CERTIK

Gem Wallet

Security Assessment

CertiK Assessed on Apr 8th, 2026

CertiK Assessed on Apr 8th, 2026

Gem Wallet

The security assessment was prepared by CertiK.

Executive Summary

TYPES	PLATFORM	METHODS
Wallet	IOS | android	Dynamic Testing, Manual Review, Testnet Deployment

LANGUAGE	TIMELINE
Kotlin, Swift	Preliminary comments published on 11/15/2025 Final report published on 04/08/2026

Vulnerability Summary

11	7	0	4	0
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed immediately. Users should be cautious when interacting with any application with outstanding critical risks.
0	High		High risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds, thief of user data, and/or loss control of the application.
6	Medium	6 Resolved	Medium risks may not pose a security risk at a large scale, but they can affect the overall functioning of a platform or be used to target a certain group of users.
5	Low	1 Resolved, 4 Acknowledged	Low risks can be any of the above, but on a smaller impact. They generally do not compromise the overall integrity of the project.
0	Informational		Informational errors are often recommendations to improve the configuration or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the application.

TABLE OF CONTENTS | GEM WALLET

APPROACH & METHODS | GEM WALLET

This report has been prepared for Gem Wallet to discover issues and vulnerabilities in the application of the Gem Wallet project. Gem Crypto Wallet is a self-custodial, open-source mobile cryptocurrency wallet that supports asset management, purchase, and on-chain swapping

The pentest was a manual assessment of the security of the application's functionality, business logic, and vulnerabilities, such as those cataloged in the OWASP Top 10. The assessment also included a review of security controls and requirements listed in the OWASP Application Security Verification Standard (ASVS). The pentesters leveraged tools to facilitate their work. However, the majority of the assessment involved manual analysis.

The main objective of the engagement is to test the overall resiliency of the application to various real-world attacks against the application's controls and functions and thereby be able to identify its weaknesses and provide recommendations to fix and improve its overall security posture.

Two members of the CertiK team were involved in completing the engagement, which took place over the course of 5 days in November 2025 and yielded 11 security-relevant findings. The most significant vulnerabilities are related to the information on the wallet signing interface lacking proper validation and warnings.

Other weaknesses were also found and are detailed in the Findings section of the report. We recommend addressing these findings to ensure a high level of security standards and industry practices and to raise the security posture of the application.

REVIEW NOTES | GEM WALLET

In-scope feature

The following features fall within the scope of this code review. The focus is on the core security logic of the wallet, especially the components responsible for key management and transaction signing.

- Key generation and storage: Secure Random Number Generation, WalletCore integration, and platform keystore usage
- Transaction signing flow: Domain separation, deterministic signing, and data isolation
- Seed import and export handling: Mnemonic validation, encryption and decryption, and clipboard exposure
- Derivation and address generation: BIP paths and cross-chain handling
- Error handling in key and signing operations: Failure recovery, handling of corrupted data, and coverage of edge cases

Wallet Secret Handling

Key creation and Storage

IOS: The wallet creation process begins with WalletCore's `HDWallet()` generating a random mnemonic from 128-bit entropy. After the user successfully confirms the mnemonic, the system automatically generates a 32-byte random password using `SecRandomCopyBytes()` and stores it in iOS Keychain. Finally, the mnemonic is imported into persistent storage via `WalletCore.KeyStore.import()`, which encrypts the mnemonic using the generated password.

Android: The android application creates mnemonic via trust wallet's `HDWallet(128, "").mnemonic()` function. After the user successfully confirms the mnemonic, it uses `SecureRandom()` to generate a 32-byte random password, encrypts it with AES256 using EncryptedSharedPreferences, where the MasterKey used for this encryption is held in the Android Keystore. Finally, it uses walletCore's `StoredKey.importHDWallet()` to import the wallet with the password and stores the encrypted material.

Export Mnemonic

IOS: In the secret phrase export function, the wallet is allowed to recover the mnemonic via `getMnemonic()`. The wallet first retrieves the stored master password from the Keychain, which is protected by the device's access control. After biometric authentication succeeds (if enabled), the application uses the retrieved password to decrypt and recover the `HDWallet` object, from which it extracts the mnemonic.

Android: The Android wallet first triggers biometric authentication (if enabled), via `onAuthRequest()`. Then it calls `handleShowPhrase()` to process mnemonic export. It retrieves password from Android EncryptedSharedPreferences, then uses walletCore's `storeKey.decryptMnemonic()` to recover mnemonic using the password.

FINDINGS | GEM WALLET

11	0	0	6	5	0
Total Findings	Critical	High	Medium	Low	Informational

This report has been prepared for Gem Wallet to identify potential vulnerabilities and security issues within the reviewed codebase. During the course of the security assessment, a total of 11 issues were identified. Leveraging a combination of Dynamic Testing, Manual Review & Testnet Deployment the following findings were uncovered:

ID	Title	Category	Severity	Status
GEW-05	Lack Of "Attestation" Origin Verification During WalletConnect Connection	Security Misconfiguration	Medium	Resolved
GEW-06	EIP-712 Chain ID Mismatch	Security Misconfiguration	Medium	Resolved
GEW-07	Deprecated `Eth_Sign` Still Supported	Security Misconfiguration	Medium	Resolved
GEW-08	EIP-4361 SIWE Validation Fail - Origin And ChainID Mismatch	Security Misconfiguration	Medium	Resolved
GEW-09	Insufficient Display Validation In Permit / Permit2 Signature Requests	Security Misconfiguration	Medium	Resolved
GEW-10	Missing Detection Of Unknown Spenders And Allowance Of Unlimited Token Approval	Security Misconfiguration	Medium	Resolved
GEW-02	Lack Of SSL Pinning	Security Misconfiguration	Low	Acknowledged
GEW-03	Custom Keyboards Allowed For Sensitive Inputs	Information Disclosure	Low	Acknowledged
GEW-04	Insecure Keystore Configuration	Security Misconfiguration	Low	Acknowledged
GEW-11	Missing Detection For Request Flooding	Security Misconfiguration	Low	Acknowledged
GEW-12	Mnemonic Display Allows Screenshot	Information Disclosure	Low	Resolved

GEW-05 Lack Of "Attestation" Origin Verification During WalletConnect Connection

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

The WalletConnect protocol recently launched a Verify API, which is used to confirm whether the connected origin matches the registered origin on the WalletConnect site. This verification occurs during the connection confirmation in the user's wallet. However, the wallet failed to ensure that the returned origin from the "Attestation" API matches the original origin from the QR code.

Impact

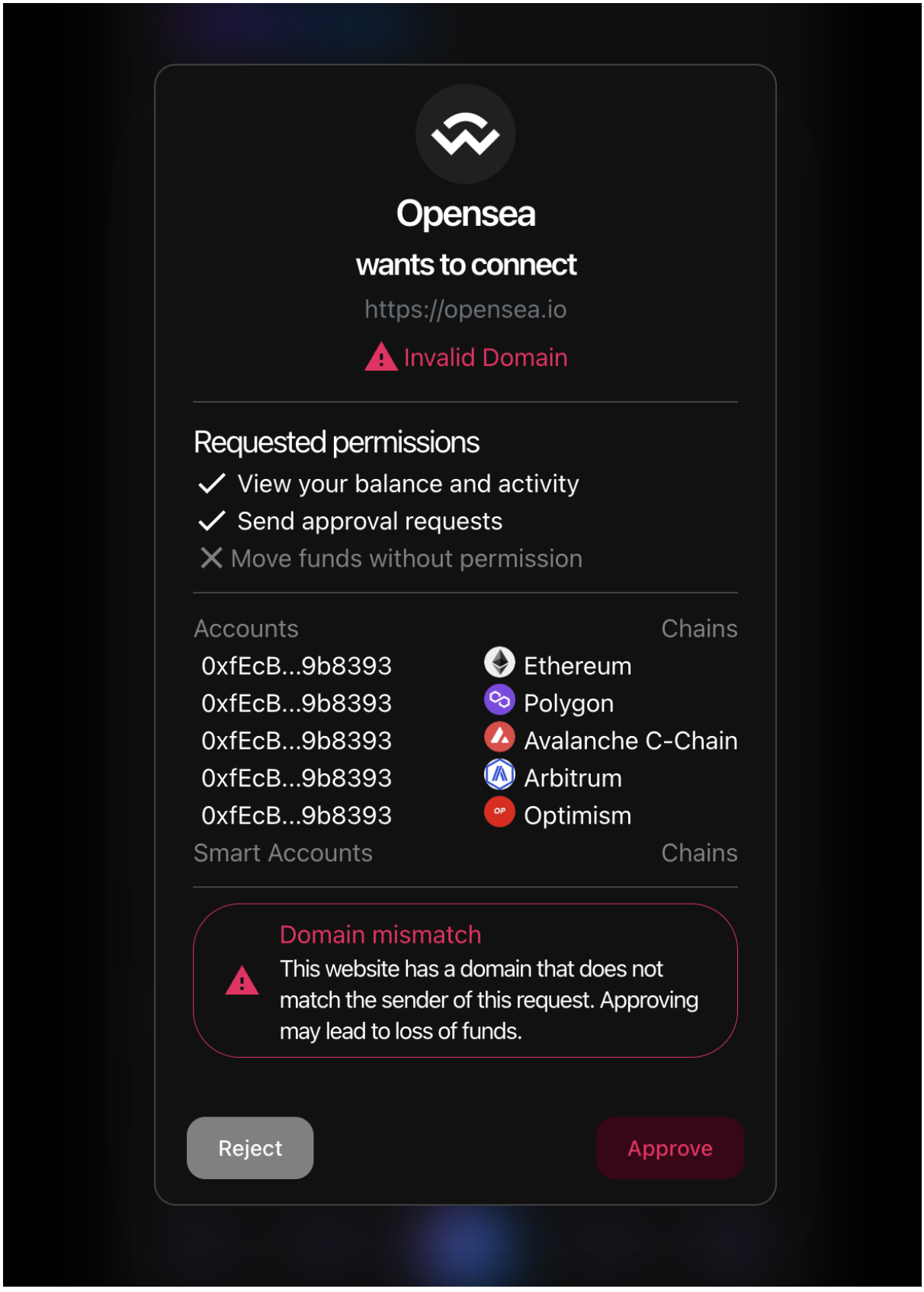
The absence of verification might increase the likelihood of wallet users falling victim to targeted phishing attacks when using WalletConnect to connect with a malicious dApp.

Proof of Concept

To demonstrate the potential impact, we deployed a phishing application at https://wc.mybit.fun/. This web application attempts to mimic the Opensea site. Please follow the steps below to replicate the issue.

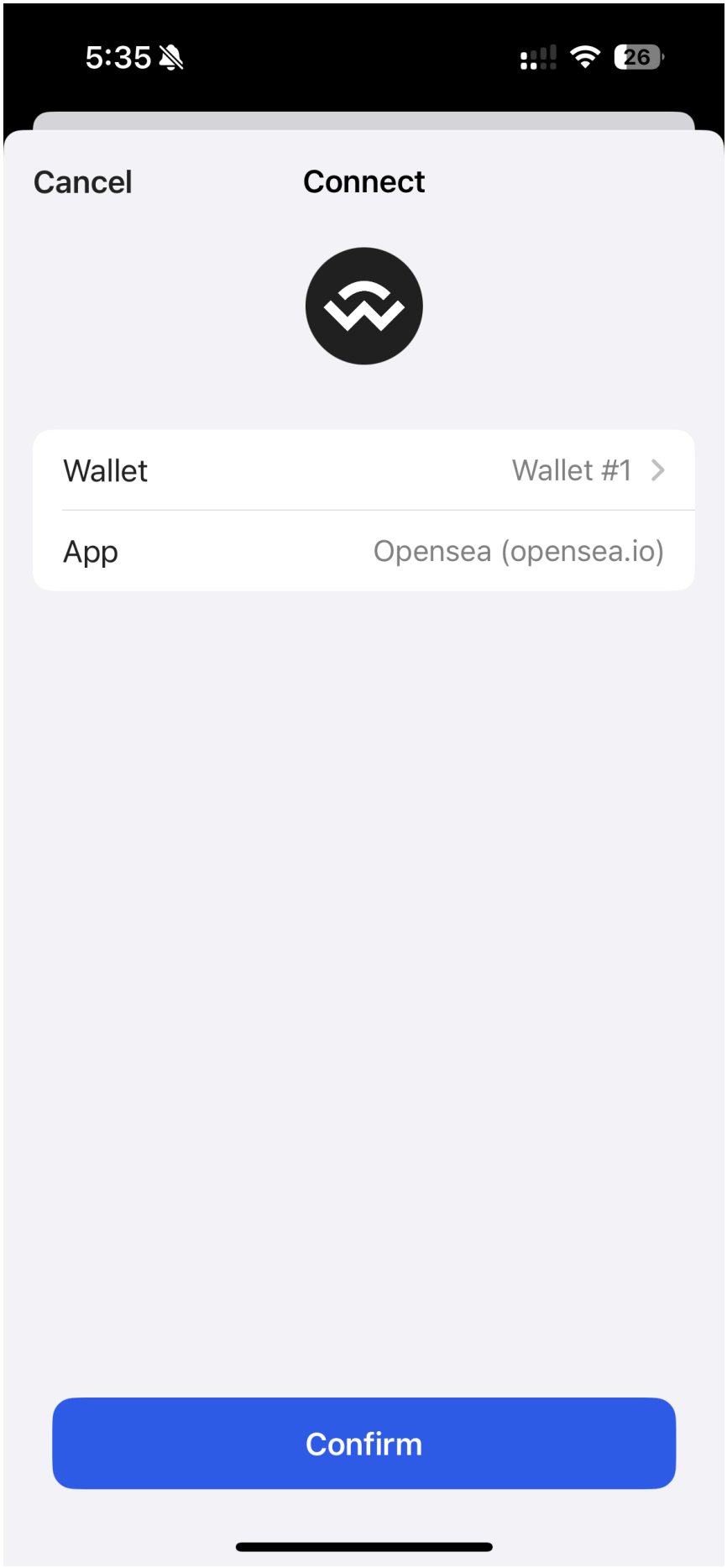
1. Click on ‘Connect Wallet’ and select WalletConnect
2. Scan the QR or copy the wc link.

The wallet shows a connection with the application as though it were OpenSea. If the wallet conducted appropriate validation of the origin, similar to the demo provided by WalletConnect (https://react-wallet.walletconnect.com/walletconnect), an error message like the following should be displayed.

