virus BULLETIN Covering the global threat landscape

INSIDE THE IOS/ADTHIEF MALWARE

Axelle Apvrille
Fortinet, France

Surprisingly (or maybe not), *iOS* malware isn't very common. At the end of 2013, there were only four different families (Ikee, FindCall, Toires and Trapsms) as well as a dozen families of adware or spyware [1]. Ikee and Trapsms require jailbroken devices, whereas FindCall and Toires work on any device.

Thus, the discovery of new *iOS* malware is generally pretty hot news for an anti-virus analyst. In March 2014, Claud Xiao discovered iOS/AdThief, a.k.a. Spad, a piece of malware which hijacks advertisement revenues and redirects them to the attacker.

However, very little information was published at the time, and the little that was published [2, 3] was difficult to understand (even for technical readers). This paper attempts to provide a clear description of the virus. In doing so, it also provides some tips for reversing *iOS* malware, and a few new findings are also disclosed.

THE MALWARE'S GOAL

iOS/AdThief.A!tr works on jailbroken *iOS* devices. It implements a *Cydia Substrate* [4] extension to hijack the revenues from advertisements on the infected device. In other words, each time you view or click an ad on an infected device, the corresponding revenue goes to the attacker, and not to the developer or the legitimate affiliate.

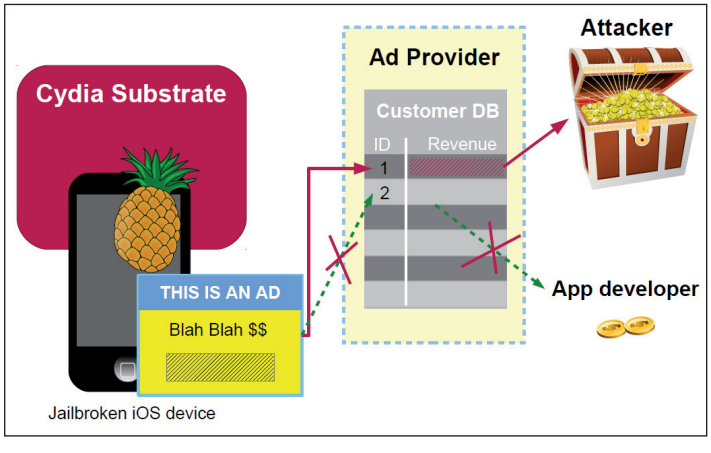

Figure 1: iOS/AdThief!tr hijacks advertisement revenues and redirects them to accounts owned by the attackers.

Cydia Substrate, which only works on jailbroken devices, is a platform for modifying existing processes. It provides an API to hook the legitimate functions, and you can add your own tweaks. This is exactly what the malware does: it hooks various advertisement functions and modifies the developer ID (a.k.a. promotion ID) to match that of the attacker (see Figure 1).

TARGETED ADKITS

A list of mobile adkits targeted by the malware is provided in [2]: *YouMi*, *Vpon*, *MobClick*, *Umeng*, *AdSage/MobiSage*, *MdotM*, *InMobi*, *Domob*, *AdWhirl*, *AdsMogo*, *Google Mobile Ads SDK*, *AderMob*, *Weibo*, *MIX SDK* and *Poly SDK*. The majority of these are Chinese, four are based in the US, and two in India.

In [2], Xiao remarks that *Weibo* is a popular social network in China, but is unable to attribute *MIX SDK* and *Poly SDK* more precisely. In fact, *Sina Weibo*, introduced in 2013, is an advertisement SDK [5], so that solves one mystery.

MIX SDK can be attributed to *GuoHeAD*. It probably refers to the *GuoHe MIX* platform for cross-promotion of mobile games [6]. This is also backed up by the name of a source file found in the malware: /Volumes/MacOsStore/Project/IOS/SpAd/SpAd/AD_GuoHe.xm.

Finally, *Poly SDK* is not a new adkit: it corresponds to *AderMob*. This is confirmed when downloading the *AderMob iOS* SDK [7].

