pixelplex Client-focused.
Result-driven

Web3 Hacks Analysis

Exploring the Biggest Hacks and Learning How to Stay Secure

Explore the intricacies of web3's most notable security breaches and equip yourself with proven strategies to navigate this digital frontier safely.

PixelPlex 2023

info@pixelplex.io pixelplex.io

Table of contents

Welcome to the world of web3

The evolution of the internet is marked by continuous innovation, and the most recent leap forward is the advent of **web3**.

The conventional web, known as web2, operates under centralized control, typically dominated by major corporations. Meanwhile, web3 represents a decentralized vision where users have control over their own data, identities, and transactions.

Mostly built on blockchain technology and underpinned by cryptographic principles, **web3 platforms promise a transparent, permissionless, and trustless digital realm**, where peer-to-peer interactions can be effectively executed without any intermediaries.

However, with great power comes great responsibility.

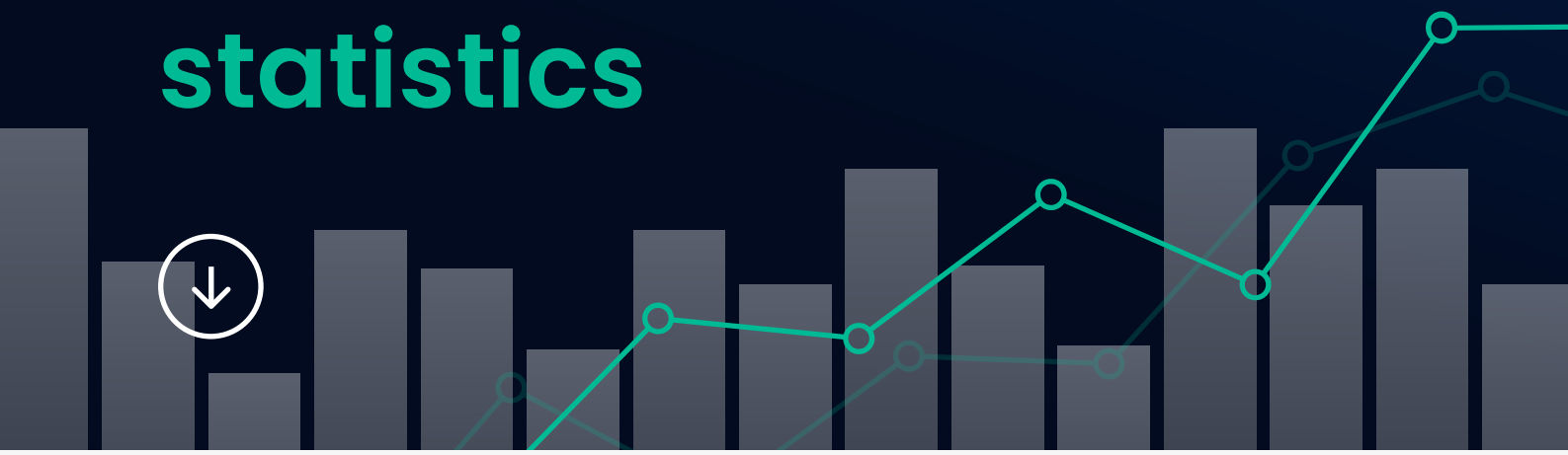
As more individuals and businesses transition to decentralized platforms, the stakes become higher. Recent times have witnessed a surge in sophisticated cyberattacks and breaches targeting web3 applications. From draining crypto wallets to capitalizing on vulnerabilities in smart contracts, malicious actors are tirelessly working to find and exploit weak points in the system.

It's crucial to recognize that in the decentralized world of web3, security is paramount.

The allure of greater control and freedom means that individuals bear a more significant burden of ensuring their assets and identities are protected. With considerable funds and personal information at risk, understanding the nuances of web3 security becomes not just a recommendation, but a necessity.

As we dive deeper into this **guide**, we'll explore the landmark hacks and provide you with strategies to ensure a secure digital journey.

