University of The Aegean
Department of Information & Communication Systems Engineering
Samos, Greece

Chung-Ang University
Department of Computer Science and Engineering
Seoul, South Korea

Lifting the veil on mobile malware: A complete dynamic solution for iOS

Dimitrios Damopoulos, Georgios Kambourakis, Stefanos Gritzalis, Sang Oh Park

{ ddamop, gkamb, manag, sgritz }@aegean.gr
{ sopark3}@gmail.com

Contents

CONTENTS

Introduction: Mobile Evolution and Threats

- Mobile devices have experienced a rapid shift from pure telecommunication devices to small and ubiquitous computing platforms
- They run sophisticated OSs that need to confront almost the same risks as desktop computers do
- Therefore, smartphones represent a promising target for any malware developer
  - over 1 million Android smartphones have been affected by some malware
  - 33.9% of the free iOS applications intent to leak private user information
- Our proposal: iOS Dynamic Malware Analyzer (iDMA)
  - an automated malware analyzer and detector for the iOS platform

CONTENTS

Preliminaries

- Malware Analysis techniques
  - **Static analysis** is conducted without executing the software
    - ✓ analyze the source code of the software sample on the corresponding binary file using reverse-engineering techniques
  - **Dynamic analysis** is conducted while executing the software sample on the host device
    - ✓ monitoring calls to API methods
    - ✓ monitoring the input passed and returned from method invocation(s)
- Malware Detection techniques
  - **Signature-based**, utilize pre-defined set of signatures which represent malicious code
  - **Behavior-based**, employ machine learning algorithms to learn from behavioral profiles in a way that potentially unauthorized actions and malicious profile patterns can be detected