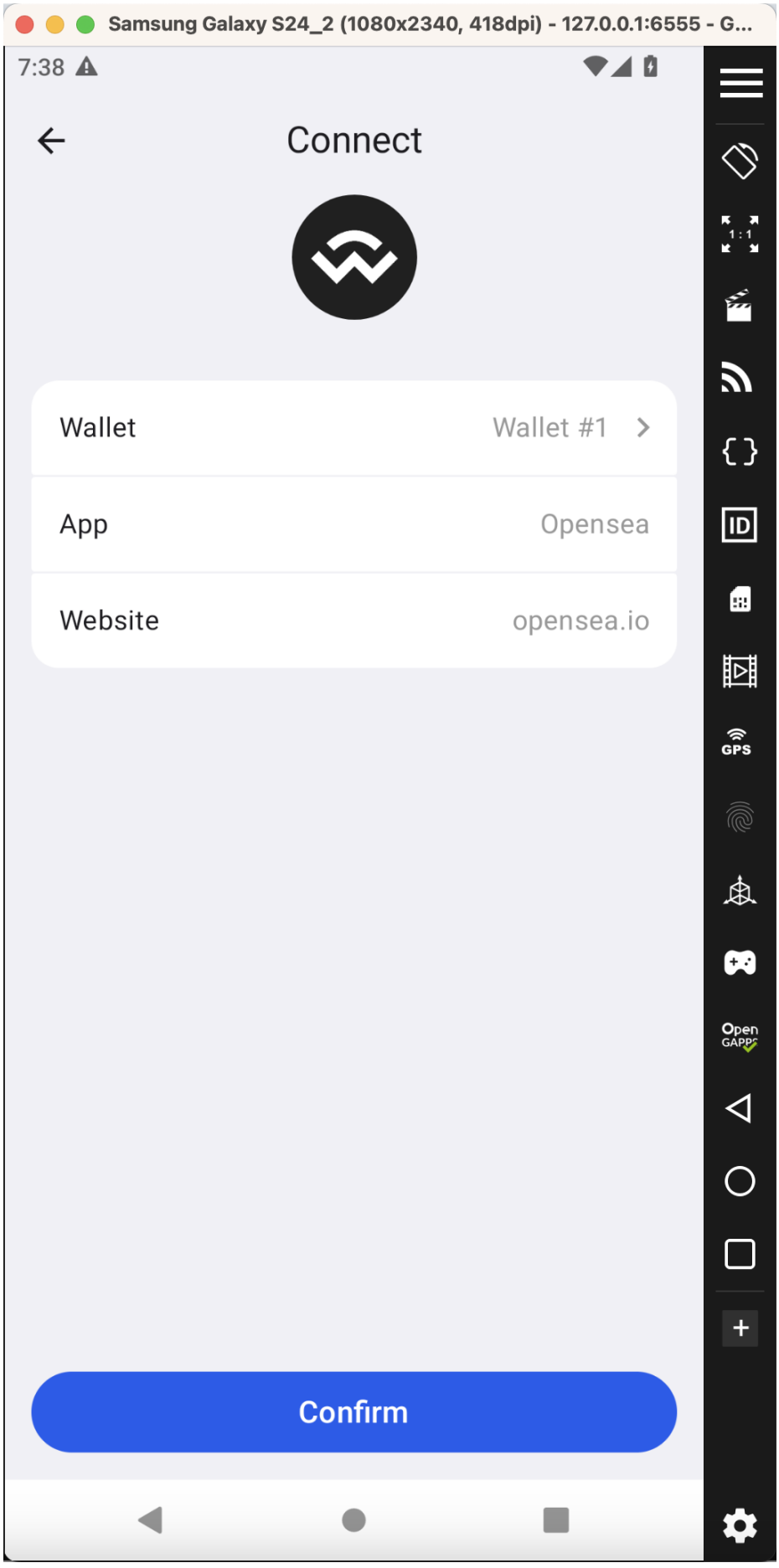


However, the Walletconnect UI shows no such warning.

IOS app:


5:35
26
Cancel
Connect
Wallet
Wallet #1
App
Opensea (opensea.io)
Confirm

Android app:


Samsung Galaxy S24_2 (1080x2340, 418dpi) - 127.0.0.1:6555 - G...
7:38
Connect
Wallet
Wallet #1
App
Opensea
Website
opensea.io
Confirm

Recommendation

It is recommended that the wallet verifies the returned "origin" value of the "attestation" request against the website the user is attempting to connect to. For more information regarding the Verify API, please refer to:

- https://medium.com/walletconnect/unlocking-the-power-of-verify-api-a-step-by-step-guide-for-wallets-4e939a273d9a
- https://docs.walletconnect.network/wallet-sdk/web/verify

Alleviation

[Gem Wallet, 04/03/2026]: The team heeded the advice and resolved the issue by displaying a warning message stating "This connection comes from an untrusted source" and blocking the connection in iOS version 1.3.355 and Android version 2.0.10.

GEW-06 | EIP-712 Chain ID Mismatch

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

During testing of the wallet's EIP-712 signature handling via WalletConnect, it was observed that the wallet signs typed structured data even when the chainId specified in the EIP-712 domain object does not match the chain of the active session.

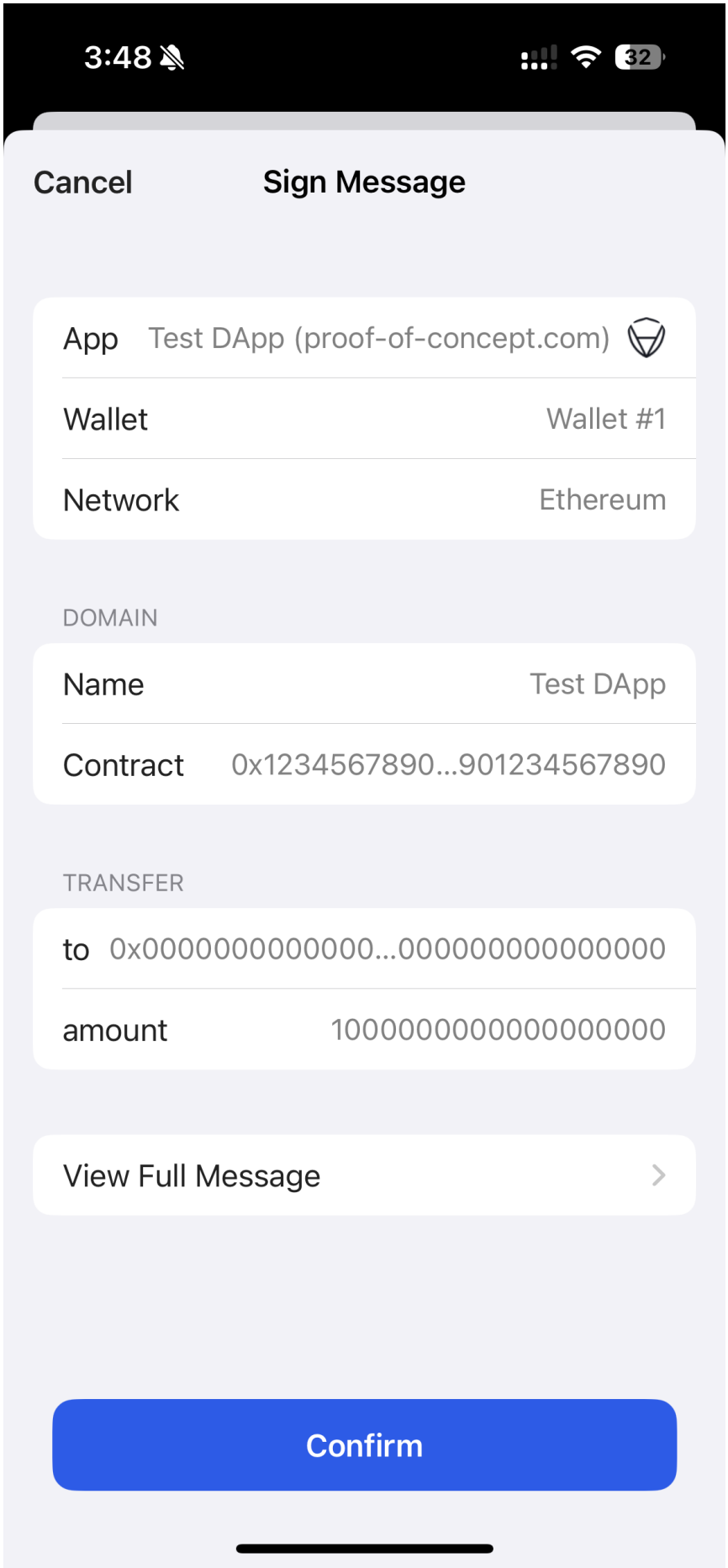
After establishing a WalletConnect session on Ethereum Mainnet (1), the test DApp sent an eth_signTypedData_v4 request containing a domain.chainId value of Polygon (137). The wallet presented the signature prompt and allowed the user to confirm the signing request without any warning or rejection.

Impact

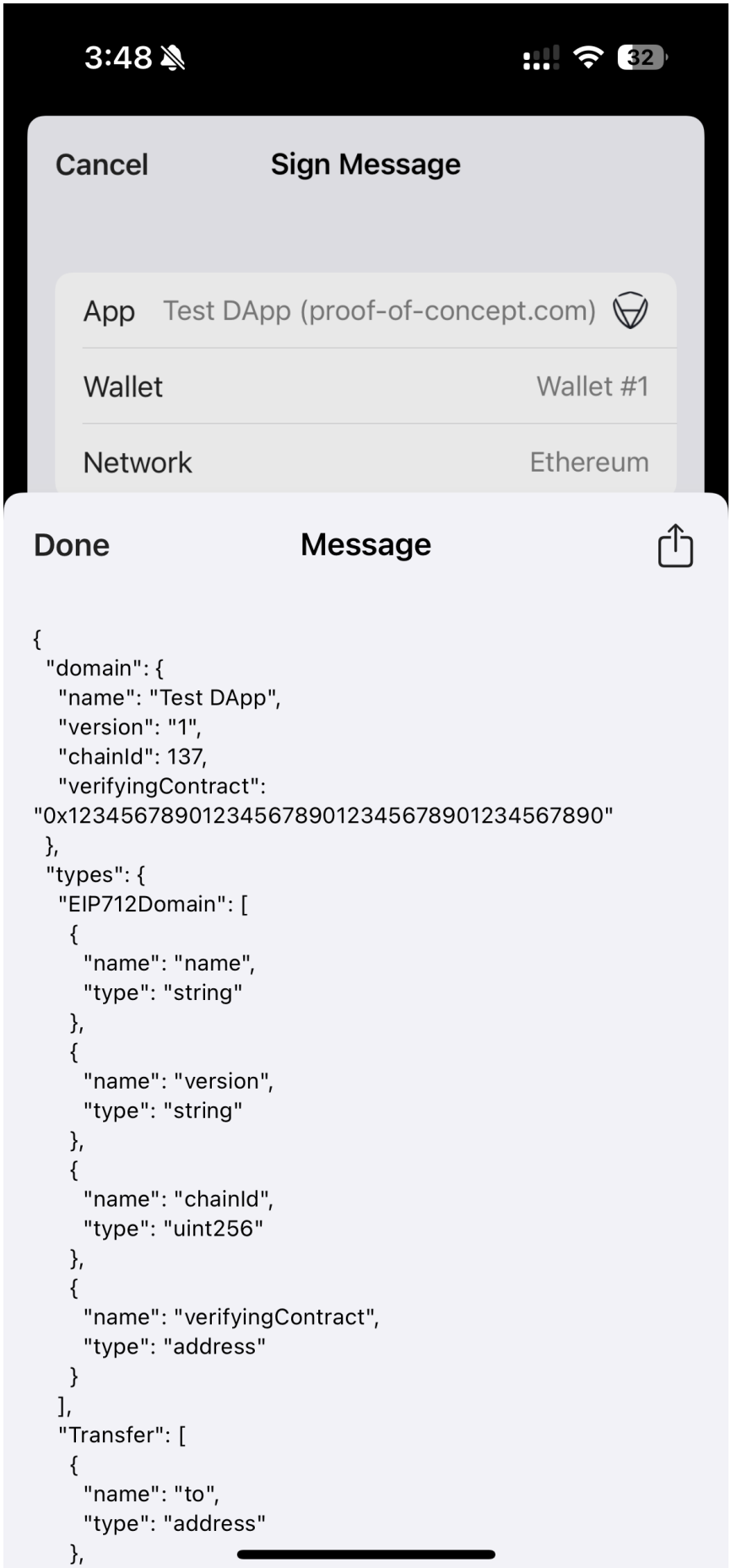
An attacker controlling a malicious DApp could exploit this behavior to trick users into signing messages intended for a different network, which may later be replayed or misused in cross-chain contexts. The lack of clear user feedback creates a phishing and replay-attack vector where users may unknowingly sign data representing transactions or approvals on a different chain.

Proof of Concept

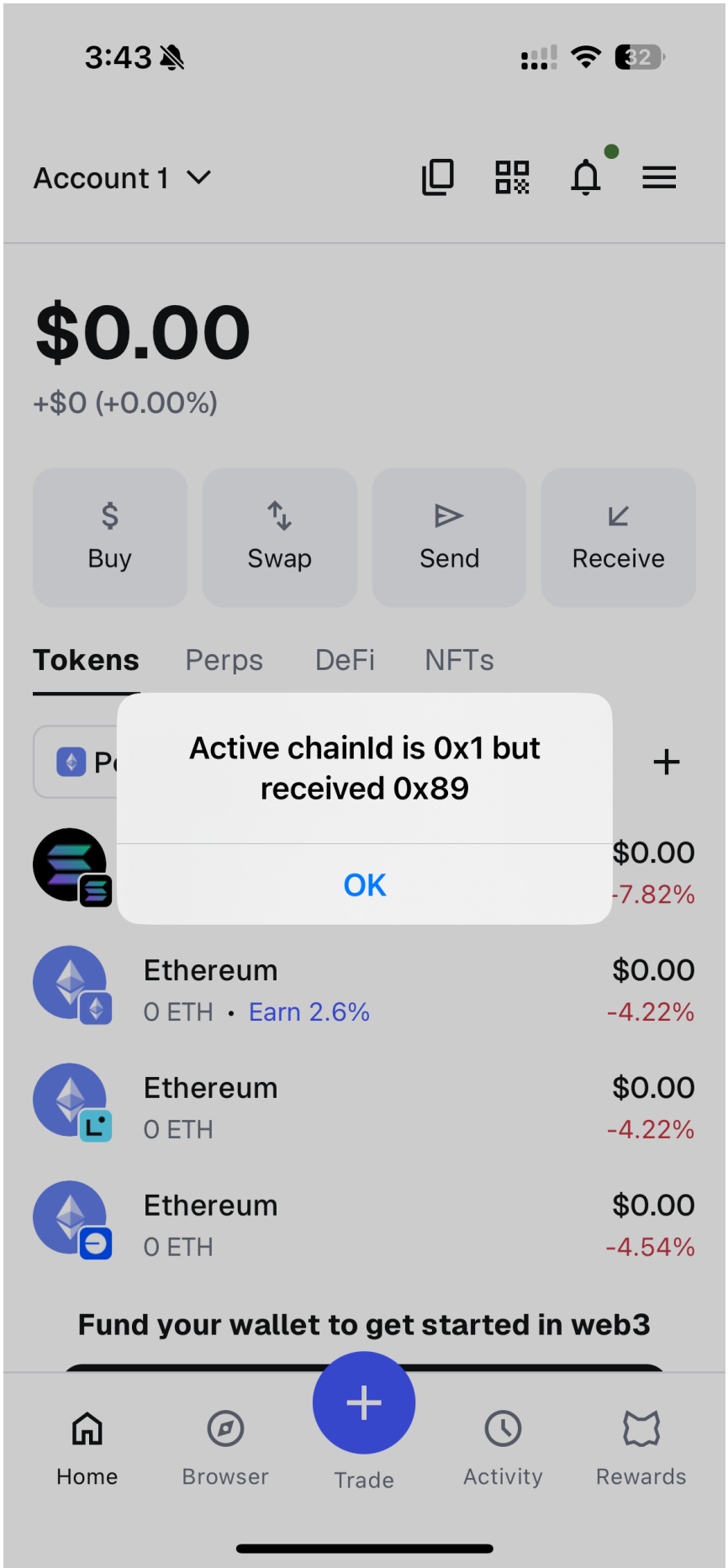
The screenshot below shows that a WalletConnect session was established on Ethereum Mainnet (chain id: 1), yet the DApp sent an EIP-712 message with domain.chainId set to 137 (Polygon) using the eth_signTypedData_v4 method, and the wallet still displayed the confirmation prompt and generated a valid signature without any warning.



The forged chain id can only be viewed via clicking full message.



As a reference, Metamask will generate an alert when a chain id mismatch is detected.


3:43
Account 1
$0.00
+$0 (+0.00%)
Buy
Swap
Send
Receive
Tokens
Perps
DeFi
NFTs
Active chainId is 0x1 but received 0x89
OK
$0.00
-7.82%
Ethereum
0 ETH • Earn 2.6%
$0.00
-4.22%
Ethereum
0 ETH
$0.00
-4.22%
Ethereum
0 ETH
$0.00
-4.54%
Fund your wallet to get started in web3
Home
Browser
Trade
Activity
Rewards

Recommendation

It's recommended that the wallet enforce strict EIP-712 chain ID validation before signing typed data. The wallet should reject or at least display a clear warning whenever the domain.chainId in the EIP-712 message differs from the active network's chain ID. It should also present the network name and chain information explicitly in the signing interface to help users understand which network the signature applies to.

Alleviation

[Gem Wallet, 04/03/2026]: The team heeded the advice and resolved the issue by rejecting invalid requests in iOS version 1.3.355 and displaying a message stating "chainId mismatch" while preventing the user from proceeding in Android version 2.0.10.