Web3 market and hacks: statistics



$3.2B

The global web3 market size in 2021

Source: Emergen Research

$81.5B

The global web3 market size by 2030

Source: Emergen Research

$8B

Stolen by hackers and fraudsters from the web3 ecosystem in 2021

Source: Crypto losses in 2022

$3.7B

The amount the web3 ecosystem lost due to hacks only in 2022

Source: Crypto losses in 2022

$174.9M

The amount of loss in the web3 ecosystem due to fraud only in 2022

Source: Crypto losses in 2022

95.6% vs 4.4%

Percentage of losses: hacks compared to frauds, scams, and rug pulls respectively

Source: Crypto losses in 2022

Although there has been a decline in the overall value of cryptocurrency scams, one shouldn't rely solely on these numbers. The frequency of successful attacks has, in fact, increased compared to previous years.

Moreover, excluding the two largest attacks that happened in 2022 — **Ronin**, with $624 million lost, and **Wormhole**, with $326 million lost — the total funds stolen in 2023 might, during the same period, exceed those of the previous year.

Thus, attacks on the web3 ecosystem still remain a great threat. This is why both projects and users **need to adopt a comprehensive approach to security** to protect themselves from risks.

The most popular types of web3 hacks

Over the last few years, we've witnessed an evolving landscape of web3 hacking attacks.

Roughly 3-5 years ago, the primary focus of these attacks was **to breach wallets and exchanges and to exploit the vulnerabilities present in smart contract codes**. Back then, **Solidity**, the predominant language used for writing smart contracts, was still in its infancy. This immaturity provided hackers a ripe opportunity to manipulate smart contracts through unanticipated behaviors in the language, leading to unauthorized fund transfers.

However, as Solidity evolved and matured, many of these language-specific vulnerabilities were identified and fixed.

However, this didn't mark the end of the vulnerabilities. As the web3 ecosystem expanded, new projects began to interlink, creating new dependencies between them, which, in turn, led to a **new generation of security threats**.

The most popular types of web3 hacks

Type of hack	What happens	Examples	How to prevent them
Attacks due to human errors	Mistakes made by individuals leading to unauthorized access or data breaches	▪ Phishing attacks ▪ Misconfigured smart contracts ▪ Unauthorized access	▪ Regular training programs ▪ Regular security audits and reviews ▪ Restricting unnecessary access to sensitive data
Exploiting code vulnerabilities	Hackers exploit weaknesses in software to cause damage and steal funds	▪ Front-running attacks ▪ Smart contract code vulnerability ▪ Cross-site scripting	▪ Regular code audits ▪ Up-to-date patching ▪ Secure coding practices ▪ Input validation and sanitization
Rug pulls	Malicious developers gain trust, abandon a project, and run away with the invested funds	▪ Scam ICOs ▪ Pump and dump schemes ▪ Fake crypto token listings	▪ Due diligence before investment ▪ Avoiding new, unproven projects ▪ Using reputable platforms ▪ Monitoring community feedback and red flags
Flash loan manipulation	Exploitation of decentralized finance protocols to benefit from manipulated prices	▪ Exploitation of arbitrage opportunities ▪ Collateral swaps ▪ Recursive borrowing	▪ Robust smart contract design ▪ Periodic audits and monitoring ▪ Setting up transaction delays ▪ Monitoring abnormal transaction patterns
Social engineering	Tricking individuals into sharing confidential information or performing specific actions, often through deception or psychological tricks	▪ Phishing attempts ▪ Fake websites ▪ Fraudulent messages and posts on social media	▪ Enabling multi-factor authentication ▪ Educating yourself about the latest scams ▪ Always verifying the source and staying skeptical about unsolicited communications

It's worth noting that the web3 industry is dynamic in its nature, with new projects continually emerging. Conversely, this evolution also serves as a challenge. Newer protocols may inadvertently introduce logical vulnerabilities.

