VIKRAM SRI NITESH TANTRAVAHI
13247361

Reverse Engineering iOS Mobile Application

College of Computing & Informatics
Drexel University

Research Paper

1. ABSTRACT

Mobile has become a vital component of our life. Sometimes it is a good factor and at times bad a bad one. With increasing technology and demands of users in the mobile world, the need of some software engineering techniques or some tools to engineer the software are very much required as a supportive base for the programmers and analysis tasks.

In this paper, we propose and compare few reverse engineering techniques like *iCRAWLER*, *HOPPER*, *iOS-RET* (Reverse Engineering Toolkit) and so on. The main functionality of these techniques or tools is to (1) dynamically run and analyze a given specific iOS application, (2) utilize and exercise its available use interface, cover the interaction state space and eventually extract information regarding the runtime behavior, and (3) a state model is then created, where the user interface states and transitions between them are captured. We finally approach and make a capable technique that automatically detects the unique states and thereby generates correct state model of a given mobile application.

2. INTRODUCTION

The demand for latest and upgraded technology has been in high demand in recent times and this has been the driving force for all the top-heir companies to strive forward for developing novel software engineering techniques. The tools that are developed relative to mobile platforms are the most vital components that act as supporting supplements to the developers for the study and analysis. This paper mainly demonstrates many of the techniques attackers use to manipulate iOS applications in order to extract confidential data from the device. Most of the time attacking iOS application is synonym to jailbreak an iDevice, decrypt the application and reverse the binaries. Before developing these items there is some interesting points to linger on, especially on regular devices.

Q. What attackers can do without jailbreaking the device?
A. Without having access to the file system it is impossible to decrypt and reverse iOS applications installed from Apple App Store. Nevertheless, this section present attacks vectors that can allow retrieving confidential information stored by miss implemented iOS application.

Q. Installing the application on a jailbroken device?

A. Apple designed the iPhone platform with the intent to control all software that is executed on the device. Thus, the design does not intend to give full system (or root) access to a user.

Moreover, only signed binaries can be executed. In others words, the loader will not execute either unsigned binaries or signed binary without a valid signature from Apple. This ensures that only unmodified Apple-approved applications are executed on the device.

An evaluation of the technique through a case study conducted on six different open-source iPhone applications. The results of our empirical evaluation show that ICRAWLER is able to identify the unique states of a given iPhone application and generate its state model accurately, within the supported transitional UI elements.

3. ATTACKING iOS APPLICATION

Most of the time attacking iOS application is synonym to jailbreak an iDevice, decrypt the application and reverse the binaries. Before developing these items there is some interesting points to linger on, especially on regular devices.

3.1 Using AFC protocol to retrieve data stored on the device.

AFC Clients like iTunes are allowed access to a "jailed" or limited area of the device memory. Actually, AFC clients can only access to certain files, namely those located in the Media and User installed applications folders. In other words, using AFC client a user/attacker can download the application resources and data. Including the default preferences file where sometimes credentials are stored. The only requirement is the device has to be unlocked. But this is definitively not a problem because an evil maid cans backdoor any iDevice Dock Station.

Figure 1 iPown Dock: Malicious dock station.

3.2 Data Retrieval from backups

The main function of the backup is to permit user to restore personal data and settings to an iPhone during a Restore (during which the content on the iPhone is typically erased).

There are three tables, which are the technical results of this attacking application process:

TABLE I. BACKUPS FILE PATH

System	Backup Path
Windows 7	C:\Users\(username)\AppData\Roaming\Apple Computer\MobileSync\Backup\
Mac OSX	/Users/(username)/Library/Application Support/MobileSync/Backup/

TABLE II. KEYCHAIN CLASS KEYS

Protection class	Description
kSecAttrAccessibleWhenUnlocked	Keychain item is accessible only after the device is unlocked
kSecAttrAccessibleAfterFirstUnlock	Keychain item is accessible only after the first unlock of the device to till reboot
kSecAttrAccessibleAlways	Keychain item is accessible even the device is locked
kSecAttrAccessibleWhenUnlocked ThisDeviceOnly	Keychain item is accessible only after the device is unlocked and the item cannot be migrated between devices
kSecAttrAccessibleAfterFirstUnlock ThisDeviceOnly	Keychain item is accessible after the first unlock of the device and the item cannot be migrated
kSecAttrAccessibleAlways ThisDeviceOnly	Keychain item is accessible even the device is locked and the item cannot be migrated

TABLE III. PROTECTION CLASSES FOR BUILT IN ITEM

Application & Item type	Protection class
WiFi Password	Always
IMAP/POP/SMTP accounts	AfterFirstUnlock
Exchange Accounts	Always
VPN	Always
LDAP/CalDAV/CardDAV Accounts	Always
iTunes backup password	WhenUnlocked ThisDeviceOnly
Device Certificate & private Key	AlwaysThisDeviceOnly

The purpose of functional specification is to specify the complete functional behavior of the software in all possible circumstances of use and to obtain agreement with the customer on the specified function as the basis for software development and certification. The specification team creates the Functional Specification document to satisfy the software requirements.

