Mactans: Injecting Malware into iOS Devices via Malicious Chargers

Billy Lau

Yeongjin Jang

Chengyu Song

Agenda

- iOS Security
- Mactans
- Discussion

an overview of

IOS SECURITY

Apple App Store

- The walled garden model
  - Acts as platform to publish apps
    - The only place to purchase/download apps
  - Completely controlled by Apple
    - All apps must be reviewed by Apple before release
    - A released app can be removed from the store if it violates policy

Code Signing in iOS

- Enforces the integrity of the boot chain and walled garden model
  - Only correctly signed apps can be installed and executed
- Signing Entities
  - Apple App Store
  - iOS developers

App Review

- Attempts to determine whether the submitted app complies with the rules
- What are the rules?
  - Largely empirical
    - Apps that make use of private APIs are rejected and banned
  - Changing regularly
- What happens during app review?
  - Static analysis and some manual testing (we think)

iOS Sandbox

- Process isolation
  - A sandboxed process cannot read other processes' memory
  - Also cannot talk to other processes using traditional IPC-like APIs
- Filesystem isolation
  - Sandboxed app can only read/write to its own filesystem
    - Can also read (but not write to) some public files
- Entitlement check
  - For some operations (e.g., change passcode), iOS enforces app Entitlements

Walled Garden Effectiveness

- The walled garden model is assumed to be secure
  - All apps are carefully vetted prior to release and thus safe
    - Right?
- Compared to Android, almost no in-the-wild malware instances for iOS