Closer inspection of the malware shows that, as well as the adkits included on the list in [2], it also supports *Komli Mobile*. This hadn't been identified earlier because the hook names were quite generic ('APIManager'). However, when searching for strings in the malware, we noticed that the name of the source file of each adkit hook is shown close to the functions it hooks. For example, the source filename 'AD MobiSage.xm' appears next to hooks to *MobiSageAdBanner* and *MobiSageAdPoster*. Since 'AD KomliMobile.xm' appears next to 'APIManager' hooks, and that name has not yet been attributed to another adkit, we assume that 'APIManager' corresponds to a class name of the *Komli Mobile* SDK.

```
/Volumes/MacOsStore/Project/IOS/SpAd/SpAd/AD_MobiSage.xm
__ZL69_logos_method$_ungrouped$MobiSageAdBanner$initWit
..
__ZL57_logos_method$_ungrouped$MobiSageAdPoster$initWit
..
__ZL69_logos_method$_ungrouped$MobiSageAdPoster$initWit
..
...
```

```
__ZL178_logos_method$_ungrouped$APIManager$for..
/Volumes/MacOsStore/Project/IOS/SpAd/SpAd/AD_KomliMobile.xm
__ZL200_logos_method$_ungrouped$APIManager$for...
```

In total, therefore, the malware hijacks advertisements from 15 different adkits (see Table 1).

AderMob	http://adermob.renren.com/	China
AdMob and Google Mobile Ads	http://www.admob.com/	USA
AdsMogo	http://www.adsmogo.com/en	China
AdSage/MobiSage	http://www.adsage.com/mobiSage	China
AdWhirl	http://www.adwhirl.com	USA
Domob	http://domob.cn	China
GuoHeAD	http://www.guohead.com	China
InMobi	http://www.inmobi.com	India
Komli Mobile	http://www.komlimobile.com/index	India
MdotM	http://www.mdotm.com	USA
MobClick	http://www.mobclix.com	USA
UMeng	http://www.umeng.com	China
Vpon	http://vpon.com	China
Weibo	http://us.weibo.com	China
YouMi	http://www.youmi.net	China

Table 1: Hijacked advertisements in iOS/AdThief.

MALWARE INTELLIGENCE

The malware author forgot to strip out some debugging information – which is helpful (for us) for identifying the adkits it targets via their source filenames. It is also helpful for identifying the malware author.

Indeed, the strings inside the malware show the following path: /Users/Rover12421/Library/Developer/Xcode/DerivedData/SpAd-gfwvssdhvecvkrggjdyjwgdpkbqd/Build/Intermediates/SpAd.build/Release-iphoneos/SpAd.build/Objects-normal/armv7/SpAd.o.

Adkit source	Filename	Typical class names
AderMob	AD Ader.xm	AderSDK*
AdMob and Google Mobile Ads SDK	AD AdMob.xm	GAD*
AdsMogo	AD AdsMongo.xm	AdMoGo*
AdSage	?	MobiSageAd*
AdWhirl	AD Adwhirl.xm	AdWhirl*
Domob	AD DoMob.xm	DM*
GuoHeAD	AD GuoHe.xm	MIXView*
InMobi	AD InMobi.xm	IMAd*
Komli Mobile	AD KomliMobile.xm	APIManager*
MdotM	AD MDotM.xm	MdotM*
MobClick	?	MobClick*
UMeng	AD UMeng.xm	UMUFP*
Vpon	AD Vpon.xm	VponAdOn*
Weibo	AD Weibo.xml	DXAdHWB*_
YouMi	AD Youmi.xm	YouMi* – delegated to Google Ads

Table 2: Implementation details of adkit hooks found in iOS/AdThief.A!tr.