GEW-07 | Deprecated `Eth_Sign` Still Supported

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

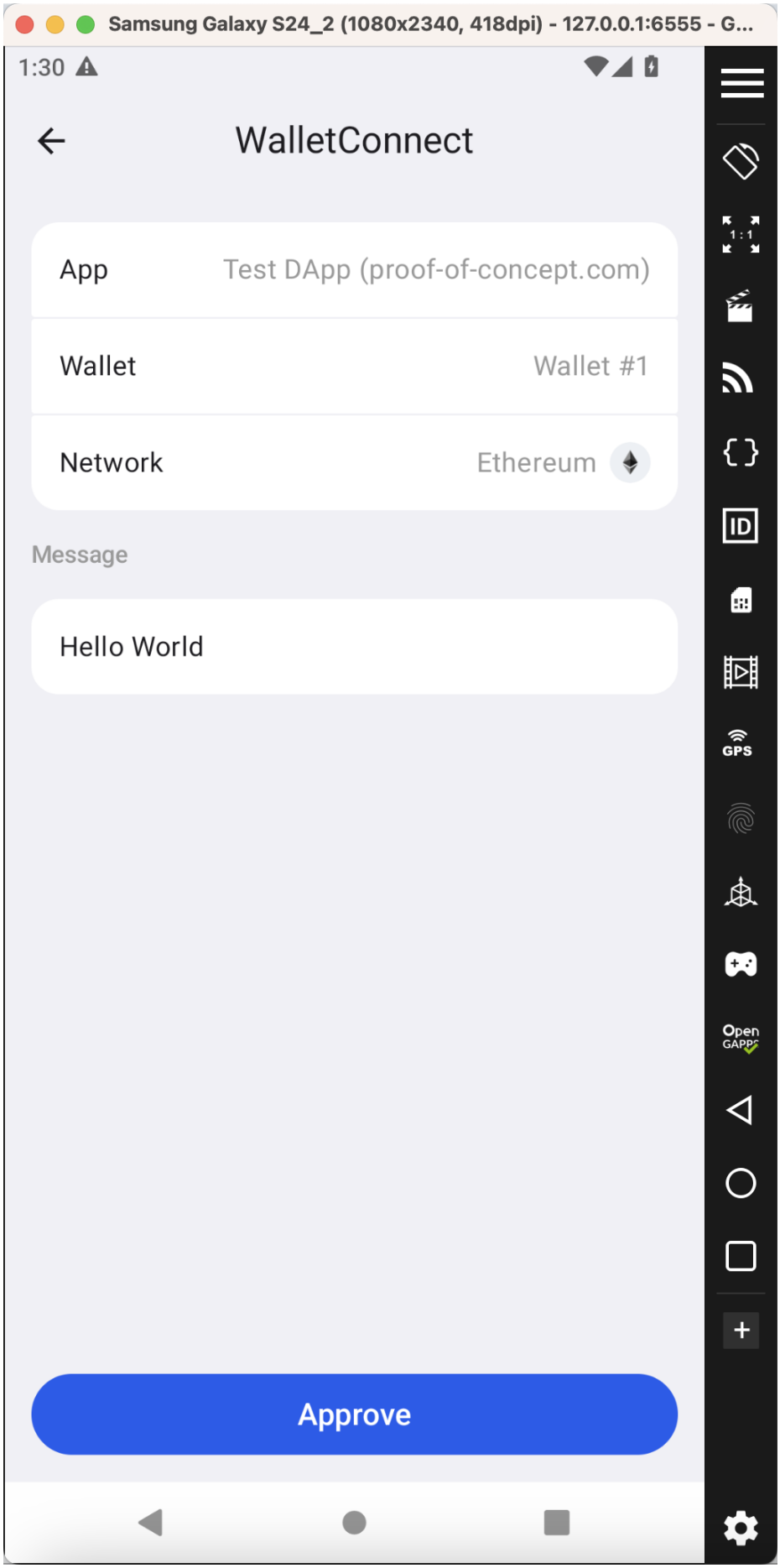
The wallet continues to accept `eth_sign` requests, allowing arbitrary data to be signed without standard domain separation or a human-readable preview. This can lead to security risks such as the user may unknowingly authorize messages that could be interpreted as permission for transactions or other sensitive actions. If the wallet interfaces do not clearly display the content of messages signed via `eth_sign`, it will increase the risk for the attackers to trick users into approving harmful operations, including phishing attempts, replay attacks, and signatures with ambiguous intent. Continued support for `eth_sign` increases users' exposure to these threats.

Impact

Users may not know what they are signing and could unknowingly approve a malicious transaction, leading to fund loss or account compromise.

Proof of Concept

The screenshot below shows that the wallet accepts an eth_sign request sent via WalletConnect and the signing prompt appears without any warning or alert about the use of eth_sign or the message details.


Samsung Galaxy S24_2 (1080x2340, 418dpi) - 127.0.0.1:6555 - G...
1:30
WalletConnect
App
Test DApp (proof-of-concept.com)
Wallet
Wallet #1
Network
Ethereum
Message
Hello World
Approve
GPS
Open GAPPS

Recommendation

It's recommended that support for the deprecated eth_sign method is removed or strictly limited to non-sensitive operations. Transition the wallet implementation to require more secure alternatives such as personal_sign (EIP-191) or eth_signTypedData (EIP-712), which offer proper domain separation and human-readable message previews. Ensure that any message signing functionality presents clear, understandable information to users before authorization, reducing the risk of phishing or unintended approvals.

Alleviation

[Gem Wallet, 03/04/2026]: The team heeded the advice and resolved the issue by removing support for eth_sign method in commit https://github.com/gemwalletcom/core/commit/ae852e1fe8b7f9df196eebf2f181d2e1907fc401.

GEW-08 | EIP-4361 SIWE Validation Fail - Origin And ChainID Mismatch

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

Incorrect validation of SIWE (EIP-4361) message domain, scheme (origin) and chainId can cause verification to fail or allow authentication messages to be accepted in the wrong context. The SIWE specification requires the domain and scheme to match the origin of the signing request with verification including contract signature checks performed on the chain specified by the declared chainId. Without these checks, an attacker could present a valid signature for one domain or chain in a different context and compromise the user's account.

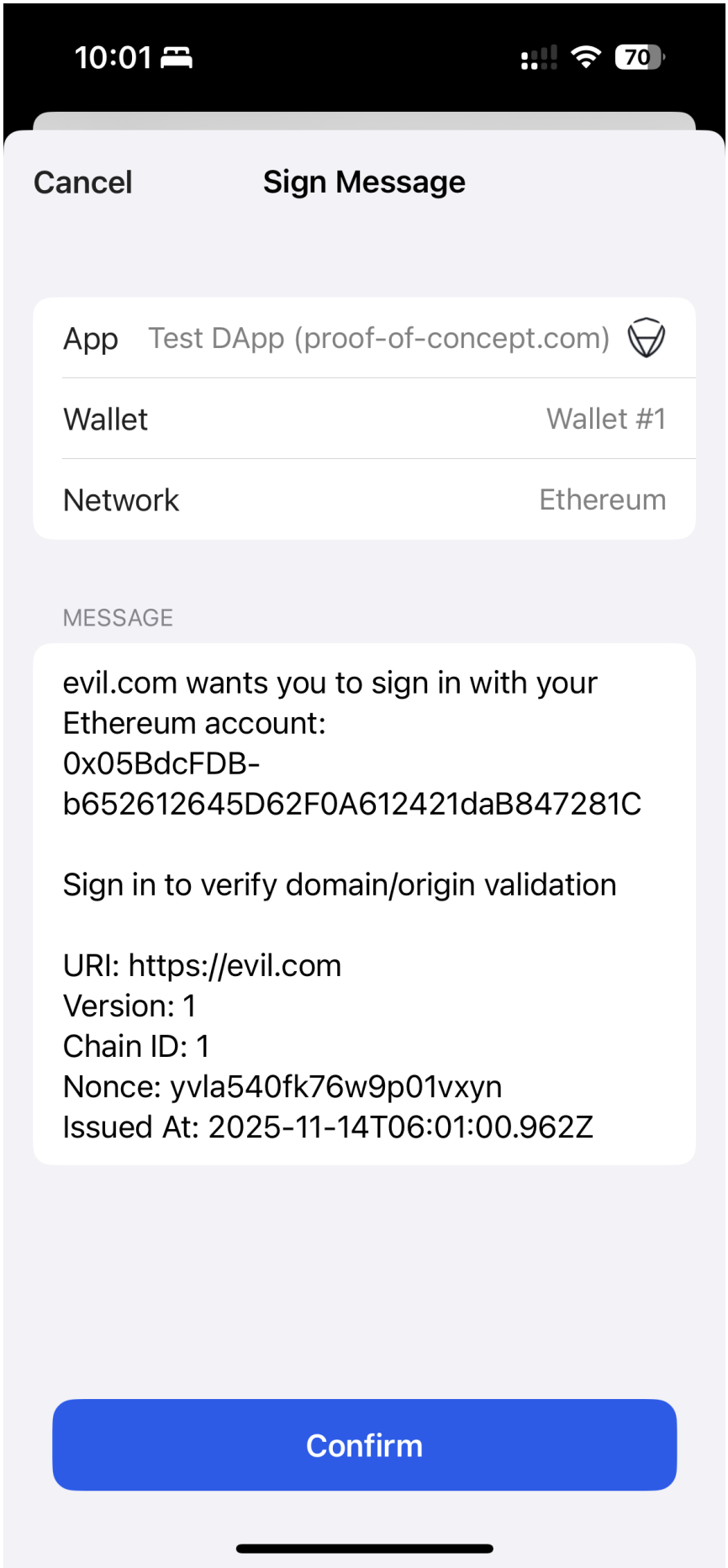
Impact

A malicious DApp can trick the wallet into signing authentication requests for a different website or blockchain network. This can lead to session hijacking or cross-origin authentication, where a signature intended for one trusted site is reused by an attacker to impersonate the user. Additionally, mismatched chain IDs may cause the wallet to generate signatures bound to unintended networks, enabling potential phishing, replay, or unauthorized login scenarios.

Proof of Concept

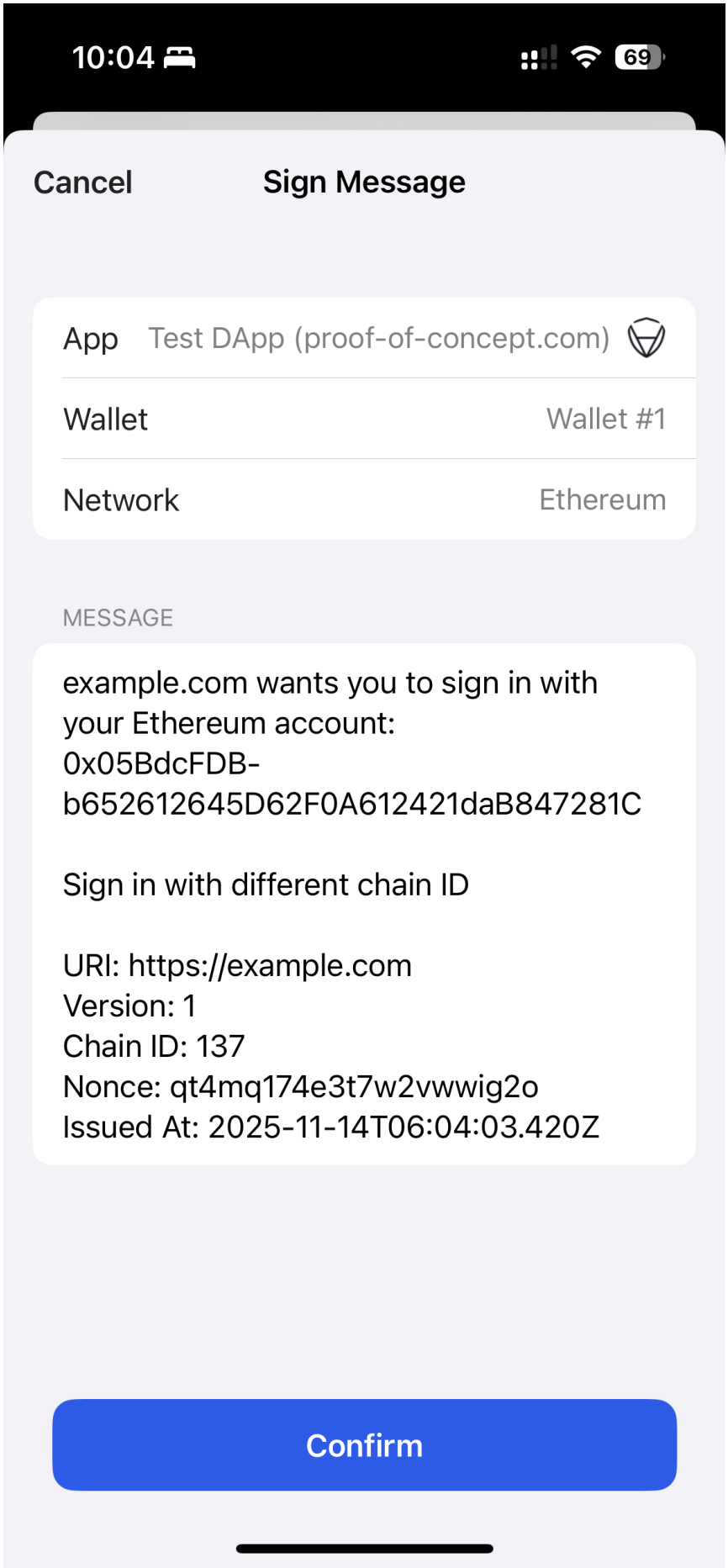
Origin Mismatch

The screenshot below shows that the domain of currently connected dApp is `proof-of-concept.com`, but the signing message displays `evil.com`, and the app does not display any warning to alert the client.


10:01
70
Cancel
Sign Message
App
Test DApp (proof-of-concept.com)
Wallet
Wallet #1
Network
Ethereum
MESSAGE
evil.com wants you to sign in with your Ethereum account:
0x05BdcFDB-b652612645D62F0A612421daB847281C
Sign in to verify domain/origin validation
URI: https://evil.com
Version: 1
Chain ID: 1
Nonce: yvla540fk76w9p01vxyn
Issued At: 2025-11-14T06:01:00.962Z
Confirm

Chain ID Mismatch

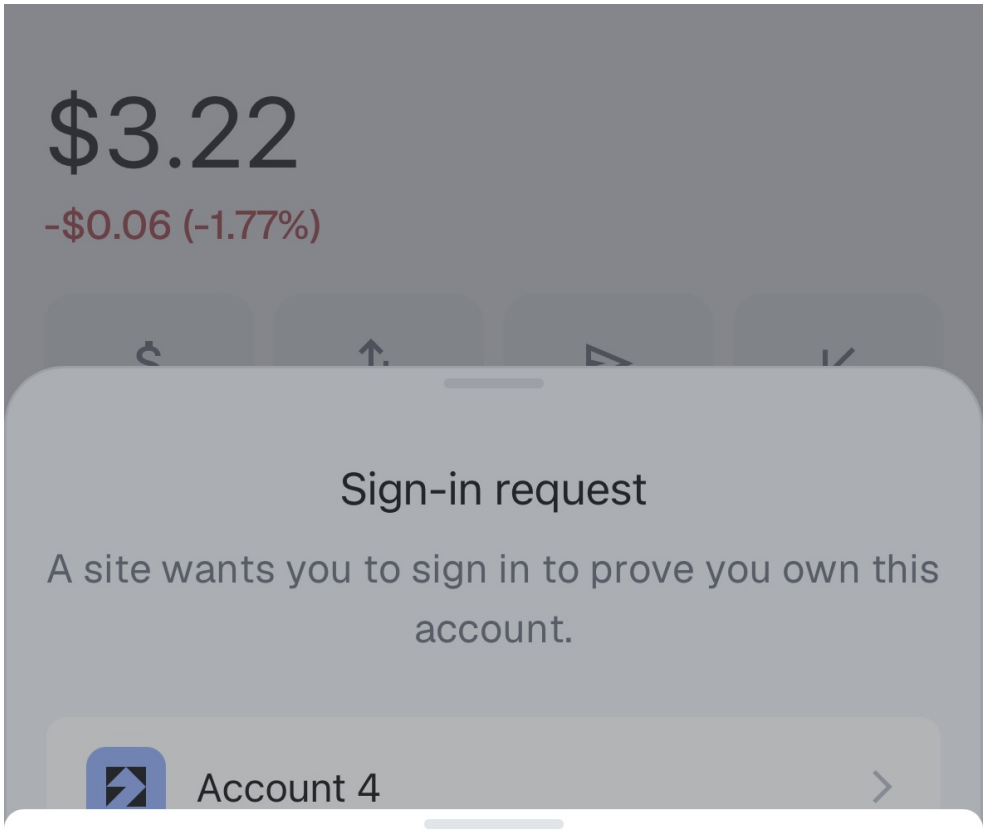
The following is another example where the chainId is `137` but the wallet is currently connected to `1`, and the app does not warn the client.