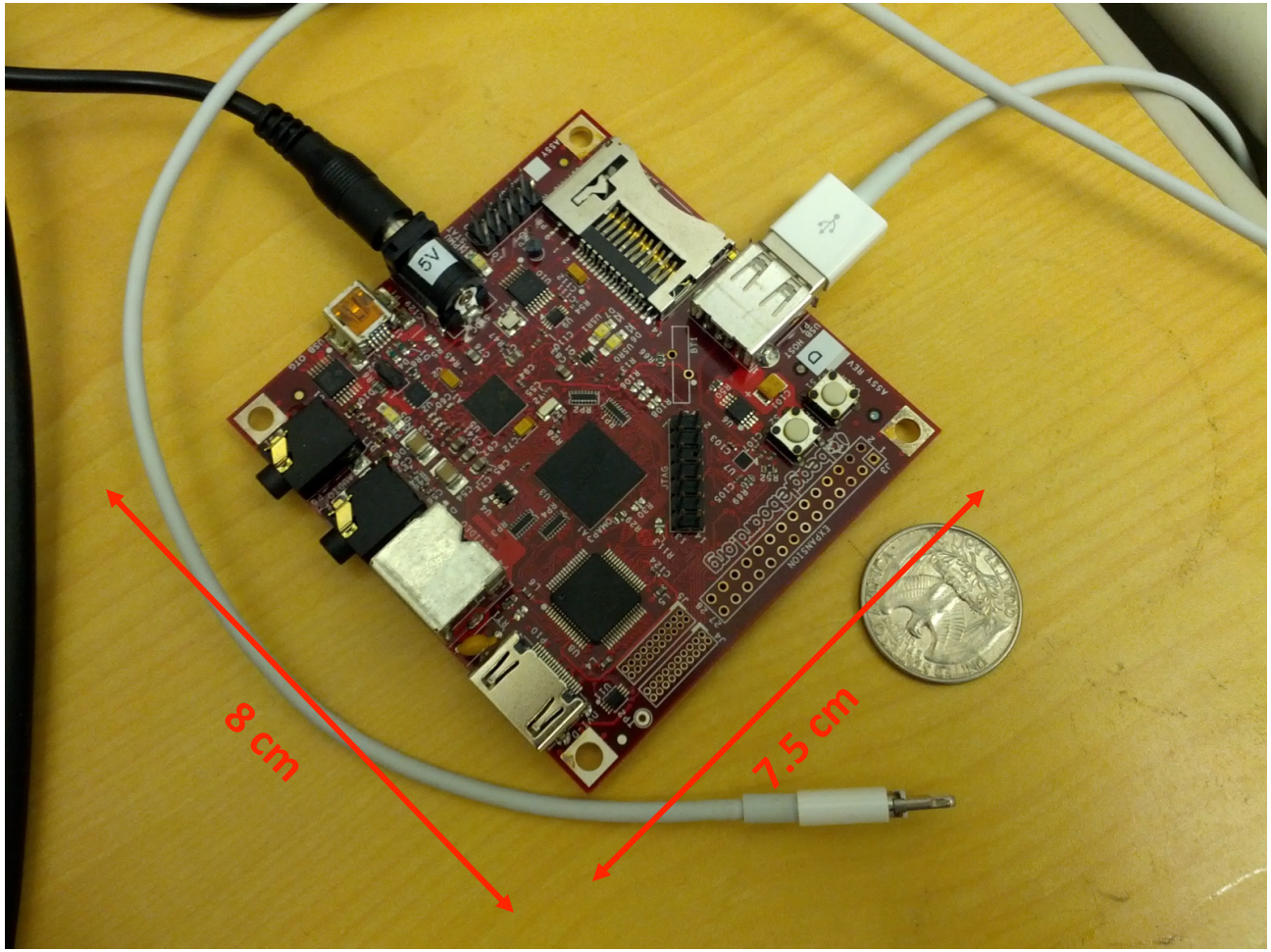
a step-by-step introduction to

MACTANS

Mactans Concept

- Not a jailbreak
  - Does not require a jailbroken device
- Automatic
  - Simply connecting the device is enough
- Stealthy
  - There are no visible clues
- Powerful
  - Does malicious things other apps cannot do

Anatomy of a Mactans Charger



Form Factor Alternatives

- Could be much smaller…

Mactans Overview

1. Obtain device UDID
2. Pair with device
3. Generate and install provisioning profile
4. Install malicious app

Universal Device Identifier (UDID)

- A 40 digit hexadecimal identifier unique to a device
- Obtaining device UDID is trivial via USB connection

Pair With Device

- Once an iOS device is connected via USB, Mactans will try to pair with it
- Mactans leverages a conceptual iOS pairing trust assumption
  - Device cannot reject pairing request
  - Device can be paired without user's consent while it is passcode-unlocked
    - Pairing can occur if device is unlocked at any time (even briefly)
  - Once paired, exploitation is possible regardless of whether or not device is locked

Pair With Device Cont'd

- Many operations can be performed via USB
  - Obtain device information (e.g., UDID, serial number)
  - Install and remove apps and provisioning profiles
  - Backup and restore, firmware reset (ipsw)
  - Debugging
- Mactans can be used to perform these functions

Provisioning Profile Details

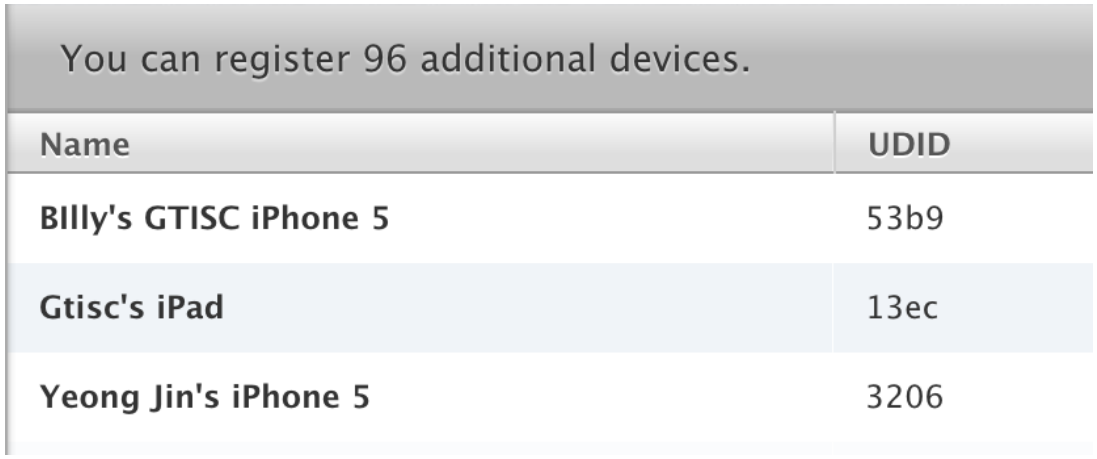
- Types of provisioning profiles
  - Individual
  - Enterprise
- Requirements for Individual profile
  - Active developer's license
  - Device UDID
  - Internet connection