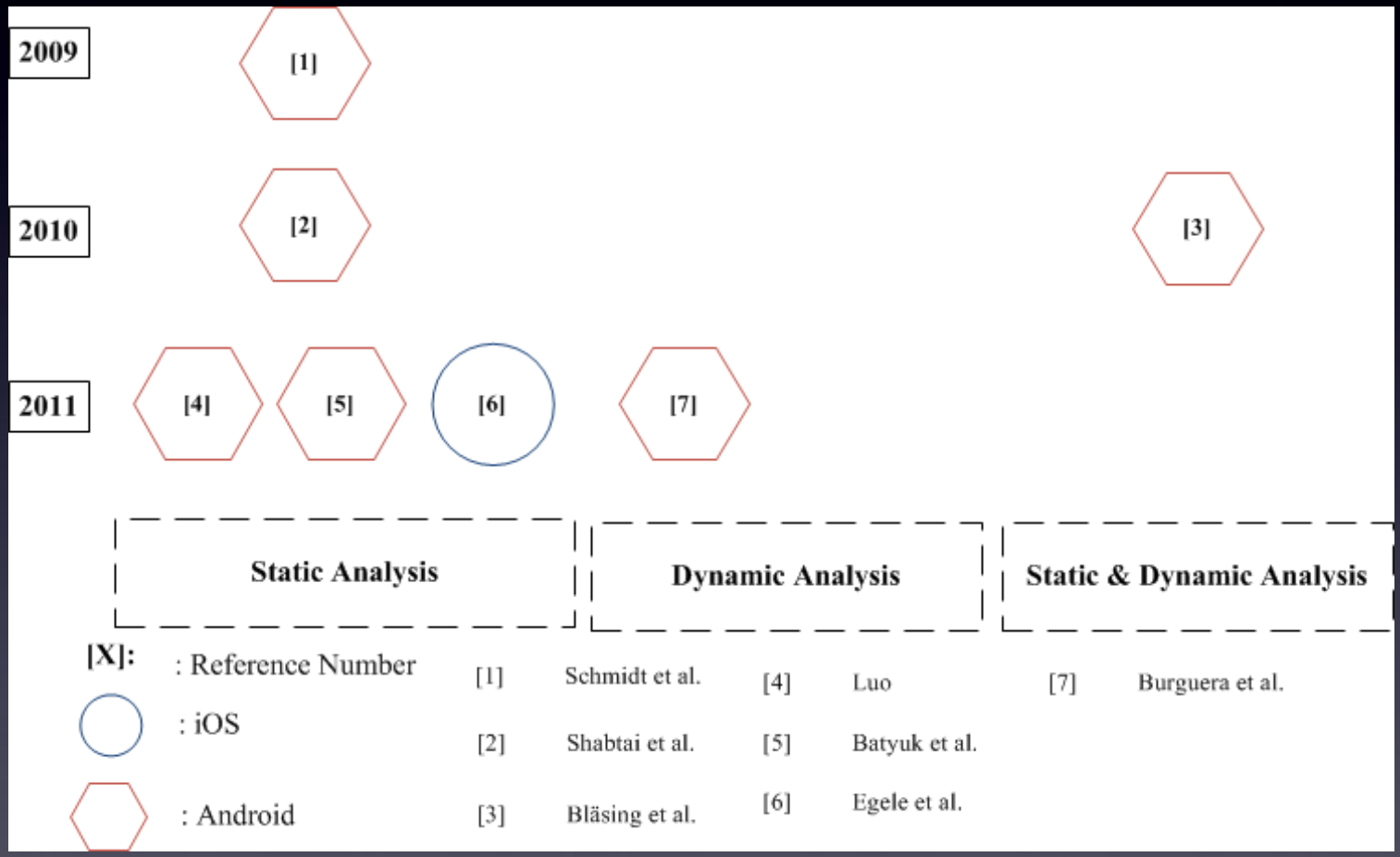
Preliminaries

- Detection Metrics
  - **True Positive Rate (TPR)** : The probability of an alarm given an actual intrusion
  - **False Positive Rate (FPR)** : The probability of an alarm given no intrusion
- **Hooking**: The process of intercepting a method call
  - MobileSubstrate Framework
- **Jailbreak:** The procedure with which a user is able to gain root permissions on iOS platforms
  - Corona Exploit , developed by Pod2G