The web3 industry continually evolves, presenting solutions to combat hacking attacks. Similarly, malicious actors relentlessly try to detect vulnerabilities in new projects, creating an ongoing cycle of challenge and response.

Given the complexity of the ecosystem, predicting every possible user scenario or combination of input parameters becomes challenging, potentially leaving gaps that hackers can exploit.

Therefore, it's vital to prioritize security in blockchain-based systems. First and foremost, staying educated about web3 scams helps prevent them. Alongside this, it's crucial to regularly refine and update preventative strategies and measures.

Analysis of the 10 most popular web3 hacks

The complex infrastructure of web3 can be exploited in various ways by adept attackers. To better understand and mitigate these risks, it's essential to study past breaches. In this section, we break down **the most well-known web3 hacks**, explain how they occurred, and provide insights on preventing such incidents.

Yearn Finance

Attack type	Consequences	Prevention tips
Attacks due to human errors	A loss of nearly $11.6 million	▪ Ensure the accuracy of initialization values ▪ Implement regular audits and security risk assessments

Yearn Finance is a DeFi protocol based on the Ethereum blockchain. **On April 13, 2023**, it fell victim to an attack, leading to a staggering loss of approximately $11.6 million. This breach was a consequence of human oversight. The developer responsible for deploying the contract on the blockchain failed to double-check the correctness of initialization values.

How did it happen?

The incident occurred due to a misconfiguration of the yUSD token's smart contract. Since this vulnerability went undetected for a full 1000 days, it implies that the project did not take sufficient measures to ensure the security of the protocol.

The core logic of the smart contract was robust, but **the vulnerability was rooted in the constructor function**. This function runs just once during contract creation on blockchain, conducting necessary initialization operations. When forming the vulnerable contract, attributes like the token's name, symbol, and decimals are sequentially assigned. Then, addresses of tokens and contracts from other protocols are stored within the contract and are then used by the token contract throughout its entire lifecycle.

In this particular case, the vulnerability emerged during the recording of addresses. The address of the iUSDT token was supposed to be assigned to the "fulcrum" variable for proper logic operation, but it was mistakenly assigned to the iUSDC address.

The attacker exploited the contract by manipulating the reserve balance of the yUSDT token within the vulnerable yUSDT contract. This enabled the attacker to generate disproportionate profits in tokens from the same vulnerable yUSDT token contract by depositing USDT into it.

How to prevent similar hacks?

- **Ensure the accuracy of initialization values**

  The attack could have been prevented if the contract deployer had correctly written the iUSDT token address in the constructor.

- **Before launching a project, thoroughly audit its smart contracts. Plus, conduct regular audits and assessments of deployed contracts to identify and mitigate potential vulnerabilities**

  Technology as well as hacking techniques evolve over time. A secure contract can become vulnerable after a while, as hackers develop algorithms that threaten even the safest protocols.

Hundred Finance

Attack type	Consequences	Prevention tips
Flash loan manipulation	A theft of nearly $7.4 million	▪ Test the smart contract logic and conduct regular security audits ▪ Implement limits on loan amounts ▪ Address the issue with non-whole numbers

Hundred Finance, a fork of the Compound protocol, **faced a significant breach in its smart contract system** on April 15, 2023. This breach led to a theft of almost $7.4 million. The attacker skillfully exploited vulnerabilities within the smart contract's logic, enabling them to effectively manipulate the token's price and carry out transactions in their favor.

How did it happen?

The project had two types of wrapped Bitcoin tokens (WBTC), one of which was actively used, while the other one remained inactive. The attacker **took out a loan of 500 WBTC through the Aave protocol and deposited them into the Hundred Finance platform**, receiving their own wrapped Bitcoin token, hWBTC, in return.

Since Solidity and Ethereum, in general, don't support decimal numbers, any fractional parts arising from calculations are rounded down, which also granted the attacker an additional 2 wei.