Provisioning Profile Details Cont'd

- Allows devices to run apps signed by a non-Apple entity
  - Provisioning profile must be signed by Apple
    - For enterprises to distribute in-house apps
    - For individual developers to perform beta testing
  - Provisioning profile must match device and app

Provisioning Profile Details

- A device must be registered to run a developer's app
  - Individual developer license allows up to 100 devices
    - Cannot remove devices once registered
  - UDID registration via `developer.apple.com`



Generating a Provisioning Profile

- A Mactans charger must add a UDID to a provisioning profile over the Internet
  - How?
- Use available Internet connection
  - A Mactans charger has a built-in Wi-Fi antenna
  - Can also be equipped with SIM card module for cellular data connection
- Creation via Apple's website is fully automatable
  - Submit UDID, check for and receive generated profile

Generating a Provisioning Profile

Adding devices to team provisioning profile

46

21

I need to add a device to my team provisioning profile, however I do not physically have the device so I can't hook it up to my computer so Xcode can't add the UDID to my devices and to the team provisioning profile. Is there a way to add it manually to the team provisioning profile, I can't figure out how to edit it. Also when I add the device in my provisioning portal it doesn't get added to my team provisioning profile automatically.

iphone provisioning

1

Per May 16th 2013, using XCode 4.6.2, I had to do the following to add a device (which I do not have physical access to) to the team provisioning profile:

1. Login to the provisioning portal through developer.apple.com
2. Add the UDID in Devices
3. Select the Team Provisioning profile in Provisioning Profiles
4. Click the Edit button
5. And under devices for that provisioning profile, click Select All, or just the devices you want included.
6. Click Generate

- Can be easily automated by browser automation tools
  - No CAPTCHA

Installing an App

- Once obtained, a provisioning profile can be installed without user's consent (or knowledge)
  - Apps owned by provisioning profile owner can then be installed via USB
- After profile installation, arbitrary apps can be installed and executed
- Next steps
  - Hide app to prevent unwanted deletion
  - Circumvent app runtime restrictions (i.e., via misuse of private APIs)

Hiding an App

- There are some hidden apps on the stock iPhone
  - /Applications/DemoApp.app
  - /Applications/FieldTest.app
- Info.plist for these apps reveals a common field

```
<key>SBAppTags</key>
<array>
    <string>hidden</string>
</array>
```

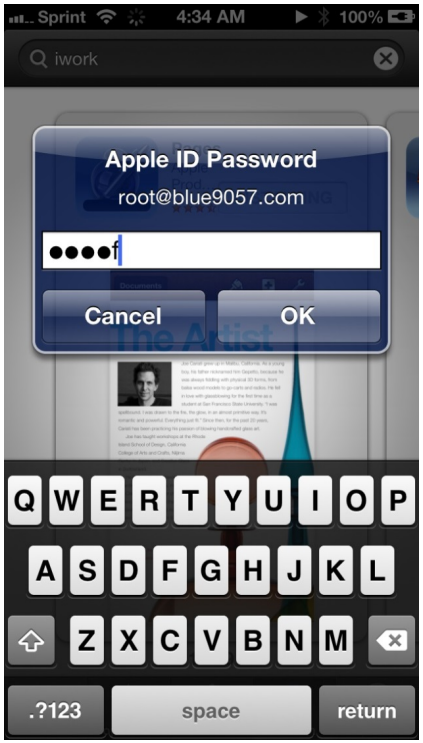
- This property hides the app on the main screen and in the task manager

Hidden App Capabilities

- iOS background execution
  - App can run without user's knowledge
  - iOS limits background execution to 10 minutes
  - Limit can be extended by several methods
    - Terminate and restart before 10 minute deadline
    - Register as VoIP app and setKeepAliveTimeout:600
  - With these methods, app can effectively run indefinitely

Hidden App Capabilities Cont'd

- Example: Taking screen shots
  - Using a Private API call, a background app can take a screenshot of current 'foreground' screen




```
+ (UIImage *)captureCurrentDisplayAsImage
{
    void* surface = [UIWindow createScreenIOSurface];
    UIImage *surfaceImage = [[UIImage alloc] _initWithIOSurface:surface
         scale:[UIScreen mainScreen].scale orientation:UIImageOrientationUp];
    CFRelease(surface);
    return surfaceImage;
}
```