We can assume that 'Rover12421' is the malware author. He maintains a blog (http://www.rover12421.com/) and works on *Android* hacks (some of which are hosted at https://github.com/rover12421). He also has a *Twitter* account (@rover12421), although that is fairly inactive. On *pediy.com* forums, he is known as 'zerofile'. He actually admits to having written at least some parts of the code found in the malware. He says he worked on 'spad' a long time ago, that it was his only *iOS* project, and that it is now 'closed' (see

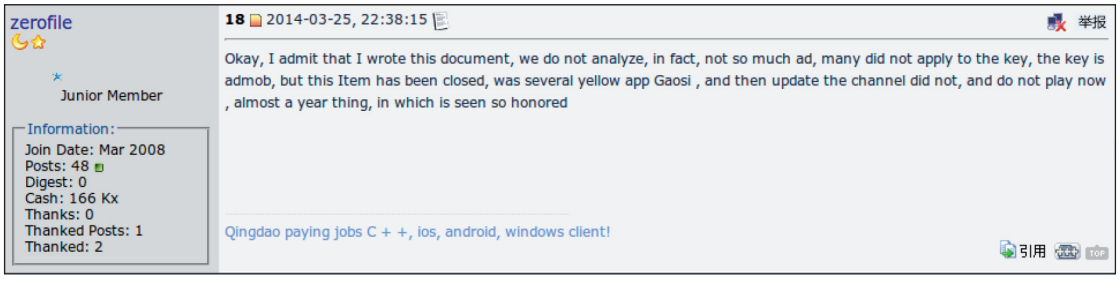

Figure 2: On a pediy.com forum, Rover12421, a.k.a. zerofile, admits to being the creator of the malware (translated from Chinese).

Figure 2). His answers aren't very clear, but it seems he only wrote a basic ad ID replacement plug-in, and that someone else improved the code (see Figure 3). He denies having participated in the propagation of the malware.

According to [3], the malware is estimated to have infected 75,000 devices, stealing revenue from around 22 million ads.

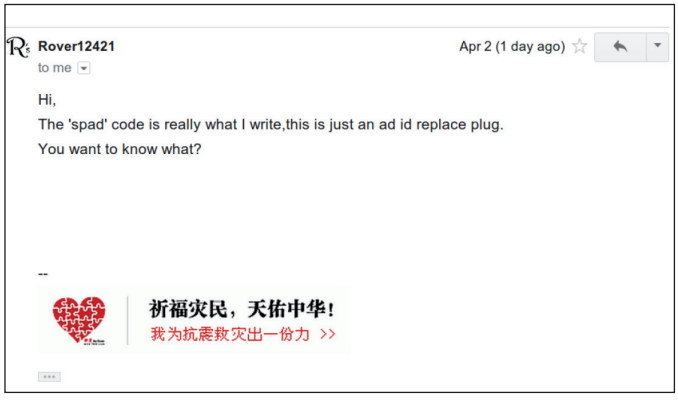

Figure 3: Private email conversation with Rover12421.

IMPLEMENTATION

Overview

The main parts of iOS/AdThief's implementation are detailed in [2]. However, Xiao's post requires some familiarity with *iOS*, *Substrate* and Objective C. Here, we attempt to provide a clearer description of the implementation.

Each time an end-user views or clicks on a given advertisement, the corresponding application developer (or partner, or affiliate) receives a small payment.

This is what advertisement companies refer to as 'cost per thousand impressions' (CPM) or 'click-through rate' (CTR). To credit the right developer when ads are viewed or clicked, adkits identify developers (or partners etc.) with a developer ID.

iOS/AdThief modifies this developer ID, replacing it with an identifier owned by the attacker. Revenues are consequently hijacked, with all of the revenue generated when an ad is viewed or clicked being assigned to the attacker's identifier.

To modify the developer ID, the malware author implements a hook for each of the adkits he wants to hijack, where he replaces the developer ID as he wishes. To do so, he takes advantage of an existing process-hooking platform, *Substrate*, which is available on jailbroken devices. To implement a *Substrate* hook, one uses a function named 'MSHookMessageEx' with four arguments:

1. The class to hook.
2. A 'selector', i.e. the name of the function to hook.
3. The address of the replacement function, i.e. the hook.
4. A stub to call the old replaced function in case it is needed. (It may be left as NULL if not needed.)