The attacker transferred WBTC tokens into the WBTC/hWBTC pool, causing the price of the hWBTC token to surge significantly. **As a result, the attacker had a balance of 500 hWBTC and 2 wei of hWBTC**. Due to the artificially inflated price of hWBTC, the attacker only needed to send 1 wei to regain their WBTC, thus reclaiming their tokens from the hWBTC pool.

How to prevent similar hacks?

- **Assess the smart contract logic and conduct regular security audits**

Even if a fork occurs and the code is entirely borrowed from a popular and audited project, it doesn't exempt the need for auditing.

- **Implement borrowing limits**

By setting borrowing limits thoughtfully, considering transaction type, user history, and other factors, the system can reduce the level of vulnerability to sudden financial issues and malicious attacks.

- **Address the issue with non-whole numbers**

Using non-whole numbers in code can cause unforeseen problems stemming from rounding errors or precision issues. These vulnerabilities must be addressed through testing and preparation.

Ordinals Finance

Attack type	Consequences	Prevention tips
Rug pull	$1 million withdrawn by the developer	▪ Conduct thorough research ▪ Search for independent audit results ▪ Scrutinize project's social media presence and website

In April 2023, Ordinals Finance, which marketed itself as a decentralized lending and borrowing platform, executed a rug pull. The **project's developer** took advantage of excessive access rights attributed to a single account, **transferred all tokens from the contract, and ran away** with roughly $1 million.

How did it happen?

The project's contract had pre-existing centralized withdrawal functionality for all funds. While such functionality isn't uncommon and is sometimes a necessary measure — for example, to enhance fault tolerance — it's often abused for fraudulent activities like the rug pull. **This method can be implemented through safuToken**.

This function is only accessible to the contract owner and enables the withdrawal of all native ETH and ERC-20 tokens from the contract. **Therefore, the attack was facilitated by overly extensive access rights granted to a single account**.

Following this attack, all accounts across social media, the project's website, and other online platforms vanished, along with user funds.

How to prevent similar hacks?

- **Conduct thorough research**

  Understand the project's goals, mechanics, and technology. Be cautious of projects that promise unrealistic returns or lack transparent information.

- **Search for independent audit results**

  In a perfect scenario, a project should have undergone multiple audits. It's essential not just to consider the number of audits, but also to weigh the credibility and track record of the auditing firms.

- **Scrutinize the project's social media presence and website**

  Verify the authenticity of the individuals behind the project, ensuring they are not only real but also easily accessible, with valid contact details provided.

Euler Finance

Attack type	Consequences	Prevention tips
Flash loan manipulation	A theft of $195 million	▪ Undergo testing and audits by multiple firms ▪ Audit each new feature added to the platform

On March 13, 2023, Euler Finance, an **Ethereum-based lending protocol, fell victim to a flash loan attack**, during which the attacker successfully stole millions of tokens in DAI, USDC, StETH, and WBTC. This resulted in a loss of $195 million.

How did it happen?

The attack occurred **due to the vulnerability within the smart contract's logic**, specifically within the donateToReserves method. Originally absent from the protocol, this method was added later through an internal specification. Making sure this operation wouldn't trigger liquidation was crucial, but unfortunately, the donateToReserves method lacked this step, inadvertently enabling system manipulation.

A second contributing factor lies in the nature of the reward structure. The concept underlying the platform is that users can't transfer their debt tokens elsewhere, and their balance consists of proportions of both asset and debt tokens, which collectively determine the user's financial health. The problem was that **the reward system was dynamic** and depended on how much the borrower's financial health dipped below 1. In contrast, protocols like Compound and Aave employed a fixed percentage.

How to prevent similar hacks?

- **Subject the project to rigorous testing and get audits from different firms**

  This collaborative scrutiny helps identify and rectify vulnerabilities across various dimensions of the system, bolstering its overall security.

- **Audit each new feature added to the platform**

  This targeted evaluation ensures that any vulnerabilities are detected and resolved promptly, safeguarding the system from exploitation.