10:04
69
Cancel
Sign Message
App
Test DApp (proof-of-concept.com)
Wallet
Wallet #1
Network
Ethereum
MESSAGE
example.com wants you to sign in with your Ethereum account:
0x05BdcFDB-b652612645D62F0A612421daB847281C
Sign in with different chain ID
URI: https://example.com
Version: 1
Chain ID: 137
Nonce: qt4mq174e3t7w2vwwig2o
Issued At: 2025-11-14T06:04:03.420Z
Confirm

Recommendation

It's recommended that developers implement strict validation to ensure the domain/scheme in SIWE (EIP-4361) messages exactly match the origin of the signing request, and that the chainId declared in the message aligns with the user's current connected network. The origin should be read from a trusted data source, such as walletConnect session, and provide a clear warning to the user if these values do not correspond. This validation should be enforced during the authentication process to prevent signatures from being misused in unintended domains or networks.

MetaMask displays the following warning when an origin mismatch is detected.



Suspicious sign-in request

The site making the request is not the site you're signing into. This could be an attempt to steal your login credentials.

☐ I have acknowledged the alert and still want to proceed

Got it

For additional reference: https://eips.ethereum.org/EIPS/eip-4361#wallet-implementer-steps

Alleviation

[Gem Wallet, 04/03/2026]: The team heeded the advice and resolved the issue by displaying mismatch error messages and blocking the user from confirming in iOS version 1.3.355 and Android version 2.0.10.

GEW-09 Insufficient Display Validation In Permit / Permit2 Signature Requests

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

During testing of EIP-2612 Permit and Permit2 signature flows through WalletConnect, it was observed that the wallet does not properly validate or display several critical fields related to token approval. When signing Permit or Permit2 requests, the wallet fails to clearly present information such as spender type, allowance range, and expiration time, and also truncates multi-token details.

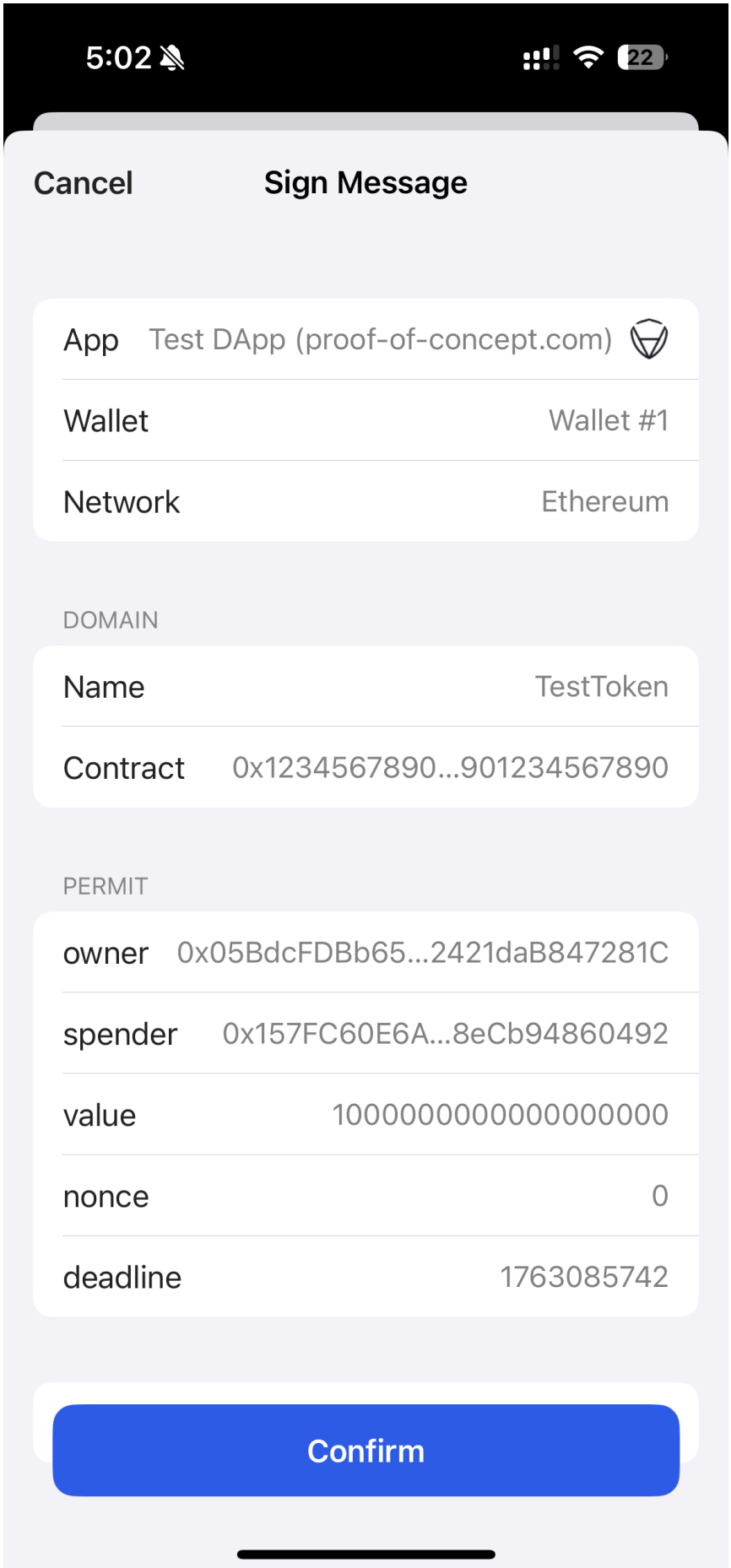
Impact

A malicious DApp can craft deceptive Permit or Permit2 signature requests to trick users into granting excessive or permanent token allowances, or to authorize arbitrary EOA spenders. Without clear display of spender type, allowance amount, expiration period, or affected tokens, users may unknowingly approve high-risk permissions.

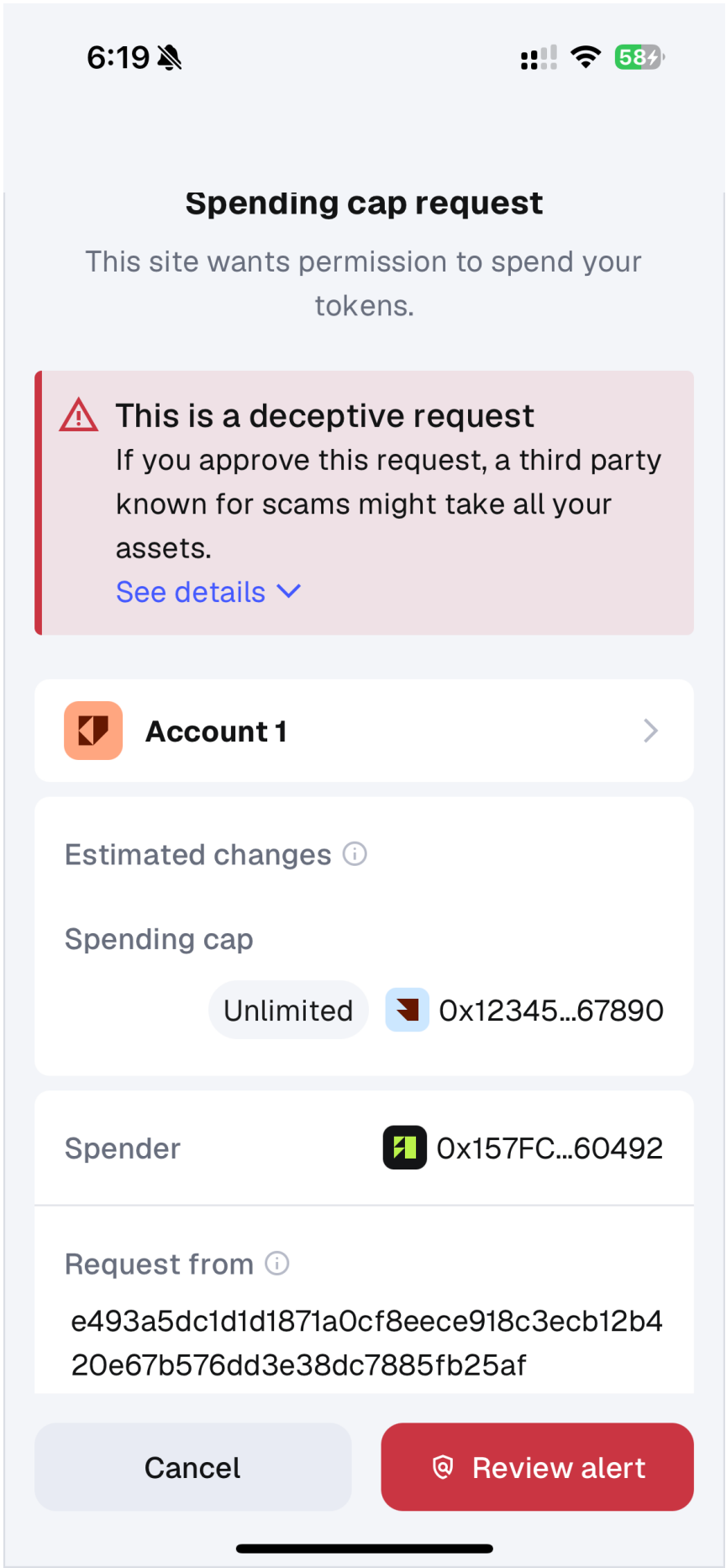
Proof of Concept

EOA Spender

When the spender field is an EOA address (non-contract), the wallet allows signing without any risk warning or indication that the spender is not a verified smart contract.

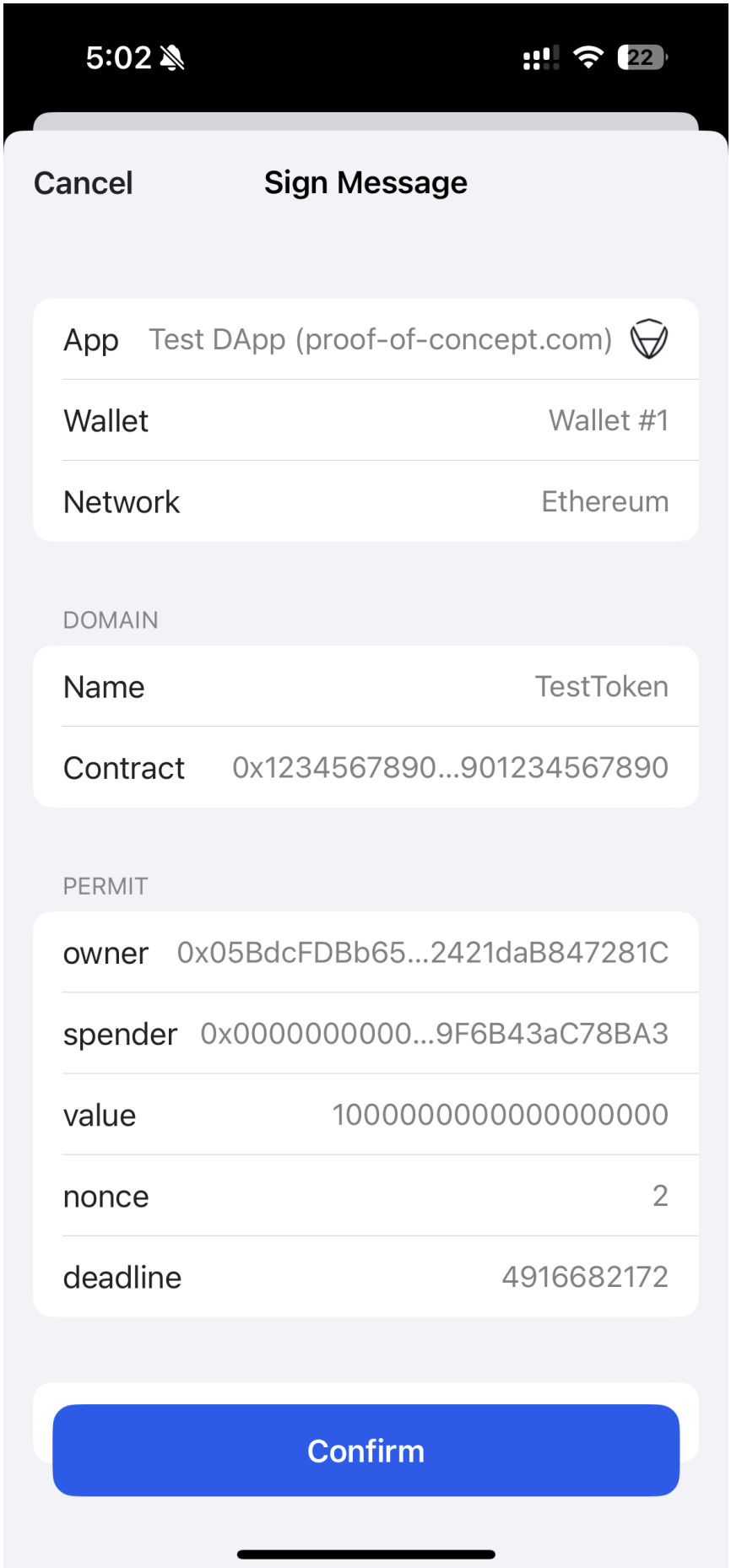


As a reference, MetaMask displays an alert indicating a deceptive request.

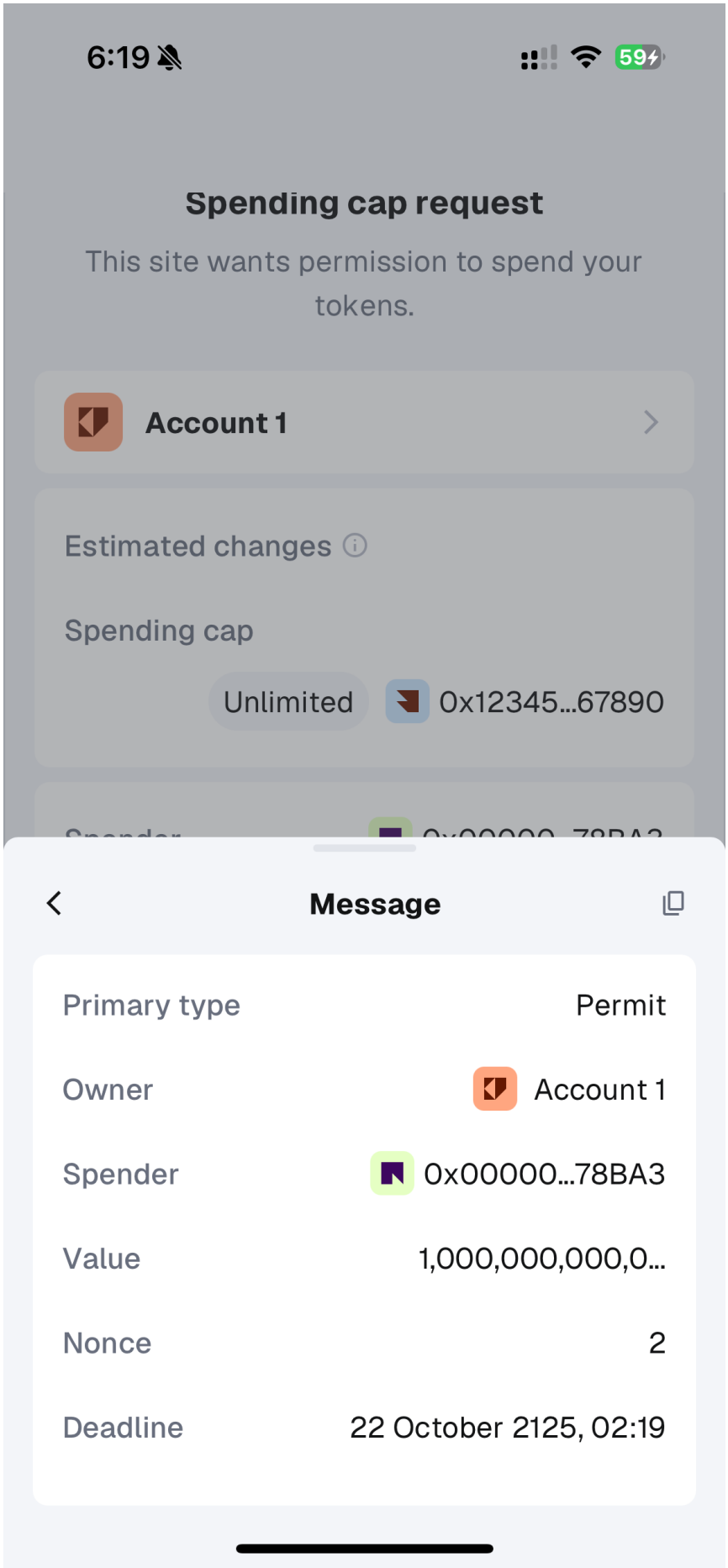

6:19
58
Spending cap request
This site wants permission to spend your tokens.
This is a deceptive request
If you approve this request, a third party known for scams might take all your assets.
See details
Account 1
Estimated changes
Spending cap
Unlimited
0x12345...67890
Spender
0x157FC...60492
Request from
e493a5dc1d1d1871a0cf8eece918c3ecb12b4
20e67b576dd3e38dc7885fb25af
Cancel
Review alert

Excessive Expiration

When the excessive expiration is set to one year later, the wallet displays the raw timestamp (e.g., 4916682172) instead of converting it into a readable date format (eg., Oct 21 2125 01:02:52 UTC/GM) and alert user about the abnormal expiration date, preventing users from understanding the actual authorization duration.