Registering a hook

For example, in the code shown in Figure 4, the author creates a YouMiView class object (line 1). Then he calls objc_msgSend on that object (line 6).

In Objective C, one does not 'call' a function, but instead 'send a message to a function' [8]. Consequently, each function call consists of a call to objc_msgSend (which is why objc_msgSend appears so frequently).

So, the author sends a message to 'instancesRespondToSelector:' with the argument 'adViewWithContentSizeIdentifier:delegate:'. For those more familiar with C or Java, this translates to:

```
obj->instancesRespondToSelector("adViewWithContentSize
Identifier:delegate:")
```

InstancesRespondToSelector [9] is a standard *iOS* function that tests 'whether instances of the receiver are capable of responding to a given selector', i.e.

```
1  v0 = objc_getClass("YouMiView");
2  v1 = (void *)v0;
3  v2 = (void *)object_getClass(v0);
4  v3 = v2;
5  if ( v2
6      && (unsigned int)objc_msgSend(v2, "instancesRespondToSelector:",
7   "adViewWithContentSizeIdentifier:delegate:") & 0xFF )
8     MSHookMessageEx(
9       v3,
10      "adViewWithContentSizeIdentifier:delegate:",
11      _logos_meta_method__ungrouped_YouMiView_...
12                  ... adViewWithContentSizeIdentifier_delegate_,
13      &unk_7814);
```

Figure 4: Decompiled function setting a hook for the YouMi adkit.

```
1  void *__fastcall _logos_meta_method__ungrouped_YouMiView_
2    adViewWithContentSizeIdentifier_
3    delegate_(int a1, int a2, int a3, int a4)
4  {
5    int v4; // r4@1
6    int v5; // r0@1
7    void *v6; // r0@1
8    void *v7; // r5@1
9
10   v4 = a4;
11   v5 = getClass_GADBannerView_YouMi();
12   v6 = objc_msgSend((void *)v5, "alloc");
13   v7 = objc_msgSend(v6, "initWithAdSize:origin:");
14   objc_msgSend(v7, "setAdUnitID:", CFSTR("a1521215ab55cd2"));
15   objc_msgSend(v7, "setRootViewController:", v4);
16   objc_msgSend(v7, "setDelegate:", v4);
17   return v7;
18 }
```

Figure 5: Decompiled YouMi hook function.

```
1  int __fastcall getClass_GADBannerView_YouMi()
2  {
3    int v0; // r4@1
4
5    v0 = objc_getClass("GADBannerView_YouMi");
6    if ( !v0 )
7    {
8      v0 = 0;
9      if ( getClass_formAdMob((int)"GADBannerView") )
10     {
11       v0 = objc_allocateClassPair();
12       class_addMethod(v0, "start", 18373, "v@:");
13       class_addMethod(v0, "setContentSizeIdentifier:", fn_void_id, "v@:@");
```

Figure 6: Decompiled code of function getClass GADBannerView YouMi().

whether the instance implements a function named 'adViewWithContentSizeIdentifier:delegate:'.

If such a function exists, the malware author decides to hook it (lines 6–7). He calls MSHookMessageEx (line 8), says he wants to hook adViewWithContentSizeIdentifier: delegate:, and that his implementation of the hook will be in _logos_meta_method_ungrouped_YouMiView_ adViewWithContentSizeIdentifier_delegate_ (lines 11–12).

Implementing a hook

Continuing the previous example, the implementation of the *YouMi* hook is in a function named logos_meta_method_ ungrouped_YouMiView_adViewWithContent_SizeIdentifier_ delegate_. Its code is shown in Figure 5.

At line 11, the malware author instantiates a 'GADBannerView YouMi' object. GAD stands for *Google Ads*, and indeed if we inspect the code of this function, we see that it corresponds to the *AdMob* kit, not *YouMi*. Furthermore, online documentation for GADBannerView exists in the *AdMob* SDK [10]. Next, the author initializes the banner view by sending the object a message 'initWithAdSize:origin:'.