Georgia Institute of Technology

Hidden App Capabilities Cont'd

- Example: Simulating screen/button presses
  - Xcode instrumentation
    - App testing can be automated
  - Simulation can also be done outside Xcode
  - DeveloperDisk
    - Has UIAutomation.framework
    - Try dlopen(), call APIs there

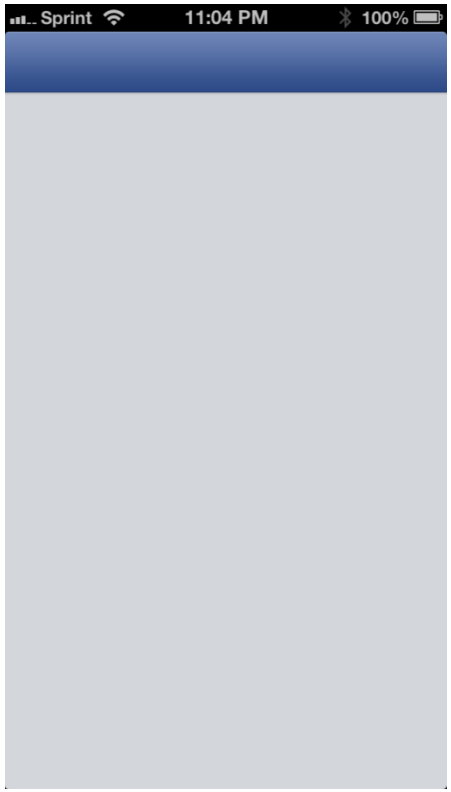
```
- (void)clickMenu;
- (void)holdMenu:(double)arg1;
- (void)lockDevice;
- (void)clickLock;
- (void)holdLock:(double)arg1;
- (void)clickVolumeUp;
- (void)holdVolumeUp:(double)arg1;
- (void)clickVolumeDown;
- (void)holdVolumeDown:(double)arg1;
- (void)setRinger:(BOOL)arg1;
- (void)shake;
- (void)touchDown:(struct CGPoint)arg1;
- (void)liftUp:(struct CGPoint)arg1;
- (void)_moveLastTouchPoint:(struct CGPoint)arg1;
- (void)sendTap:(struct CGPoint)arg1;
- (void)sendDoubleTap:(struct CGPoint)arg1;
- (void)sendDoubleFingerTap:(struct CGPoint)arg1;
```

iOS Trojan Horse

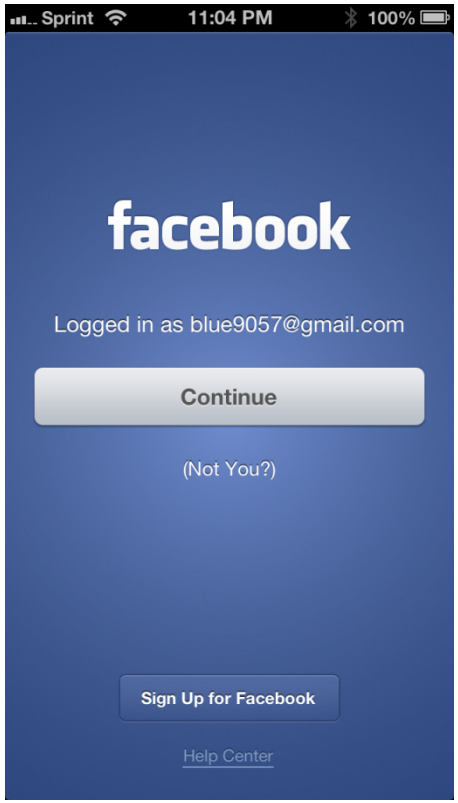
- Surreptitiously replace existing app with Trojan
  - Obtain a set of original apps (Facebook, Skype)
  - Repackage apps with Info.plist that has SBAppTags/ hidden property
  - Sign app and Info.plist with attacker-owned developer key and load onto Mactans charger
  - After pairing
    - Replace original app with repackaged, hidden version
    - Install new, malicious app with icon of replaced app
    - When launched, new app performs malicious actions, then executes repackaged (hidden) app

