analysis in regards to the host network		Analysis of configurations and dependencies

As we did when auditing the cryptocurrency wallet, we tested the dApp using manual code review, automated static analysis, and manual attacks based on discovered vulnerabilities.

Our audit showed that the overall quality of the wallet and dApp was high and that the products were mostly secure. However, we did find some low- and medium-risk vulnerabilities in all components except for the frontend library.

SOME OF THE VULNERABILITIES DISCOVERED DURING THE AUDIT
and what can be done about them

WALLET	Unencrypted private key stored in-memory	Consider modifying the wallet to store private keys encrypted at all times and decrypt them only when necessary to sign data.
	MiTM attacks	Consider adding strict certificate pinning to avoid tampering with wallet traffic.
SMART CONTRACT	Implicit function visibility level	Function visibility should be explicitly defined to prevent confusion.
	Integer overflow	Consider using a safemath library for arithmetic operations to prevent the possibility of integer overflow.
BOT APPLICATION	Type-safe operations and code style	Consider using linters to preview and fix such problems.

Apriorit specialists also discovered an error in the refund logic that wouldn't cause any security issues but might lead to financial losses in case of high usage of the bot application.

Additionally, our blockchain specialists discovered that the bot application used an outdated version of the node-fetch dependency. The version used might have contained security vulnerabilities, as the update for it was marked as an important security release.

When auditing the frontend library, our specialists didn't find any vulnerabilities. However, we discovered other security risks related to the frontend project.

Fixing and reporting

The final stage of the project included three sub-stages:

- Preliminary reporting
- Reauditing the fixed code
- Issuing the final report

FINISHING THE AUDIT OF BLOCKCHAIN WALLET AND DAPP SECURITY

1 **Preliminary reporting**

2 **Reauditing the fixed code**

3 **Issuing the final report**

Once all planned auditing activities on the client’s wallet and dApp were finished, we prepared a preliminary report summarizing all information on the discovered vulnerabilities. In the report, we classified all of the vulnerabilities from low-risk to high-risk based on their impact and exploitability and offered suggestions on possible ways to mitigate them.

The client used our report to evaluate the overall state of their products and determine which security issues they wanted to address first.

After the client eliminated all critical vulnerabilities, we once more audited the reworked parts of the code and prepared a final report.

In the report, we outlined the key strengths of the audited product, all discovered vulnerabilities with current statuses (fixed or open), and our recommendations for addressing the security gaps. The report also includes detailed information on the methodologies and risk rating approaches used.

The impact

Regular audits are critical for maintaining a high level of security for sensitive data, especially for companies providing financial solutions. They give you a clear vision of all weaknesses in your product and help you choose the most efficient strategy for eliminating them.

By delegating auditing to Apriorit, Cryptonomic was able to get an objective evaluation of the security of their key products — a blockchain wallet and a dApp — from a team of trustworthy professionals.

It's especially important that their smart contract, which was the core of their dApp, was audited before its release to the mainnet. Smart contracts are nearly impossible to change after release. So if you find any code issues after the release, you have two options:

- Try to apply extremely complex development techniques (which may have severe negative effects on the smart contract itself)
- Spend extra resources on releasing a new smart contract

Thorough auditing and testing of a smart contract can prevent you from having to make such a choice.

Apriorit specialists also discovered a logical error in the client's bot application. Fixing this error will help our client reduce the cost of smart contract operations.

Finally, successfully passing a dApp and crypto wallet security audits was important for our client to build a positive brand image and gain the well-deserved trust of end customers.

Want to make sure that your blockchain-based solution contains zero security flaws? Task Apriorit's professionals with auditing the security of your blockchain wallets!