The important part lies in line 14. Onto this *Google Ads* banner, the author sets the 'unit identifier' (*AdMob* Publisher ID) to his preferred ID (a1521215ab55cd2 in this particular case). Then, he sets the root window (provided as the fourth argument – a4) and that same window as a delegate (line 16). Delegation is a mechanism by which a given object can act on behalf of another one, and coordinate with it. This explains the relationship between *Google Ads* and *YouMi*: in this particular case, the author does not directly hook *YouMi* ads, but *Google* ads, which act under the delegation of *YouMi*.

Consolidated data

We have described the details of *YouMi* revenue hijacking. The hijacking of other adkits is achieved using the same strategy:

- registering one or multiple hooks for the adkit
- implementing each hook.

Like *YouMi*, three other adkit hooks work when *Google Ads* acts under delegation: *DoMob*, *Umeng* and *Weibo*. The other adkits work on their own – without delegation.

CONCLUSION

iOS/AdThief is a technical and malicious piece of code which hijacks revenue from 15 different adkits. It is built on top of the *Cydia Substrate* platform, available for jailbroken devices, which provides it with an easy way to modify advertisement SDKs. With *Substrate*, the malware needs only to focus on the call and implementation of each hook.

At first, the identification of every adkit the malware targets was difficult because the code mentions only class names used by each adkit SDK. However, the fact that the malware author did not strip out debugging information helped us to identify all 15 adkits. In particular, this is how support for *Komli Mobile* and *GuoHeAD* was detected.

The debugging information also helped us to track down the malware's creator, a Chinese hacker specializing in mobile platforms. The hacker claims to have written parts of the code some time ago, but that a third party then improved it. He denies having participated in the spreading of the malware. With 75,000 infected devices, iOS/AdThief is not extremely prevalent. However, there are an estimated 22 million hijacked ads, so the malware has probably had a fair amount of impact and generated significant revenue for the owner(s).

ACKNOWLEDGEMENTS

I would like to thank Claud Xiao, Tim Strazzere, 'baronpan7' and Dong Xie for their help with this research.

REFERENCES

[1] FortiGuard Labs. Mobile threats. http://www.fortiguard.com/antivirus/mobile_threats.html.

[2] Xiao, C. New iOS malware use Cydia Substrate to steal advertisement promotion fee, March 2014. http://www.claudxiao.net/2014/03/ios_malware_spad/.

[3] [original] ios app simple analysis of malicious, March 2014. http://bbs.pediy.com/showthread.php?p=1270415.

[4] Cydia Substrate. http://www.cydiasubstrate.com/.

[5] Ong, J. Sina Weibo introduces Twitter-like in-stream advertising in quest to monetize its 400m user base. http://thenextweb.com/asia/2013/01/18/sina-weibo-activates-in-stream-advertising-in-quest-to-monetize-microblog/.

[6] Lu, G. GuoheAD Introduces Guohe MIX, Cross-Promotion Platform for Mobile Games. http://technode.com/2012/08/11/guohead-introduces-guohe-mix-cross-promotion-platform-for-mobile-games/.

[7] Ader. http://adermob.renren.com/s/download.html.

[8] Shub-Nigurrath. Primer on Reversing Jailbroken iPhone Native Applications v1.0, May 2008. http://tutorials.accessroot.com/arteam/site/download.php?view.222.

[9] Apple. Nsobject class reference. https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSObject.

[10] GADBannerView Class Reference. http://cocoadocs.org/docsets/AdMob/6.5.0/Classes/GADBannerView.html.

Editor: Martijn Grooten

Chief of Operations: John Hawes

Security Test Engineers: Scott James, Tony Oliveira

Sales Executive: Allison Sketchley

Editorial Assistant: Helen Martin

Perl Developer: Tom Gracey

Consultant Technical Editors: Dr Morton Swimmer, Ian Whalley

Tel: +44 (0)1235 555139. Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com

Web: http://www.virusbtn.com/