TON生态火爆，需警惕TonConnect SDK 的Origin伪造风险

By: Thinking

### 背景

随着 TON 生态项目的升温，Web3 钓鱼团伙也开始进入 TON 生态的战场。目前 TON 生态中使用 TonConnect SDK 来解决跨平台/应用钱包连接和交互的问题，这类方案都难免会遇到一个难题：如何解决跨平台/应用通讯时的域名验证？

通常为了能够让用户在使用钱包连接 DApp 或确认签名请求的来源是否可靠时，钱包都会在请求批准页面提示来源的域名，便于用户更好地验证并确认请求的来源是否与自己操作的来源一致，从而避免遭受恶意来源签名请求的欺诈。

慢雾安全团队此前就发现过这类钱包和 DApp 跨平台/应用通讯时的域名验证安全问题，我们跟 MetaMask SDK 和 WalletConnect Web3Modal 的项目方团队进行了沟通和交流，发现该问题较难处理。因此，目前 MetaMask 和 WalletConnect 尚未完全解决这个问题。

近日，我们发现 TON 生态中的 TonConnect SDK 也存在相同的问题，于是在此披露，希望可以帮助用户识别和防范这类风险。

### 分析

通常浏览器扩展钱包和 DApp 进行交互时会在网页上注入 JS 脚本(content script)，用于转发网页和浏览器扩展之间的消息。网页和 content script 进行通讯时使用的是 window.postmessage 和 window.addEventListener，而 window.addEventListener 可以通过获取消息的 origin 来进一步处理消息，具体操作包含在浏览器扩展钱包展示消息的 origin，判断消息的 origin 是否在黑名单内，对消息的 origin 进行鉴权等操作。由于 origin 依赖浏览器提供的函数功能来获取，因此无法被伪造。

然而跨平台/应用消息通讯时，通常通过消息转发服务器对消息进行转发，而消息转发服务器很难对消息发起的域名进行检查（因为客户端的数据可以被伪造），因此存在消息来源被伪造的问题，以下是跨平台/应用消息通讯的 2 个场景：

以 TonConnect SDK 为例，DApp 使用 TonConnect SDK 作为钱包和 DApp 进行消息通讯的工具，需要在接入 TonConnect SDK 的时候配置好 dappMetadata，然而 dappMetadata 的数据是难以验证的，恶意 DApp 可以通过修改 dappMetadata 伪装成可信的网站，从而对用户进行欺诈。

import { SendTransactionRequest, TonConnect, UserRejectsError, WalletInfo, WalletInfoInjected } from '@tonconnect/sdk';import { notification } from 'antd';import { isMobile, openLink } from 'src/utils';

const dappMetadata = { manifestUrl: 'https://x.x.x/tonconnect-manifest.json',};

export const connector = new TonConnect(dappMetadata);

把 manifest.json 设置成以下内容就可以将 origin 伪造成 ton.org：

{ "url": "https://ton.org","name":"FakeandevilDApp", "iconUrl": "https://ton-connect.github.io/demo-dapp/apple-touch-icon.png", "termsOfUseUrl": "https://ton-connect.github.io/demo-dapp/terms-of-use.txt", "privacyPolicyUrl": "https://ton-connect.github.io/demo-dapp/privacy-policy.txt"}

以下是部署上述代码后的 PoC，接下来我们扫描并解析二维码。



TonConnect SDK 通过二维码的方式将 manifestUrl 的数据传递给钱包应用，不像其他 SDK 通过消息转发服务器转发。钱包应用会解析扫描获得的 manifestUrl 的数据。可以发现，我们轻易伪造成任意 DApp 的 origin 与钱包进行了通讯，也就是说，攻击者可以利用这个缺陷伪造知名的 DApp 实施钓鱼和欺诈攻击。

https://app.tonkeeper.com/ton-connect?v=2&id=24e3f2fdbea19fcd4fe03da3bc3d534d8b988edd01736f0b127a70cf2c531661&r={"manifestUrl":"https://tonconnect.pages.dev/tonconnect-manifest.json","items":[{"name":"ton\_addr"}]}



连接成功后，伪造的 DApp 通过 TonConnect 发起签名申请，一旦用户确认了，钱包就会将签名后的数据广播到区块链上。由于 origin 伪造极具欺骗性，用户很难鉴别连接和签名申请的来源。



在 MetaMask SDK 中，通过修改 dappMetadata 即可伪造成知名的 DApp 实施钓鱼和欺诈攻击：

同理，在 WalletConnectModalSign 中，修改 metadata 即可：

import { WalletConnectModalSign } from "https://unpkg.com/@walletconnect/modal-sign-html@2.5.8";

const connectButton = document.getElementById("connect-button");

const web3Modal = new WalletConnectModalSign({ projectId: "32832b2f363c3e960ea28541f380b8d1", metadata: {name:"FakeandevilDApp",description:"FakeandevilDApp", url: "https://walletconnect.com/", icons: ["https://walletconnect.com/static/favicon.png"], },});

### 总结

由于目前主流钱包和 DApp 跨平台/应用通讯时的域名验证问题暂时没有较好的解决方案，因此 SDK 项目方通常会额外增加一些验证方式，如：WalletConnect 的 Verify 机制（https://docs.walletconnect.com/cloud/verify），在 DApp 通过验证域名后，钱包方可以通过 Verify API 来判断域名是否可信。

但是，很多主流的 DApp 没有对域名进行 Verify，所以这个解决方案也很难解决伪造 origin 钓鱼攻击。如果绝大多数 DApp 使用 Verify 的方式对域名进行认证，那么将很大程度上让用户避免遭受日益猖獗的伪造 origin 钓鱼攻击。慢雾安全团队在此也建议用户注意识别打开的网站是否和请求批准展示的域名一致，避免遭受此类攻击。