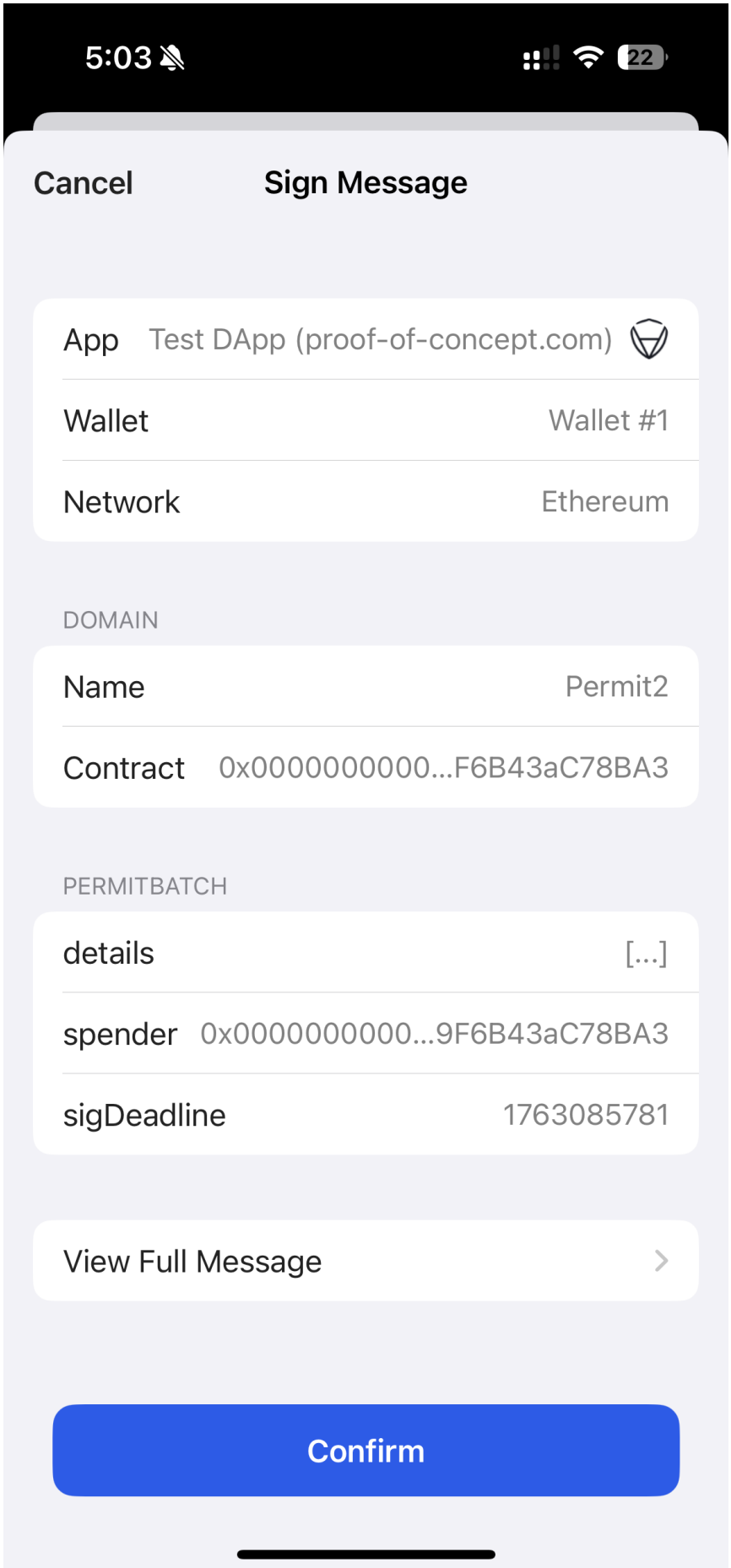


As a reference, MetaMask displays the readable date in message detail.

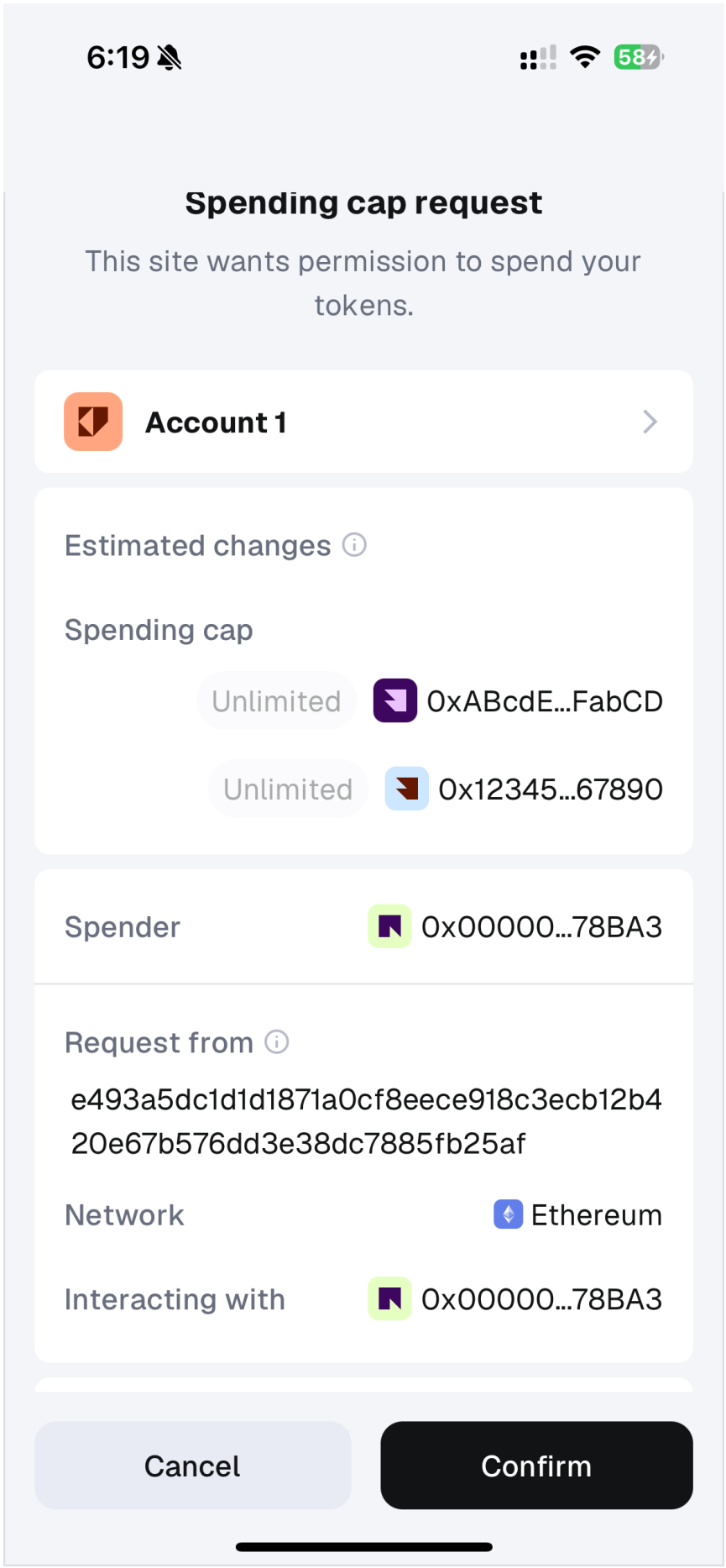

6:19
Spending cap request
This site wants permission to spend your tokens.
Account 1
Estimated changes
Spending cap
Unlimited
0x12345...67890
Message
Primary type
Permit
Owner
Account 1
Spender
0x00000...78BA3
Value
1,000,000,000,0...
Nonce
2
Deadline
22 October 2125, 02:19

Permit2 Multi-token

In Permit2 multi-token requests, the wallet collapses the list of token details into an abbreviated array representation such as [...], without showing token symbols or contract addresses, leaving users unable to identify which tokens are being authorized.



As a reference, MetaMask clearly displays multi-token information in signing request.


6:19
58
Spending cap request
This site wants permission to spend your tokens.
Account 1
Estimated changes
Spending cap
Unlimited
0xABcdE...FabCD
Unlimited
0x12345...67890
Spender
0x00000...78BA3
Request from
e493a5dc1d1d1871a0cf8eece918c3ecb12b4
20e67b576dd3e38dc7885fb25af
Network
Ethereum
Interacting with
0x00000...78BA3
Cancel
Confirm

Recommendation

The wallet should validate spender type and display a warning for EOAs, convert expiration timestamps to human-readable time, and show full token details in multi-token Permit2 requests to ensure users clearly understand authorization scope and risk.

Alleviation

[Gem Wallet, 04/07/2026]: The team heeded the advice and resolved the issue by implementing the following fixes in iOS version 1.3.355 and Android version 2.0.16. The wallet displays warnings for unlimited token approvals and excessive expiration, and provides detailed information for Permit2 multi-token requests. It displays an error message and blocks confirmation when the spender is an EOA address.

GEW-10 Missing Detection Of Unknown Spenders And Allowance Of Unlimited Token Approval

Category	Severity	Location	Status
Security Misconfiguration	Medium	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Resolved

Description

The application lacks mechanisms to detect or alert users when they grant high or unlimited token allowances to unknown or unverified spenders. When approving allowances for ERC-20 (approve) and ERC-721 / ERC-1155 (setApprovalForAll) assets to a suspicious or unknown spender (eg. 0x0000000000000000000000000000000000000000), the wallet failed to display the approved amount. No warnings were shown for unlimited allowance values and suspicious spender addresses, leaving users unable to understand what they were authorizing. Approving unlimited or unusually large allowances for such spenders increases the risk that malicious contracts can transfer or drain user tokens without additional authorization. This risk can also amplified if the application's interface does not distinguish between trusted and unknown spenders or fails to notify users about the potential consequences of these approvals.

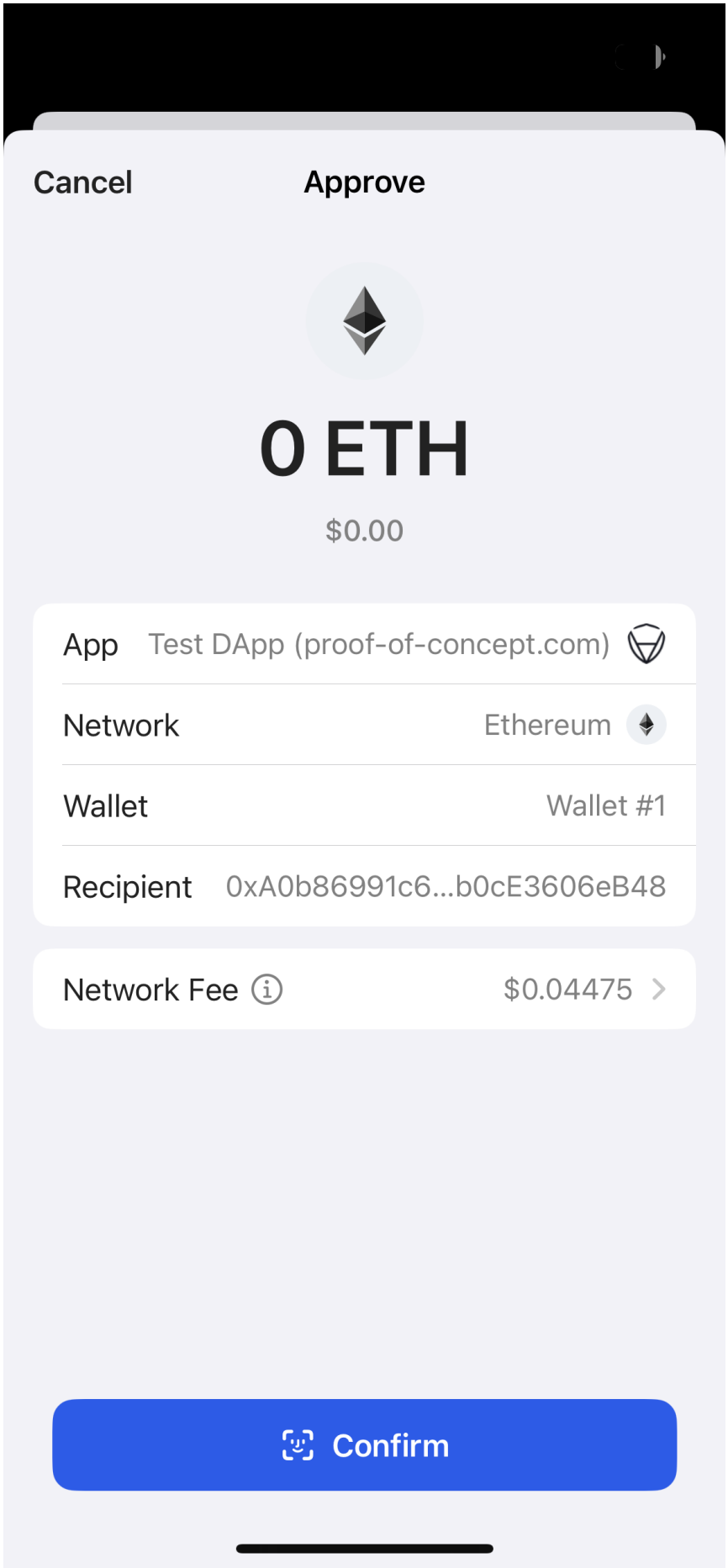
Impact

Users may not realize what they are approving and could unknowingly authorize a malicious contract to move unlimited assets from their account, resulting in loss of funds and account compromise.