Merlin DEX 

Attack type	Consequences	Prevention tips
Rug pull	A theft of $1.82 million	▪ Undertake thorough testing, auditing, and issue resolution ▪ Strive for the maximum level of decentralization

On April 26, 2023, an attack occurred on the Merlin protocol, resulting in the theft of $1.82 million. The incident **took place during the sale of MAGE tokens**, which was an organized liquidity generation event.

How did it happen?

Merlin DEX is a zkSync-based decentralized exchange platform. **Before its launch, the company conducted a security audit from CertiK**, which identified a series of issues. According to the auditors' statement, most of these problems were addressed.

Initially, the **MerlinSwapFactory contract was created to facilitate the establishment of pools**. Each pool contained two types of tokens that constituted a trading pair. In this case, the USDC-WETH trading pair was generated. During the initialization of the trading pair contract, there was logic to approve all funds on the contract's balance to the FeeTo address. After the pool contract was triggered, the FeeTo address seamlessly withdrew all tokens from this pool.

There were multiple versions of explanations for why this occurred, but during the investigation, it was determined that a rug pull had taken place. **However, not the entire Merlin team was at fault — rather a group of developers within the team**.

One reason for the hack was Merlin not following the auditor's advice about too much centralization. Another reason was some harmful logic in Merlin's contracts, which the auditor didn't point out.

How to prevent similar hacks?

- **Undertake thorough testing, auditing, and issue resolution**

  Get security checks from several audit firms and address the issues they find in order to strengthen the security of the system.

- **Strive for the maximum level of decentralization**

  Avoid placing trust in a single wallet and utilize a multi-signature wallet for added security.

MEV Bots

Attack type	Consequences	Prevention tips
Exploiting code vulnerabilities	A loss of $25 million	▪ Adopt a decentralized mechanism for transaction ordering ▪ Implement measures that protect the integrity of the block contents

MEV bots were designed to identify opportunities for front-running. **In blockchain, when a transaction waits in a public pool to be added to a block, MEV bots can copy it**. They increase the transaction fee to get it added faster, making money for the bot users.

To prevent front-running opportunities, MEV bots employ MEV relays that delineate the roles of builders, proposers, and validators. Builders create blocks of ordered transactions and submit proposals, proposers accept the highest fee block, and validators approve and sign transactions.

Importantly, **proposers typically can't see the contents of a block** until they've committed to it by signing its header, a feature designed to mitigate front-running.

However, recently, an attack on Ethereum used a weakness in the way MEV relays work.

How did it happen?

The attack targeted a particular MEV bot harnessing a flaw within the Ethereum network. The anonymous user **executed a sandwich attack** by manipulating transactions to raise the price of tokens and make profits. The user ran a bot that managed to reap considerable benefits by preempting transactions awaiting confirmation on the platform.

The attack happened due to a vulnerability in the relay. Typically, the relay just sends a block's summary to the validator without showing the transactions inside. But due to the flaw, it sent the whole block, letting the validator see all transactions early.

The attacker first added 32 ETH to their wallet to become a validator. Then, leveraging the relay's vulnerability, the hacker gained access to the transactions contained in the formed block. After that, they constructed their own block by removing the sandwich attacks of MEV bots, and replacing them with their own. As a result, the attack caused a loss of over $25 million.

How to prevent similar hacks?

- **Adopt a decentralized mechanism for transaction ordering**

  Ensure that no single entity has undue influence over which transactions get processed first.

- **Implement measures to protect the integrity of block contents**

  These involve employing data encryption to safeguard transaction details, utilizing hash validation to ensure that block data remains unchanged during transmission and processing, and performing regular audits of the processes and systems involved in the validation of blocks.

Bitrue

Attack type	Consequences	Prevention tips
Attacks due to human errors	A loss of $23 million	▪ Prioritize cold wallet storage ▪ Implement additional security measures such as multi-factor authentication

On April 14, 2023, Bitrue experienced an exploit, resulting in a loss of approximately $23 million including ETH, QNT, GALA, SHIB, HOT, and MATIC.