The real time examples of cracking applications are very intelligent and most productive means of applying the reverse engineering techniques. One of the major parts is the "crackulous" software below.

Figure 5 Cracking application with crackulous.

The black box gives an external description of the external behavior of a system or system part in terms of a mathematical function.

4. OPERATION OF iCRAWLER

There exists a distinctive relation between the techniques we are using and the selected mobile application.

Step 1 - Hooking into the Application: As soon as the application is started, our technique kicks in by setting up a shared instance object. As shown in Figure 3, we immediately hook into and monitor the application delegate object to identify the initial view controller and infer information about its UI components.

Step 2 - Analyzing UI Elements: After obtaining the ini- tial view controller, we have access to all its UI ele- ments. We keep this information in an array associated to the view controller. Meanwhile, our technique recog- nizes the different types of UI elements, such as labels, buttons, table cells, and tabs, and identifies which UI elements have an event listener assigned to them.

Step 3 - Exercising UI Elements: To exercise a UI ele- ment, we look for an unvisited UI element that has an event listener. As depicted in Figure 3, after gathering all the information about the event listeners, the UI object, and its action and target, we generate an event on that element and pass it to UIApplication object, which is responsible for receiving the events and dispatching them to the code for further handling.

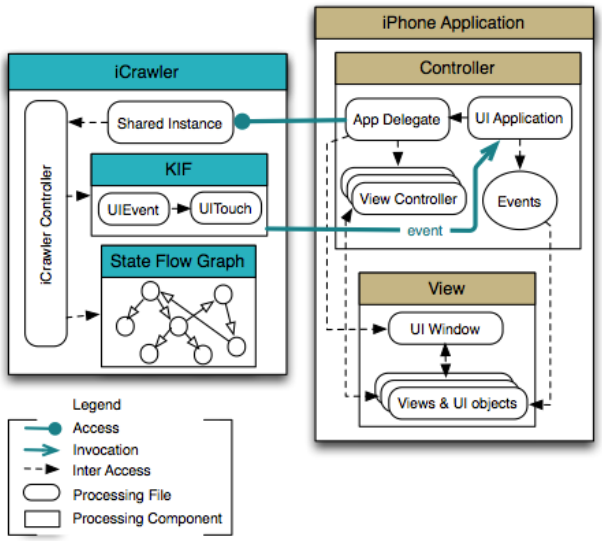
Step 4 - Accessing Next View Controller: By observing the changes on the current view controller, we obtain the next view controller and analyze the behaviour. The event could lead to four scenarios: no user interface change, the changes are within the current view con- troller, going to a new view controller, or going to the previous view controller.

Figure 3: Relation between ICRAWLER and a given iPhone application. The right side of the graph shows key components of an iPhone application taken from [17].

Step 5 - Analyzing New UI Elements: After new view controller, we collect all its UI elements. If the action has resulted in staying in the current view controller, we record the changes on the UI elements.

Step 6 - Comparing UI States: Once we get the new view controller and its UI elements, we need to com- pare the new state with all the previously visited unique states. This way, we can determine if the action changes the current state or ends up on a state that has already been analyzed. If the state is not visited before, it is added to the set of unique visited states.

Step 7 - Recursive Call: We recursively repeat from step 3 until no other executable UI elements are left within the view controller and we have traversed all the view controllers.



Some of the methods that are being injected in the process for the implementation are:

```
- (void) icDismissModalVC:(BOOL) animated {
    [[NSUserDefaults standardUserDefaults] setBool:YES ↩
        forKey:@"IC_isDismissed"];
    // Call the original (now renamed) method
    [self icDismissModalVC:animated];
}
```

```
+ (void)load {
  if (self == [UIViewController class]) {
    Method originalMethod = class_getInstanceMethod(↩
        self, @selector(↩
        dismissModalViewControllerAnimated:));

    Method replacedMethod = class_getInstanceMethod(↩
        self, @selector(icDismissModalVC:));

    swap(originalMethod, replacedMethod);
  }
}
```

5. CONCLUSION

Developers should apply the following recommendation in order to mitigate the risks.

- Do not relay only on iOS security,

- Do not store credential using standardUserDefaults method.

- Encrypt your data even when stored in the keychain.

- Do not store crypto keys on the device.

- Check your code, classes, functions, methods integrity,

- Detect the jailbreak,

- Properly implement cryptography in applications (simple implementation are the most secure),

- Remove all debug information from the final release,

- Minimize use of Objective-C for critical functions & security features.

Users and companies should not blindly thrust iOS application vendors when talking about security.

6. REFERENCES

[1] *Berg Insight, "The mobile application market,"* http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf.

[2] "App Store Metrics," http://148apps.biz/app-store-metrics/.

[3] H. Kim, B. Choi, and W. Wong, "Performance testing of mo- bile applications at the unit test level," in *Proceedings of the 3rd International Conference on Secure Software Integration and Reliability Improvement*. IEEE Computer Society, 2009, pp. 171–180.

[4] "UI/Application Exerciser Monkey," 2010, http://developer.android.com/guide/developing/tools/monkey.html.

[5] Mathieu_Renard_Practical_iOS_Apps_hacking