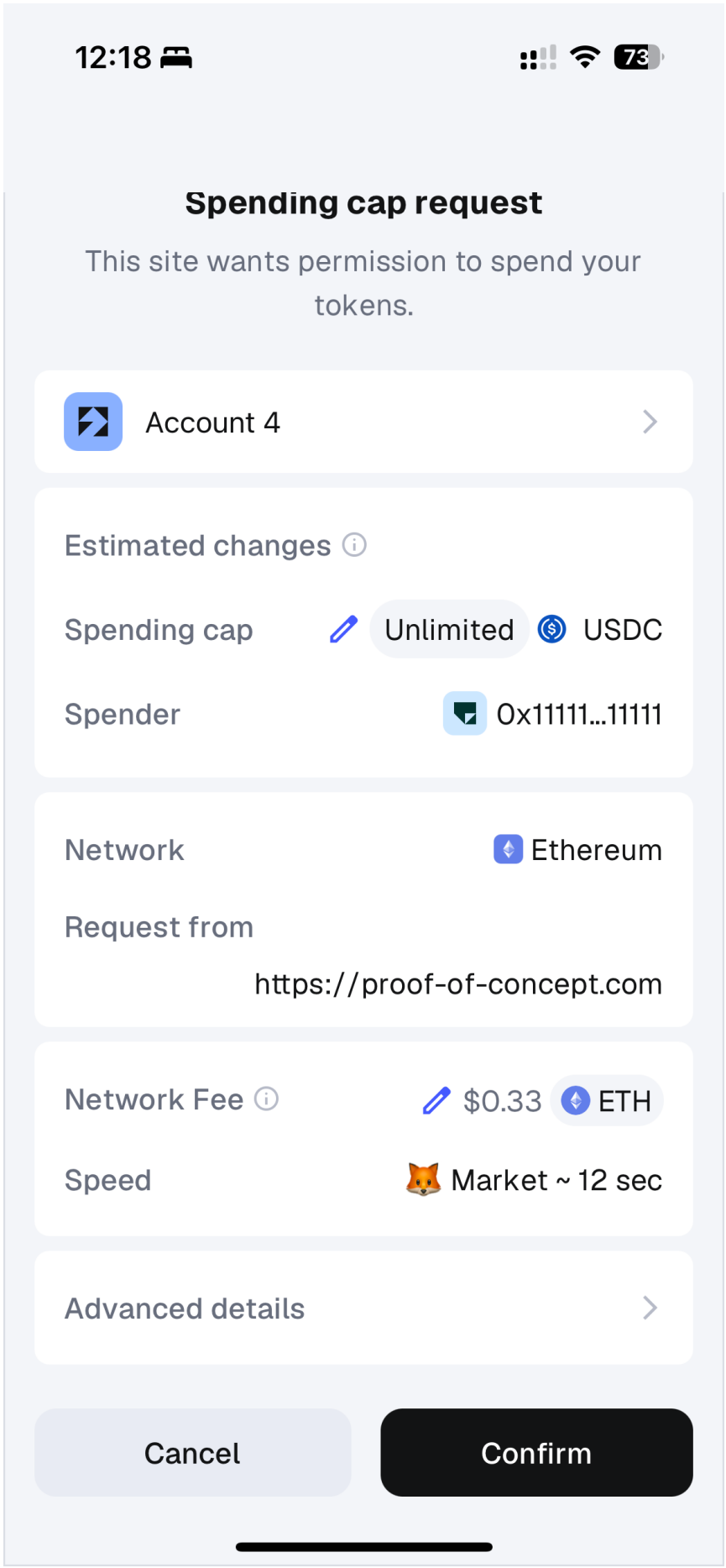
Reproduce Steps

ERC 20

The screenshot below shows that the wallet failed to detect and alert abnormal or unlimited amount of ERC20 token approval, and suspicious spender(0x1111111111111111111111111111111111111111).

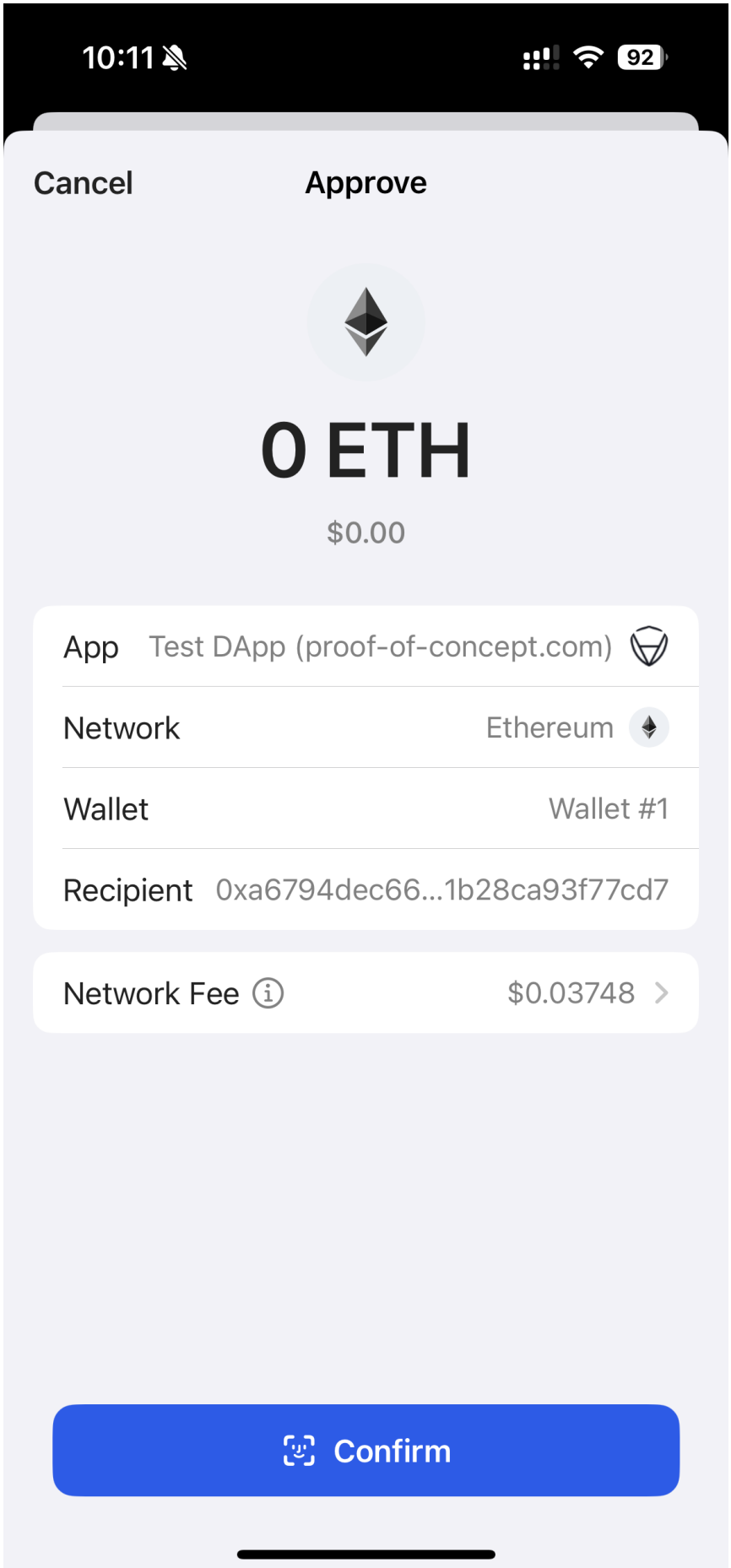


As a reference, Metamask displays unlimited token allowance in signing request.

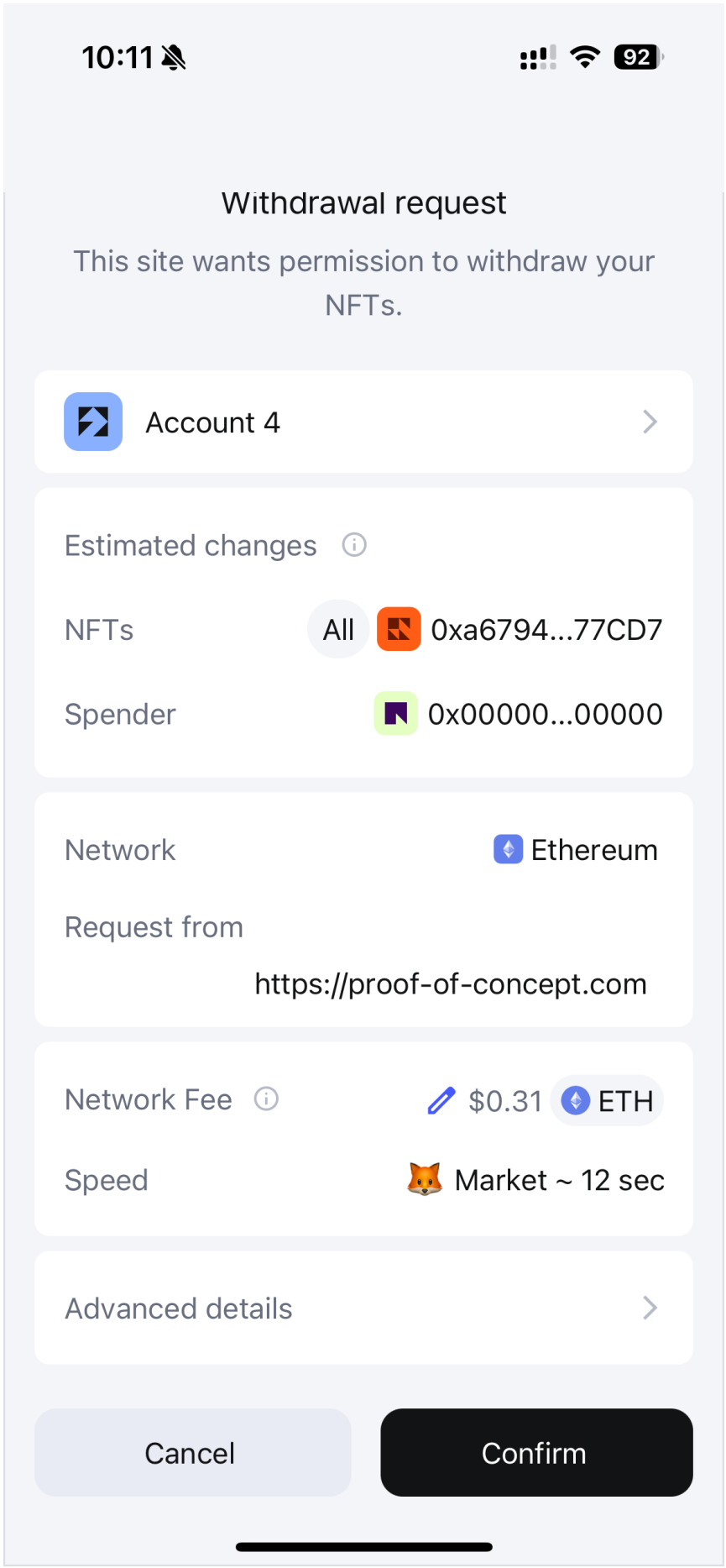

12:18
73
Spending cap request
This site wants permission to spend your tokens.
Account 4
Estimated changes
Spending cap
Unlimited
USDC
Spender
0x11111...11111
Network
Ethereum
Request from
https://proof-of-concept.com
Network Fee
$0.33
ETH
Speed
Market ~ 12 sec
Advanced details
Cancel
Confirm

ERC-721

The screenshot below shows that the wallet failed to detect and warn about an ERC-721 `setApprovalForAll` authorization request.

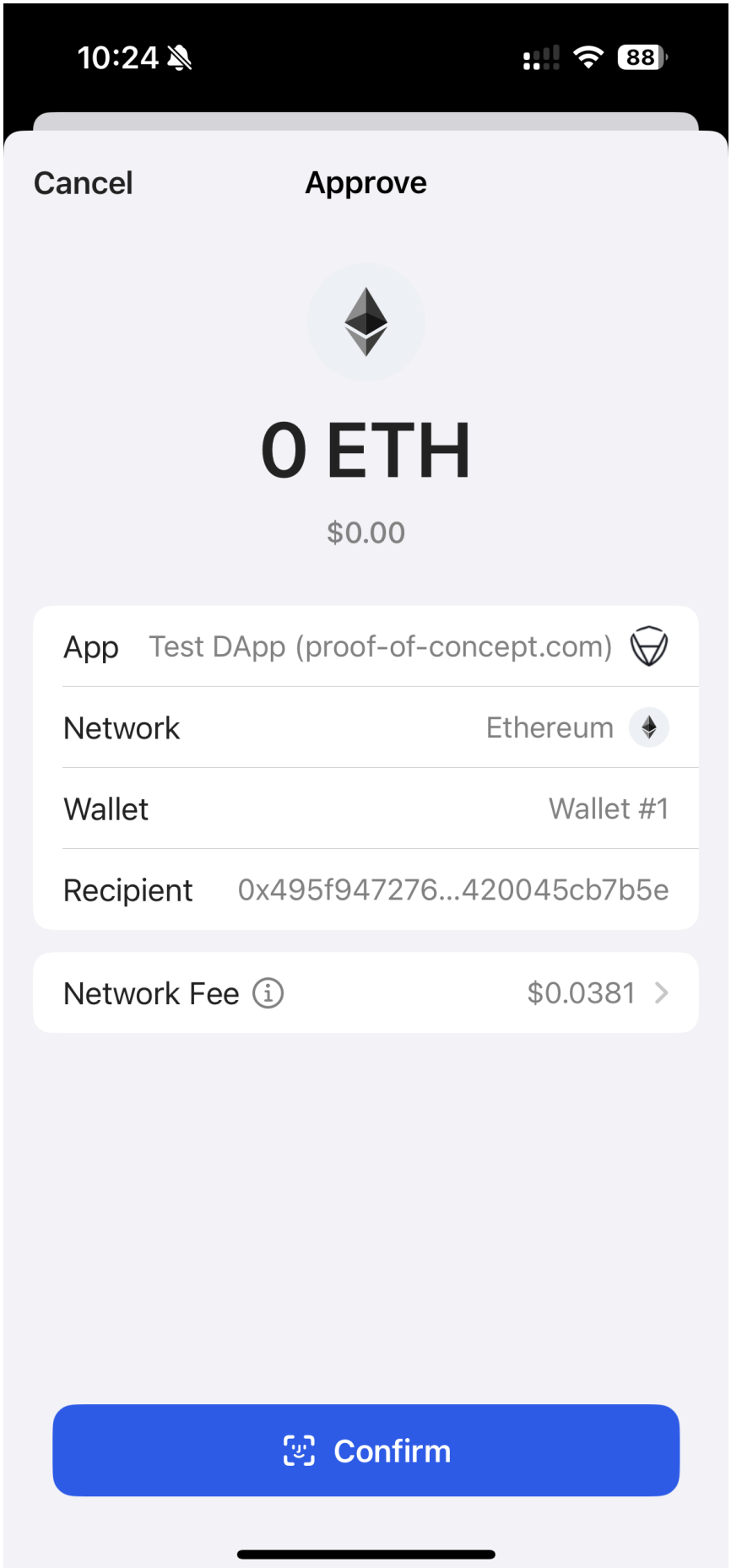


As a reference, Metamask displays approval for all ERC-721 tokens in signing request.

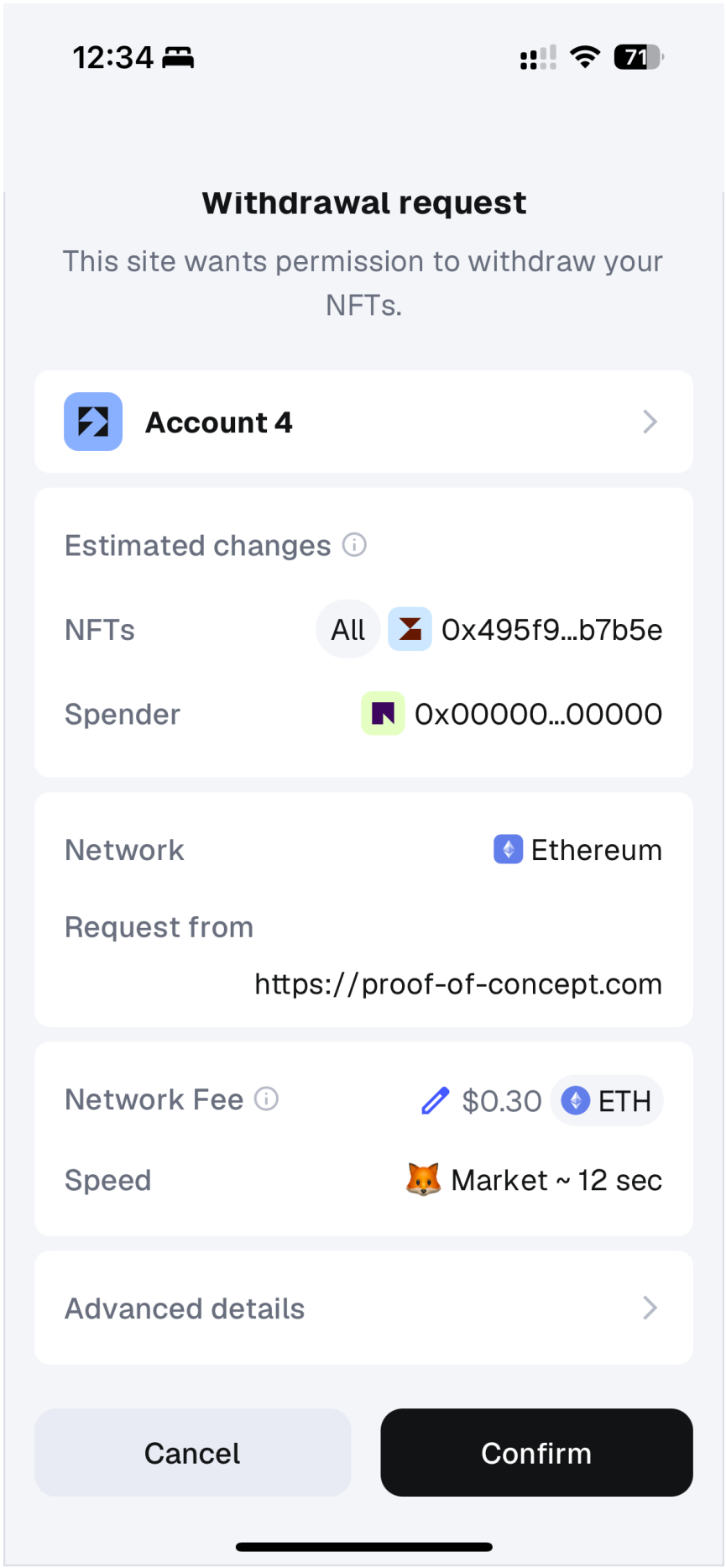

10:11
92
Withdrawal request
This site wants permission to withdraw your NFTs.
Account 4
Estimated changes
NFTs
All
0xa6794...77CD7
Spender
0x00000...00000
Network
Ethereum
Request from
https://proof-of-concept.com
Network Fee
$0.31
ETH
Speed
Market ~ 12 sec
Advanced details
Cancel
Confirm

ERC-1155

The screenshot below shows that the wallet failed to detect and warn about an ERC-1155 `setApprovalForAll` authorization request.