How did it happen?

The attacker **gained unauthorized access and took control of the exchange's hot wallet**, enabling them to seize the funds.

CyversAlerts was among the first to report suspicious activity on Bitrue's Ethereum address. The platform detected a significant drop in the exchange's balance and alerted Bitrue via Twitter, providing detailed information about the suspicious transactions.

Bitrue confirmed that their Ethereum hot wallet had been compromised and shared details of the stolen assets in a series of tweets. The exchange **promptly addressed the exploit**, preventing further damage to its assets.

How to prevent similar hacks?

- **Prioritize cold wallet storage**

Centralized exchanges must emphasize the security of their hot wallets, which are inherently risky due to their constant online exposure. To minimize these risks, it is recommended to use cold wallets, which are offline and less susceptible to hacking.

- **Implement additional security measures, including multi-factor authentication**

It's essential for users to activate multi-factor authentication (MFA) on their accounts. Exchanges, for their part, should make MFA mandatory during account setup and important transactions to add an extra layer of security.

GDCA

Attack type	Consequences	Prevention tips
Attacks due to human errors	A loss of $13.9 million	▪ Prioritize cold wallet storage ▪ Consider decentralized exchanges

The South Korean cryptocurrency exchange and blockchain platform GDAC fell victim to a devastating hack, resulting in the theft of various cryptocurrencies totaling approximately $13.9 million.

How did it happen?

Similarly to the Bitrue exchange, the attacker gained unauthorized access to the exchange's **hot wallets**.

On April 10, 2023, GDAC's CEO, Han Sunghwan, announced that the attack occurred on the morning **of April 9, 2023**. During the incident, a hacker managed to gain control over a portion of the exchange's hot wallets.

How to prevent similar hacks?

- **Prioritize cold wallet storage**

Since cold wallets are offline, they are not susceptible to vulnerabilities that might exist in exchange software, wallet software, or the wider web infrastructure. Cold wallet owners have full control over their private keys, ensuring that third-party service compromises won't impact them.

- **Consider using decentralized exchanges**

Opting for decentralized exchanges can offer users a safer alternative. Unlike centralized exchanges, decentralized platforms don't have a central authority, which leads to better protection of user crypto assets.

BonqDAO

Attack type	Consequences	Prevention tips
Exploiting code vulnerabilities	A loss of nearly $120 million	▪ Implement a multi-faceted approach and cross-assessment ▪ Secure inter-protocol links ▪ Eliminate instant price updates

On February 1, 2023, BONq DAO experienced an Oracle attack, leading to a loss of nearly $120 million. In these attacks, external data sources, known as "Oracles", which smart contracts rely on to gather relevant information, are manipulated or tampered with.

How did it happen?

A user exploited a flaw in the **Oracle's logic** by meeting certain conditions that allowed them to directly manipulate token prices for their personal gain.

The main reason for the attack stemmed from an oversight in how BONq protocol set up its system alongside the Tellor Oracle. This setup instantly used the newest data from the TellorFlexOracle. Anyone was permitted to put up 10 TRB tokens and adjust a token's price in the Tellor Oracle. However, submitting inaccurate information posed the risk of asset loss.

The attacker put 10 TRB on TellorFlex and proceeded to input a false value for WALBT tokens. With this fake high price, the hacker borrowed a huge amount of $BEUR tokens and traded them for USDC. Then, they created a new account and deposited additional WALBT tokens into it.

In a parallel maneuver, from a different account, they again changed the price of WALBT, but this time made it remarkably low. Leveraging this devalued price, they sold WALBT tokens.

Through this sequence of actions, the attacker managed to abscond with **113.8 million WALBT** tokens and **98 million BEUR tokens**.

How to prevent similar hacks?