Trojan Horse Workflow

Main Screen Shows Trojan



User Launches Trojan



Trojan Launches Real, Hidden App

Attack Scenarios

- General
  - Use enterprise provisioning profile to setup public charging stations (e.g., at airports, libraries)
- Targeted
  - Exchange or provide charger to target
  - Use a priori knowledge to selectively modify environment (e.g., specific airplane seat, hotel room)

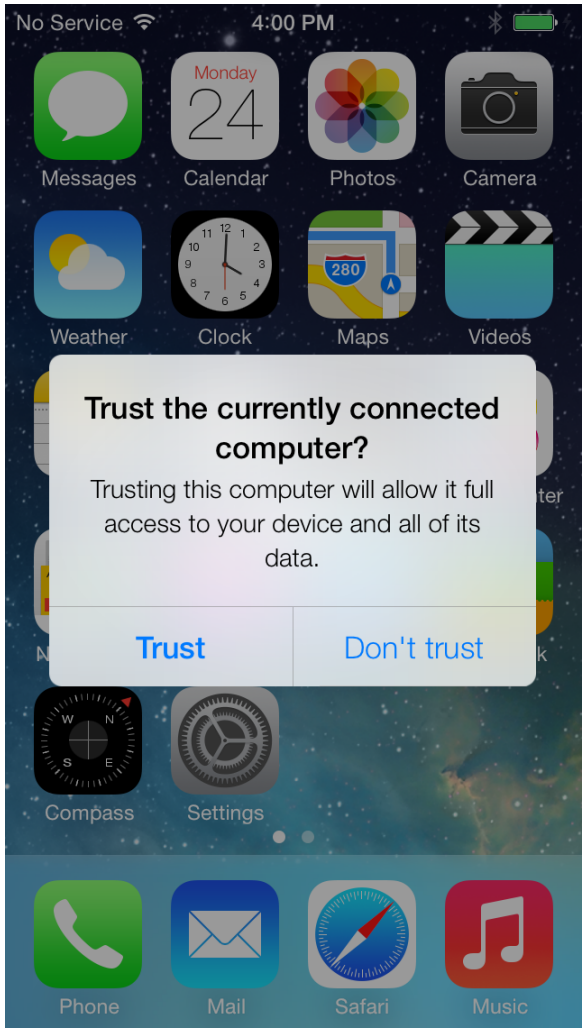
DISCUSSION

Problem #1

- Incorrect trust model for pairing
  - Any host is implicitly trusted if the phone is not passcode protected
  - Once pairing is established, it is permanent

Fix for Problem #1

- Use explicit authorization
  - Coming to iOS 7
- Trusted host management
  - Synonymous with Wi-Fi management



Problem #2

- No visual cues to differentiate a charger versus a computing device
  - iOS only has an indicator for synchronization, and only shows that indicator during synchronization

Fix for Problem #2

- Visual indicator to differentiate charge mode and pair mode
  - Fix for Problem #1 also fixes this problem
  - Android generates a notification when the phone is connected to a host and always shows the indicator

Problem #3

- Provisioning profile abuse
  - Apple pays lots of attention to app signing, but little attention to provisioning profile signing

Fix for Problem #3

- Add procedures to prevent provisioning profile generation
  - Use CAPTCHA
  - Implement mechanisms to detect suspicious developer activity

Problem #4

- Over-privileged default capabilities for USB
  - Obtain device information (e.g., UDID, serial number)
  - Install and remove apps and provisioning profiles
  - Backup and restore, firmware reset (ipsw)
  - Debugging

Fix for Problem #4

- Tighten default USB connection settings
  - Reduce default connection mode privileges
  - Require explicit authorization for provisioning profile installation

Problem #5

- Third party hidden apps considered harmful
  - Few or no legitimate uses
  - High abuse potential

Fix for Problem #5

- Restrict the ability to set hidden property
  - Only allow apps developed by Apple to use this property

One more thing …

- You do not need a malicious charger to bypass the protections of the walled garden model
  - Jekyll on iOS: When Benign Apps Become Evil. Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee, Georgia Institute of Technology.
  - To appear in proceedings of the 2013 USENIX Security Conference, August 14-16, 2013.

Please fill out your feedback forms.

Questions?