As a reference, Metamask displays approval for all ERC-1155 tokens in signing request.


12:34
71
Withdrawal request
This site wants permission to withdraw your NFTs.
Account 4
Estimated changes
NFTs
All
0x495f9...b7b5e
Spender
0x00000...00000
Network
Ethereum
Request from
https://proof-of-concept.com
Network Fee
$0.30
ETH
Speed
Market ~ 12 sec
Advanced details
Cancel
Confirm

Recommendation

It's recommended to enhance spender verification and user-facing approval transparency. The wallet should detect and flag unknown or unverified spender addresses, and require explicit confirmation for unlimited or unusually large approvals. When such cases occur, the wallet should clearly display the approved amount and warn the user that the spender is unrecognized or being granted broad access for the first time.

Alleviation

[Gem Wallet, 04/07/2026]: The team heeded the advice and resolved the issue by implementing appropriate warnings for `unlimited ERC-20 approval`, `ERC-721 Approval for All`, `ERC-1155 Approval for All` in iOS version 1.3.355 & Android version 2.0.16. The wallet displays an error message and blocks user confirmation when the spender is an EOA address, which may be overly restrictive, potentially preventing legitimate interactions.

GEW-02 | Lack Of SSL Pinning

Category	Severity	Location	Status
Security Misconfiguration	Low	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Acknowledged

Description

Certificate pinning is the process of associating the backend server with a particular X.509 certificate or public key instead of accepting any certificate signed by a trusted certificate authority. After storing ("pinning") the server certificate or public key, the mobile app will subsequently connect to the known server only. Withdrawing trust from external certificate authorities reduces the attack surface (after all, there are many cases of certificate authorities that have been compromised or tricked into issuing certificates to impostors).

The certificate can be pinned and hardcoded into the app or retrieved at the time the app first connects to the backend. In the latter case, the certificate is associated with ("pinned" to) the host when the host is seen for the first time. This alternative is less secure because attackers intercepting the initial connection can inject their own certificates.

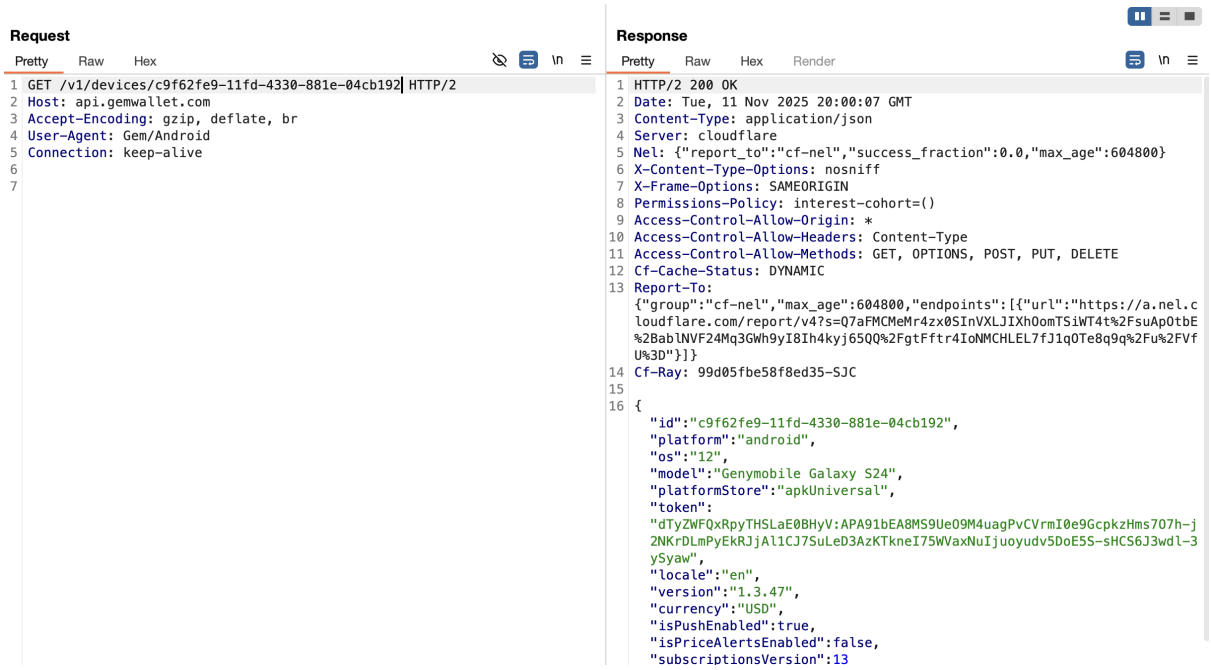
Impact

A lack of of SSL Pinning renders the application vulnerable to Man-in-the-Middle (MitM) attacks. In such an attack, an adversary can intercept, inspect, and potentially alter all network traffic between the application and its backend servers. For a cryptocurrency application, this has severe consequences: sensitive user data (e.g., transaction details, wallet balances, authentication tokens) could be intercepted.

Furthermore, an attacker could potentially inject malicious responses, manipulate transaction details, or redirect users to phishing sites, leading to unauthorized fund transfers, account compromise, or significant financial loss without the user's knowledge.

Reproduce Steps

The below screenshots show the intercepted requests in BurpSuite.



Recommendation

To effectively mitigate the risk of MitM attacks, the application must implement SSL Pinning. This involves implementing certificate pinning or, preferably, public key pinning, where the application explicitly trusts only specific server certificates or their public keys. The pinning logic should be implemented in native code (e.g., Kotlin/Java for Android) and should be highly resistant to runtime modification or bypass attempts.

Additionally, robust anti-tampering and anti-debugging techniques should be integrated to detect and prevent tools like Frida or Objection from interfering with the pinning mechanism. This includes integrity checks on the pinning code itself and obfuscation to make reverse engineering more difficult. Regularly updating the pinned certificates/keys is also crucial to maintain security.

Alleviation

[Gem Wallet, 02/27/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement.

GEW-03 | Custom Keyboards Allowed For Sensitive Inputs

Category	Severity	Location	Status
Information Disclosure	Low	Android Production Build v1.3.47 IOS Production Build v1.3.292(7781)	Acknowledged

Description

Custom keyboards are standalone keyboard applications installed by device users to provide additional functionality when typing, such as swipe-style text input or additional characters. Allowing the use of custom keyboards can increase the risk of keylogging.

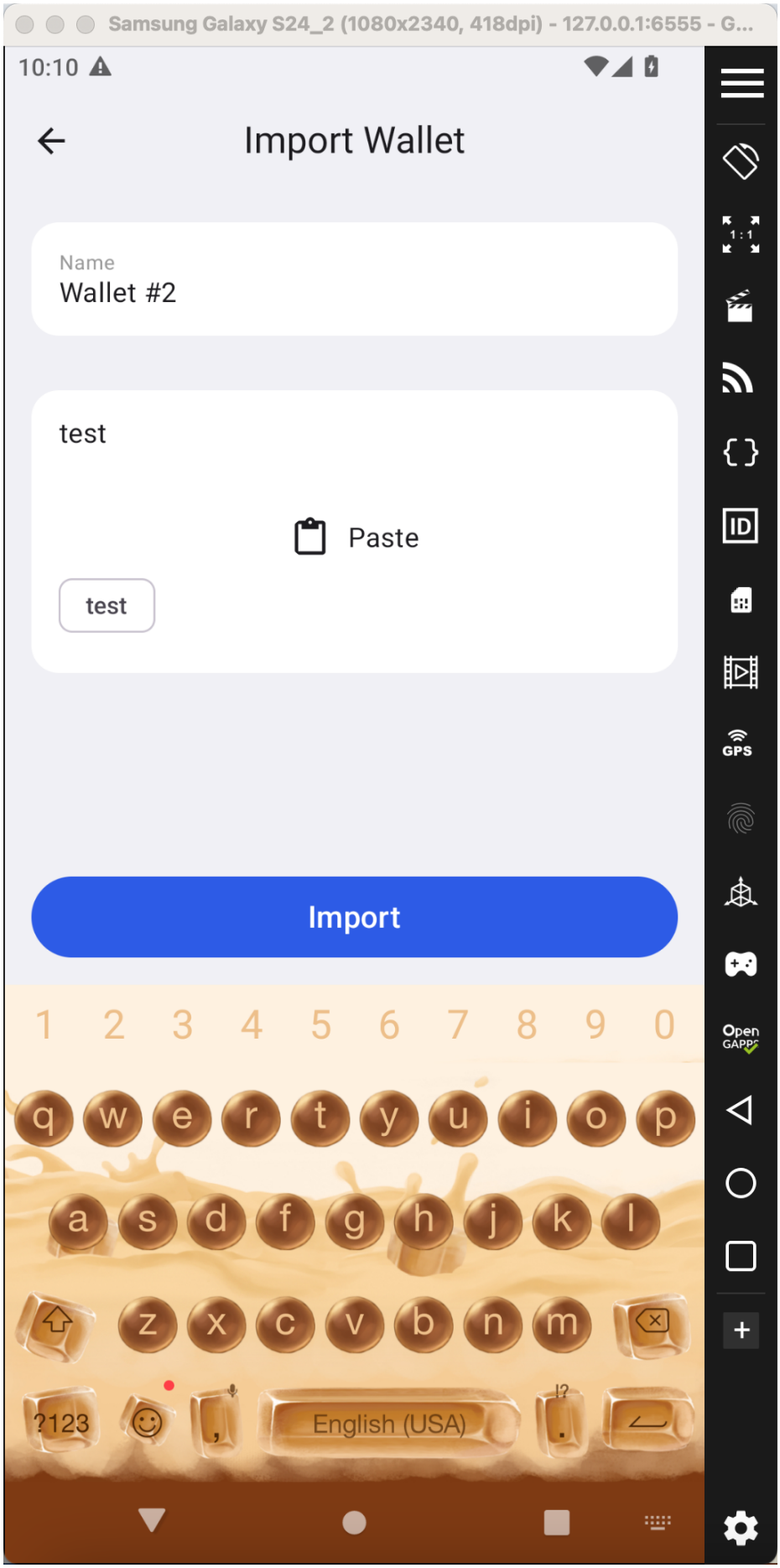
Impact

The use of custom keyboards is considered dangerous as these are usually developed by third parties and are often capable of recording and transmitting data to third party servers. It is therefore possible that a malicious keyboard application could record all user input and transmit this data to an attacker's server in order to harvest sensitive information. The application allowed the usage of third party keyboards for the wallet seed phrase forms. Use of third-party or untrusted Input Method Editors (IME) could unnecessarily expose sensitive user information as there is a risk that data entry to the application may be intercepted, logged or transmitted to a malicious third party.

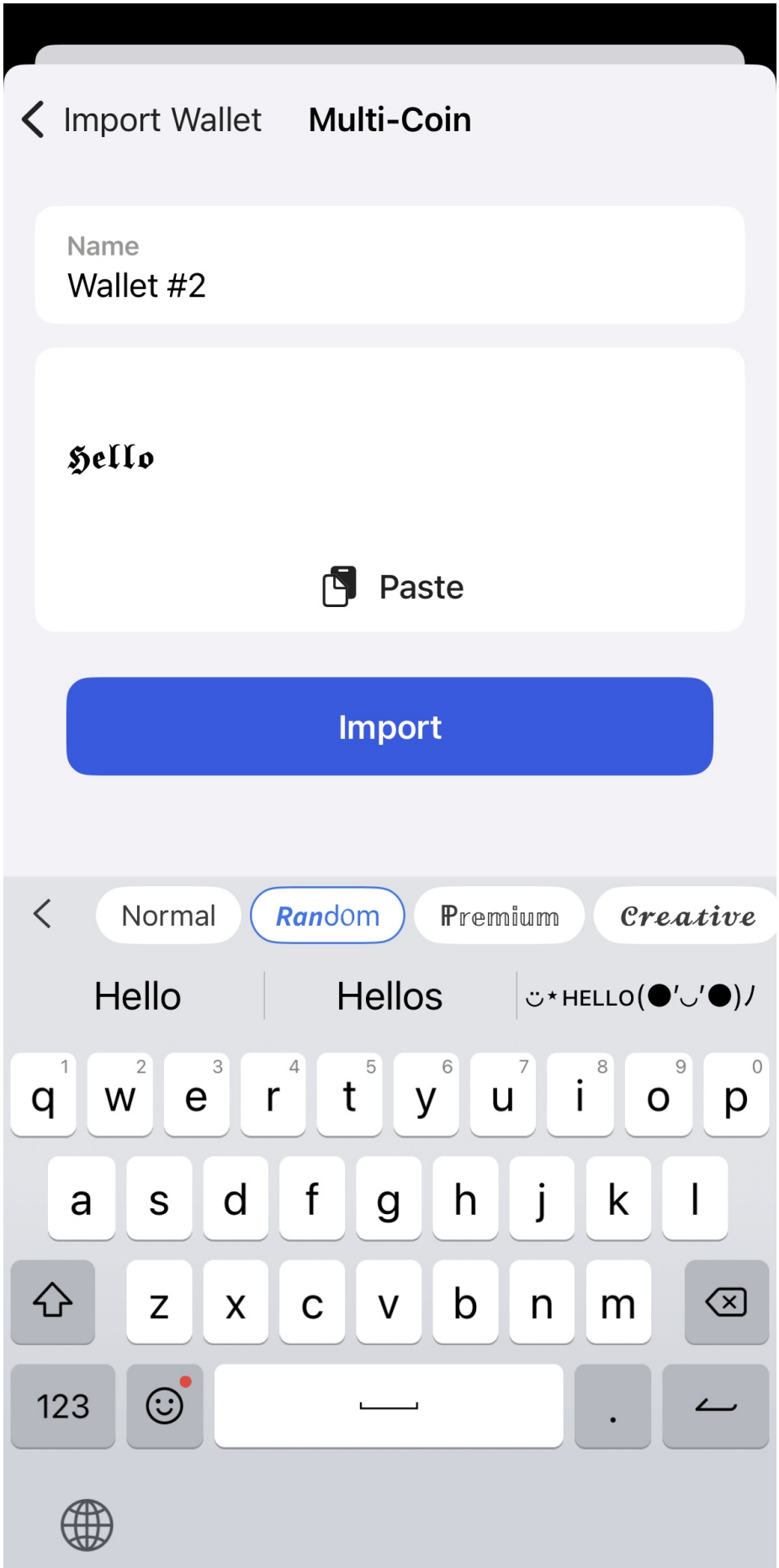
Proof of Concept

The following screenshot shows that it was possible to set the keyboard to the custom keyboard.