- **Implement a collaborative approach and cross-assessment**

As projects forge connections to build a unified ecosystem, this interconnectedness allows for new forms of hacks since a vulnerability in one protocol can be used to attack another. This is why it's vital for all parties involved, including data providers and protocols, to take action to counteract potential threats.

- **Enhance the security of inter-protocol links**

Weaknesses in inter-protocol links present prime opportunities for exploitation by hackers. Therefore, each side needs to engage in rigorous testing of their own systems, the counterpart's systems, and their interplay. It's crucial to check how actions from one side impact the project overall.

- **Eliminate instant price updates**

Immediate acceptance and integration of new data create avenues for malicious manipulation. By delaying the acceptance or by averaging data over time, the effects of sudden or manipulative inputs can be reduced, resulting in greater resilience against such threats.

SushiSwap

Attack type	Consequences	Prevention tips
Exploiting code vulnerabilities	A loss of about $3.3 million	▪ Implement access control ▪ Incorporate robust security measures for approval-dependent tasks

On April 9, 2023, SushiSwap, one of the key players in the DeFi space, suffered an attack due to a vulnerability in its router processor smart contract. The contract functioned for about two weeks before the attack took place, resulting in a loss of $3.3 million.

How did it happen?

The router contract had a method called **"processRoute"** that allowed for setting up a new liquidity pool. However, this pool had hidden malicious logic, and the router lacked a mechanism to validate the legitimacy of this pool contract.

This oversight was the first step in the attack, giving the hacker the **ability to launch a swap function** and set their malicious pool as the final one in the sequence.

Afterwards, the wrongdoer exploited the **router's swap feature** in their malicious pool, which allowed them to take user tokens and bypass access checks.

How to prevent similar hacks?

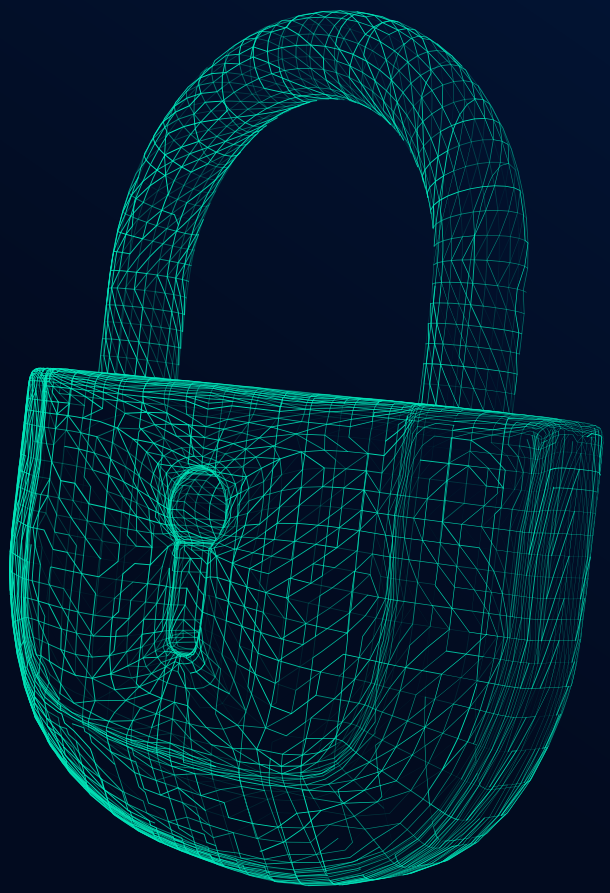
- **Implement access control mechanisms**

You must ensure that only the contract owner can perform critical transactions. Furthermore, essential conditions shouldn't be bypassed by any type of privilege escalations.

- **Incorporate robust security measures for approval-dependent tasks**

Ensure that all operations dependent on user approvals are thoroughly protected. You should also educate users on the most effective security practices. For example, encourage practices like revoking approvals when unnecessary. You may well consider promoting the use of one-time message signatures, known as "Permit", enabling transactions without continuous access to users' funds by the protocol.

Web3 security essentials: from builders to users



Equipped with a solid knowledge base and the most efficient strategies, it is possible to minimize and even prevent risks associated with web3 hacks.

Below, you'll find **essential guidelines and best strategies** for both users and creators that will reinforce your footing within the web3 landscape.

Best security practices for projects and businesses

Conduct regular security audits	■ Given that hacking techniques are always evolving, even highly secure contracts should be audited regularly ■ Even if the code is borrowed from a widely used and thoroughly vetted project, it should be audited ■ When a project releases a new contract version or makes any improvements, a full re-audit is essential, as even minor changes can affect the overall logic
Opt for audits by multiple auditing firms	■ Order audits not just from one but multiple respected auditing firms ■ Different auditors bring unique perspectives and expertise, ensuring increased security of your project
Utilize bug bounty programs	■ Bug bounty programs are excellent initiatives, allowing ethical hackers to find potential issues and vulnerabilities in a smart contract
Stick to proven security practices	■ Employ trusted libraries and frameworks, multi-factor authentication, and tight access control ■ Proactively adopt necessary security measures. For example, setting limits on loan amounts can curb the risks associated with flash loan attacks ■ Prioritize prevention measures for users
Prioritize extensive testing	■ Perform extensive testing of your project and use advanced security verification tools, such as formal verification, static code analysis, and smart contract testing frameworks ■ Enhance the efficiency of your monitoring system to enable the quick detection of suspicious activity
Strive for complete decentralization	■ Strive for maximum decentralization and avoid excessive reliance on a single wallet

Best security practices for users

Do your own research

- Carefully analyze the project before investing in it Verify the project's authenticity through legitimate social media profiles and accurate contact information
- Evaluate the project's reputation, recognition, and audit outcomes

Ensure there are robust security measures in place

- Use reputable wallets, preferably cold wallets, which are offline and less susceptible to hacking
- Enable multi-factor authentication to fortify your account security
- Store your private keys and seed phrases offline
- Use security tools like Web3 Antivirus to promptly detect vulnerabilities and prevent phishing attempts
- Regularly update your software and wallets

Stay updated on the latest threats

- Stay informed about web3 technologies, recent hacks, and security incidents to effectively identify vulnerabilities
- Exercise caution when encountering suspicious links and potential phishing attempts

Take proactive steps to safeguard your assets

- Diversify your portfolio to minimize the risk of losing all assets in case of a protocol attack
- Grant approval for a limited amount of tokens rather than the entire balance

Conclusion

Web3, with its vast opportunities, presents a target for both creators and scammers. The hacks we've delved into in this guide underscores that even minor errors can lead to substantial losses and damage a project's reputation. Therefore, the **responsibility lies with both** projects and users to prioritize safety.

Projects must treat **audits** as indispensable, preferably engaging diverse and reputable firms. It's also important to regularly conduct audits, given that hackers continually devise new tactics. Therefore, staying one step ahead becomes imperative.

For users, employing safety measures like **multi-factor authentication and advanced web3 security tools** is highly advisable. It's also paramount to stay abreast of the latest trends and threats in the web3 realm. In this ever-changing environment, continuous learning and a proactive approach will serve as the strongest safeguards for your assets.

Stay alert, stay informed, and stay safe.

About PixelPlex

PixelPlex is a custom software development company with 16 years of experience and solid expertise in **blockchain, machine learning, big data, and security**.

PixelPlex offers end-to-end solutions, spanning from R&D to development and security, catering to businesses across multiple domains, including FinTech and banking, real estate, eCommerce, and supply chain.

PixelPlex has **150+** in-house experts and boasts **450+** successful projects delivered for startups, middle-size businesses, and enterprises. Having a strong focus on blockchain development, the PixelPlex team has delivered **80+** blockchain-based projects, including **2** unicorns.

Contact us

Thank you for reading!

Want to get more insights?

Visit our website:

pixelplex.io

or contact us via email:

info@pixelplex.io

Anastasia Kapura

Business Development Manager