- Android


Samsung Galaxy S24_2 (1080x2340, 418dpi) - 127.0.0.1:6555 - G...
10:10
Import Wallet
Name
Wallet #2
test
Paste
test
Import
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
English (USA)

- IOS


Import Wallet
Multi-Coin
Name
Wallet #2
Hello
Paste
Import
Normal
Random
Premium
Creative
Hello
Hellos
ʕ•ᴥ•ʔ HELLO (●'◡'●) ﾉ
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123

Recommendation

On iOS, third-party keyboards can be explicitly disabled for the application by specifying the appropriate extension point identifier, as detailed in Apple's documentation:

- https://developer.apple.com/documentation/uikit/uiapplication/extensionpointidentifier

On Android, there is no official method to force a specific keyboard for data entry. However, it is recommended to implement a custom in-app keyboard mechanism for all sensitive input fields. This keyboard should not be an external IME but should be built using the app's UI components. For instance, a custom keyboard can be created by extending the LinearLayout class and defining Button widgets for keys. These buttons can use onClick listeners to return character values securely. This approach keeps all key handling within the application context, reducing the risk of keystroke logging.

Alleviation

[Gem Wallet, 02/27/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. They plan to improve this with custom keyboard in the future.

GEW-04 | Insecure Keystore Configuration

Category	Severity	Location	Status
Security Misconfiguration	Low	Android Production Build v1.3.47	Acknowledged

Description

The Android application is designed to create a cryptographic key from the Keystore. However, the Keystore configuration does not implement the recommended security options.

- `setUserAuthenticationRequired(true)`: It forces the user to have authenticated recently.
- `setUserAuthenticationValidityDurationSeconds(-1)`: Define how long authentication is valid. With `-1`, the app requires authentication for every operation.
- `setIsStrongBoxBacked(true)`: StrongBox is a dedicated hardware security module (HSM). The `MasterKey` API is designed to fall back to a TEE (Trusted Execution Environment)-backed Keystore if StrongBox is requested but not available.

```
    private fun getStore(): SharedPreferences {
        val masterKey = MasterKey.Builder(context)
            .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
            .build()
        return EncryptedSharedPreferences.create(
            context,
            "pwd",
            masterKey,
            EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
            EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM,
        )
    }
```

Impact

The absence of these security protections leaves the keystore entries vulnerable to unauthorized access. An attacker who gains access to the device might be able to retrieve sensitive cryptographic keys, thereby compromising the confidentiality and integrity of the application.

Recommendation

It is recommended the following actions to improve the security of the keystore entries:

- **Enforce user authentication for key access:** Update keystore entries to require user authentication before granting access. Configure this by setting `setUserAuthenticationRequired(true)` during key generation and define an

appropriate authentication validity period using `setUserAuthenticationValidityDurationSeconds(int seconds)` (e.g., `-1` for per-operation authentication on highly sensitive keys).

- **Leverage hardware-backed key security:** Ensure keys are protected by secure hardware (TEE or StrongBox) for an additional layer of security against software and hardware attacks. For critical entries, prioritize StrongBox by configuring keys with `setIsStrongBoxBacked(true)` (API 28+), which also ensures that user authentication requirements are enforced by dedicated secure hardware.

The following configurations increase the security level but may impact the user experience.

- **Implement user presence verification for critical operations:** For keys guarding highly sensitive operations, ensure the user is physically present and actively verifying their identity. Configure such operational keys by setting `setTrustedUserPresenceRequired(true)` (typically requiring fresh biometric input).
- **Require user confirmation for sensitive actions:** Ensure users must explicitly confirm their intention to perform sensitive operations, such as signing a transaction. Configure the relevant operational keys by setting `setUserConfirmationRequired(true)` to trigger a system-provided confirmation prompt.

Alleviation

[Gem Wallet, 02/27/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement.

GEW-11 | Missing Detection For Request Flooding

Category	Severity	Location	Status
Security Misconfiguration	Low	IOS Production Build v1.3.292(7781) Android Production Build v1.3.47	Acknowledged

Description

When receiving a large number of signature requests within a short period from a malicious DApp, the wallet failed to detect or mitigate request flooding. This behavior can overwhelm the user interface, render the wallet temporarily unusable, and increase the likelihood of user fatigue leading to accidental confirmations. Furthermore, the wallet does not alert the user or provide an option to temporarily block the DApp during such flooding events.

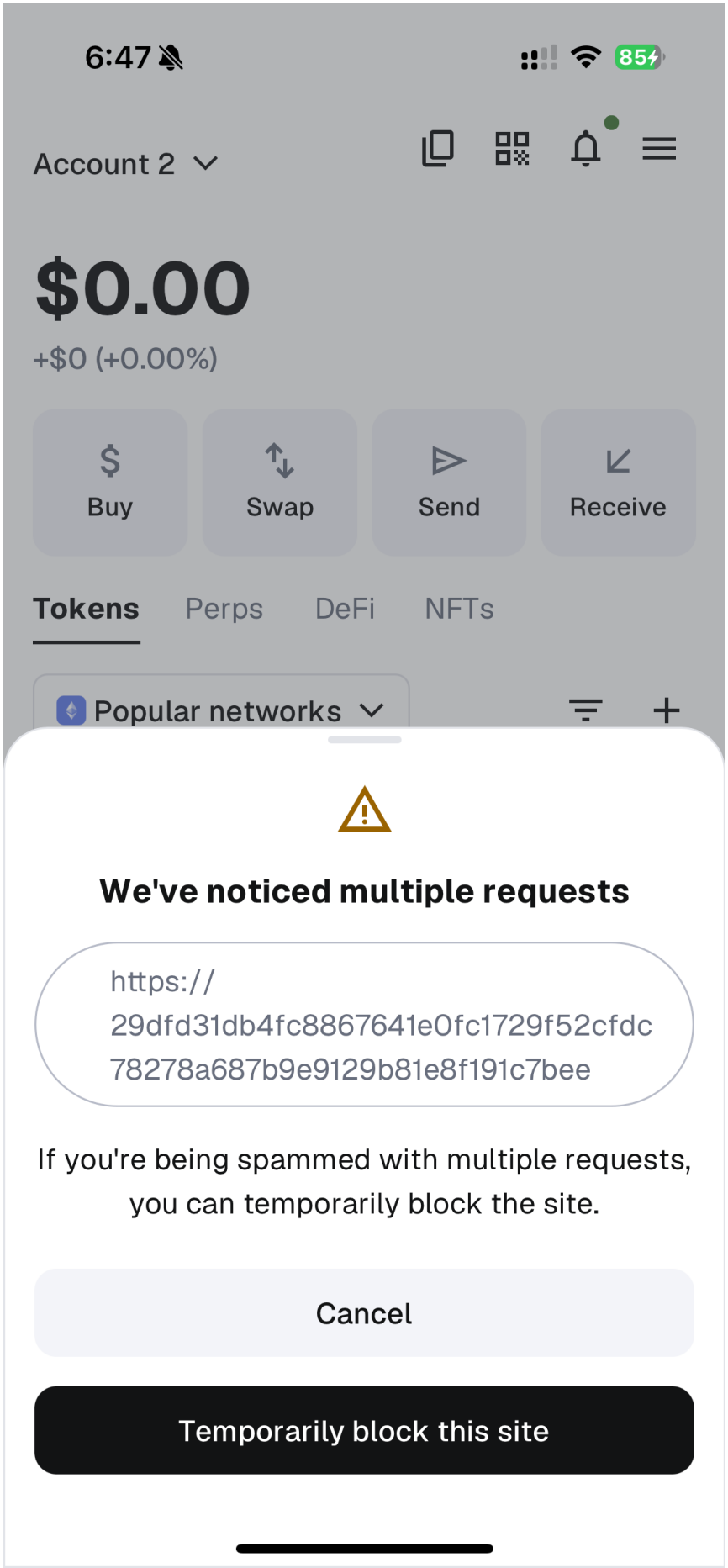
Impact

A malicious DApp may trigger a denial-of-service by flooding the wallet with queued signature requests. Users may be forced to manually reject each request, disrupting normal wallet operation and potentially leading to unintended signatures.

Recommendation

If multiple requests from the same domain are detected in a short period, the wallet should prompt the user to verify whether the domain is legitimate. Unknown or suspicious domains should be blocked by default. If the user chooses to block the domain, the application should prevent further connections to it, mitigating potential request flooding or malicious activity.

The following example from metamask shows that when multiple requests are detected as flooding, the user is allowed to block them.


6:47
Account 2
$0.00
+$0 (+0.00%)
Buy
Swap
Send
Receive
Tokens
Perps
DeFi
NFTs
Popular networks
We've noticed multiple requests
https://
29dfd31db4fc8867641e0fc1729f52cfdc
78278a687b9e9129b81e8f191c7bee
If you're being spammed with multiple requests, you can temporarily block the site.
Cancel
Temporarily block this site

Alleviation

[Gem Wallet, 02/27/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement.

GEW-12 | Mnemonic Display Allows Screenshot

Category	Severity	Location	Status
Information Disclosure	Low	IOS Production Build v1.3.292(7781)	Resolved

Description

The mnemonic phrase is a list of words used to recover a wallet account. Obtaining the mnemonic or private key could potentially allow an attacker to gain full control of the wallet. The application does not have a mechanism to prevent users from taking screenshots of the displayed wallet secrets, nor does it display a warning to remind users of the risks associated with taking screenshots.

Impact

Third-party apps with full photo access on an iPhone can access screenshots stored on the device. These apps could potentially retrieve the mnemonic if it is included in a screenshot taken by the user.

Recommendation

There is no built-in solution on iOS to prevent users from taking screenshots. It is recommended to hide or mask sensitive information when a screenshot is detected, or display a warning advising users not to capture screenshots while viewing wallet secrets.

Alleviation

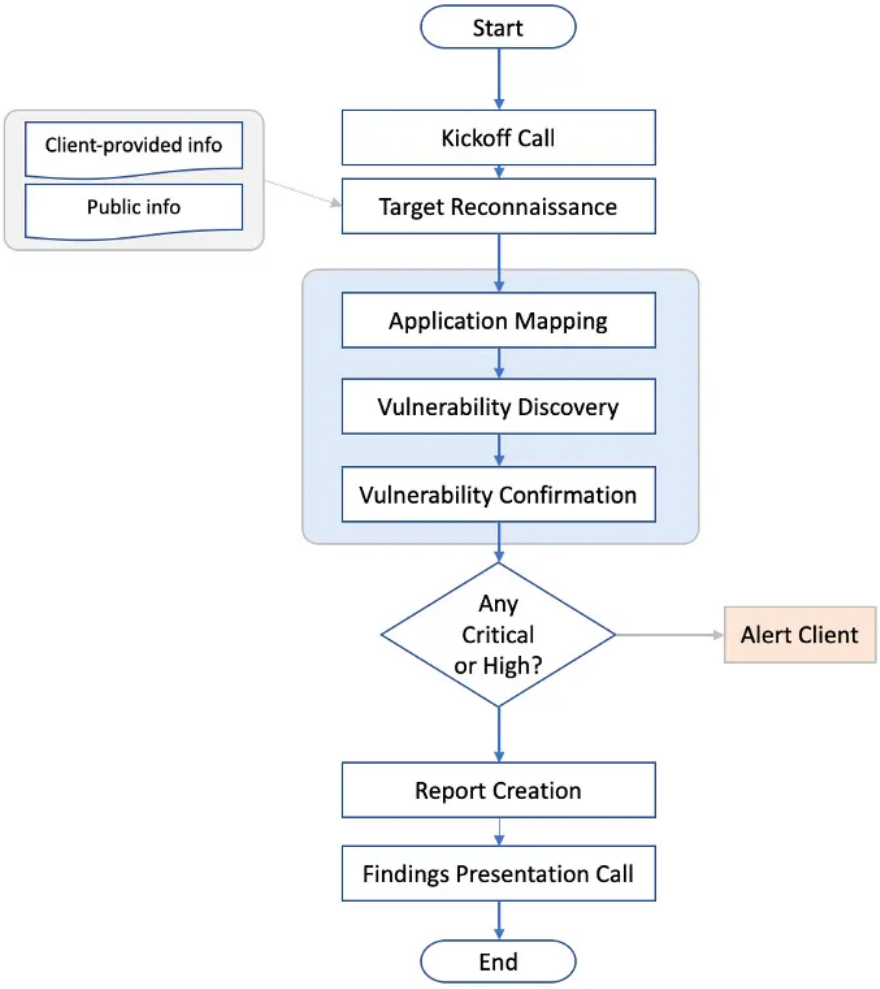
[Gem Wallet, 03/03/2026]: The team heeded the advice and resolved the issue by adding a warning to the user when a screenshot of the mnemonic is detected in commit 8cb8481933b29e2e0fba07e2083a414257af1196.

APPENDIX | GEM WALLET

Methodology

CertiK uses a comprehensive penetration testing methodology which adheres to industry best practices and standards in security assessments including from OWASP (Open Web Application Security Project), NIST, PTES (Penetration Testing Execution Standard).

Below is a flowchart of our assessment process:



Coverage and Prioritization

As many components as possible will be tested manually. Priority is generally based on three factors: critical security controls, sensitive data, and the likelihood of vulnerability.

Critical security controls will always receive the top priority in the test. If a vulnerability is discovered in the critical security control, the entire application is likely to be compromised, resulting in a critical-risk to the business. For most applications, critical controls will include the login page, but it could also include major workflows such as the checkout function in an online store.

The Second priority is given to application components that handle sensitive data. This is dependent on business priorities,

but common examples include payment card data, financial data, or authentication credentials.

Final priority includes areas of the application that are most likely to be vulnerable. This is based on CertiK' experience with similar applications developed using the same technology or with other applications that fit the same business role. For example, large applications will often have older sections that are less likely to utilize modern security techniques.

Reconnaissance

CertiK gathers information about the target application from various sources depending on the type of test being performed. CertiK obtains whatever information that is possible and appropriate from the client during scoping and supplements it with relevant information that can be gathered from public sources. This helps provide a better overall picture and understanding of the target.

Application Mapping

CertiK examines the application, reviewing its contents, and mapping out all its functionalities and components. CertiK makes use of different tools and techniques to traverse the entire application and document all input areas and processes. Automated tools are used to scan the application and it is then manually examined for all its parameters and functionalities. With this, CertiK creates and widens the overall attack surface of the target application.

Vulnerability Discovery

Using the information that is gathered, CertiK comes up with various attack vectors to test against the application. CertiK uses a combination of automated tools and manual techniques to identify vulnerabilities and weaknesses. Industry-recognized testing tools will be used, including Burp Suite, Nikto, Metasploit, and Kali. Furthermore, any controls in place that would inhibit the successful exploitation of a particular system will be noted.

Vulnerability Confirmation

After discovering vulnerabilities in the application, CertiK validates the vulnerabilities and assesses its overall impact. To validate, CertiK performs a Proof-of-Concept of an attack on the vulnerability, simulating real world scenarios to prove the risk and overall impact of the vulnerability.

Through CertiK's knowledge and experience on attacks and exploitation techniques, CertiK is able to process all weaknesses and examine how they can be combined to compromise the application. CertiK may use different attack chains, leveraging different weaknesses to escalate and gain a more significant compromise.

To minimize any potential negative impact, vulnerability exploitation was only attempted when it would not adversely affect production applications and systems, and then only to confirm the presence of a specific vulnerability. Any attack with the potential to cause system downtime or seriously impact business continuity was not performed. Vulnerabilities were never exploited to delete or modify data; only read-level access was attempted. If it appeared possible to modify data, this was noted in the list of vulnerabilities below.

Immediate Escalation of High or Critical Findings

If critical or high findings are found whereby application elements are compromised, client's key security contacts will be notified immediately.

Risk Assessment

Risk Level	CVSS Score	Impact	Exploitability
Critical	9.0-10.0	Root-level or full-system compromise, large-scale data breach	Trivial and straightforward
High	7.0-8.9	Elevated privilege access, significant data loss or downtime	Easy, vulnerability details or exploit code are publicly available, but may need additional attack vectors (e.g., social engineering)
Medium	4.0-6.9	Limited access but can still cause loss of tangible assets, which may violate, harm, or impede the org's mission, reputation, or interests.	Difficult, requires a skilled attacker, needs additional attack vectors, attacker must reside on the same network, requires user privileges
Low	0.1-3.9	Very little impact on an org's business	Extremely difficult, requires local or physical system access
Informational	0.0	Discloses information that may be of interest to an attacker.	Not exploitable but rather is a weakness that may be useful to an attacker should a higher risk issue be found that allows for a system exploit

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevate Your Web3 Journey

CertiK is the largest Web3 security platform combining formal verification with audits and comprehensive security solutions.