CONTENTS

Related Work

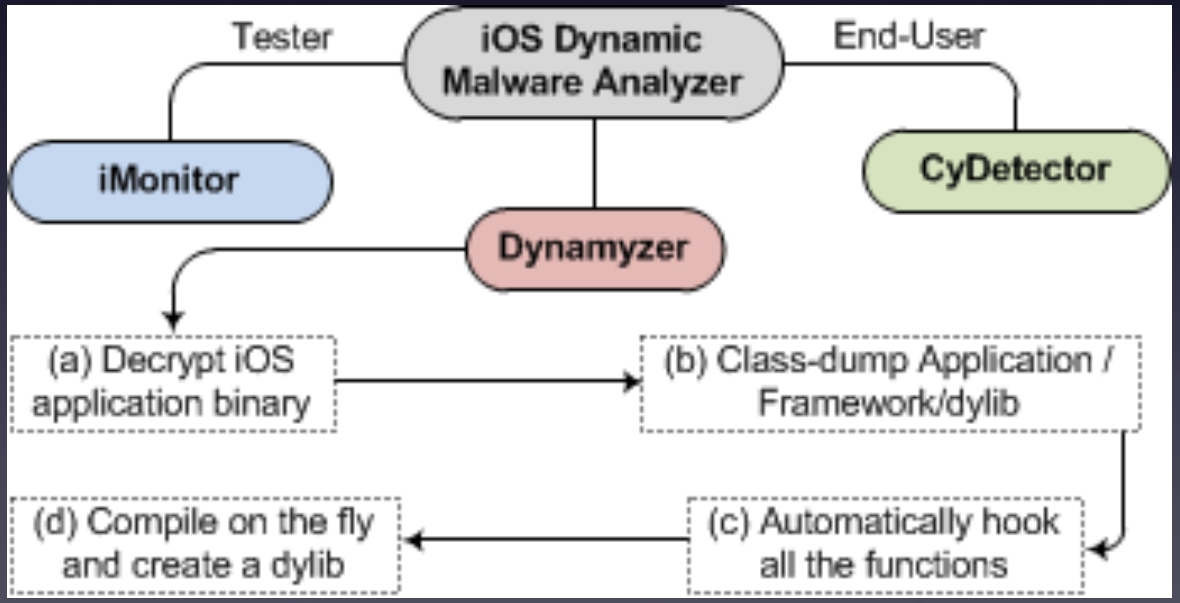
- Relative Research Works



CONTENTS

iDMA: iOS Dynamic Malware Analyzer

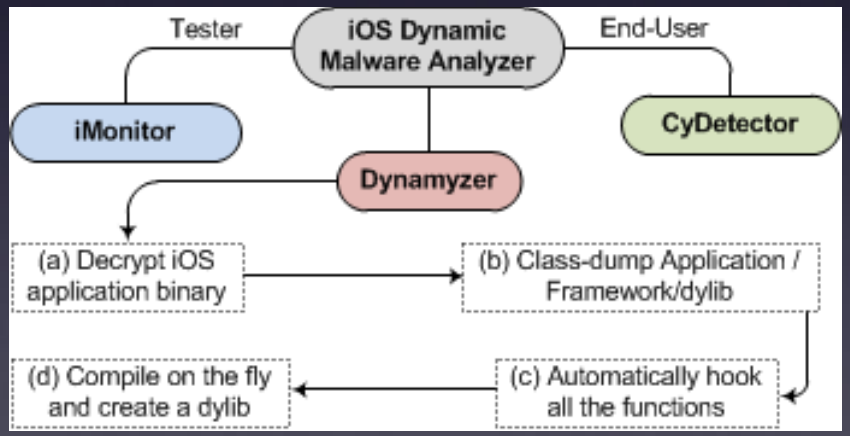
- iOS Dynamic Malware Analyzer (iDMA): an automated malware analyzer and detector for the iOS platforms
  - its main daemon written in Objective-C
  - it includes 3 subroutines written as Objective-C functions, dynamic libraries (dylibs) or bash scripts
    - iMonitor
    - Dynamyzer
    - Cydetector



Dynamizer

- **Dynamizer** is able to create a proper dylib for hooking and monitoring the API class methods as they are derived from the analyzed software (Application, Framework, dylib)

(a) decrypt the binary file in case of an encrypted application (usual case)
(b) class-dump it to generate the header file(s)
(c) hook all methods by injecting the proper source code in order to monitor them
(d) compile and create dylib's to monitor the behavior of the application




```
%hook SBApplicationIcon                 # hooking the SBApplicationIcon header
-(void)launch{                          # override the function
        WritingAtPath: @"/User/Library/Logs/FunctionBehave.txt"
        start synchronizeFile           # only one function is able to write at the log file
        seekToEndOfFile                 # finds the end of file
        writeData: displayName \n       # appends the name of the function and the returned
        end synchronizeFile
        closeFile
}
%end                                    # end of hooking
```

iMonitor

- **iMonitor** is responsible for monitoring and logging the behavior of the running application(s) during the interaction with native or non-native iOS frameworks
- It consists of a combination of dylibs, one for each header, created by the Dynamyzer
- Currently, it tracks in real-time
  - 51 public and 121 private iOS frameworks provided by Apple
  - 81 standard C frameworks for Portable Operating System Interface (POSIX) systems
- It is able to support third-party frameworks (or generally any kind of API) after analyzing it through the Dynamizer module

iDMA: iOS Dynamic Malware Analyzer

- **CyDetector** is a dynamic signature-based detection tool able to detect only specific system calls commonly used by third-party applications to secretly acquire access to user private data
- Detects the most important calls to native iOS API methods that have been highlighted in the literature to be responsible for leaking sensitive user data

Resources	Methods being monitored	iOS Class
Address Book	-(void)ABAddressBookRef ABAddressBookCreate	ABAddressBook
Photo Album	- (void)imagePickerController:(UIImagePickerController *)picker	UIImagePickerController
Unique Device ID	- (UIDevice *)currentDevice	UIDevice
GPS coordinates	- (void)locationUpdate:(CLLocation *)location	CoreLocation
Siri Authentication	- (void)setSessionValidationData:(NSData *)_data	SACreateAssistant
WiFi Connection	- (SCNetworkReachabilityRef) SCNetworkReachabilityCreateWithAddress (CFAllocatorRef allocator, const struct sockaddr *address)	SCNetworkReachability

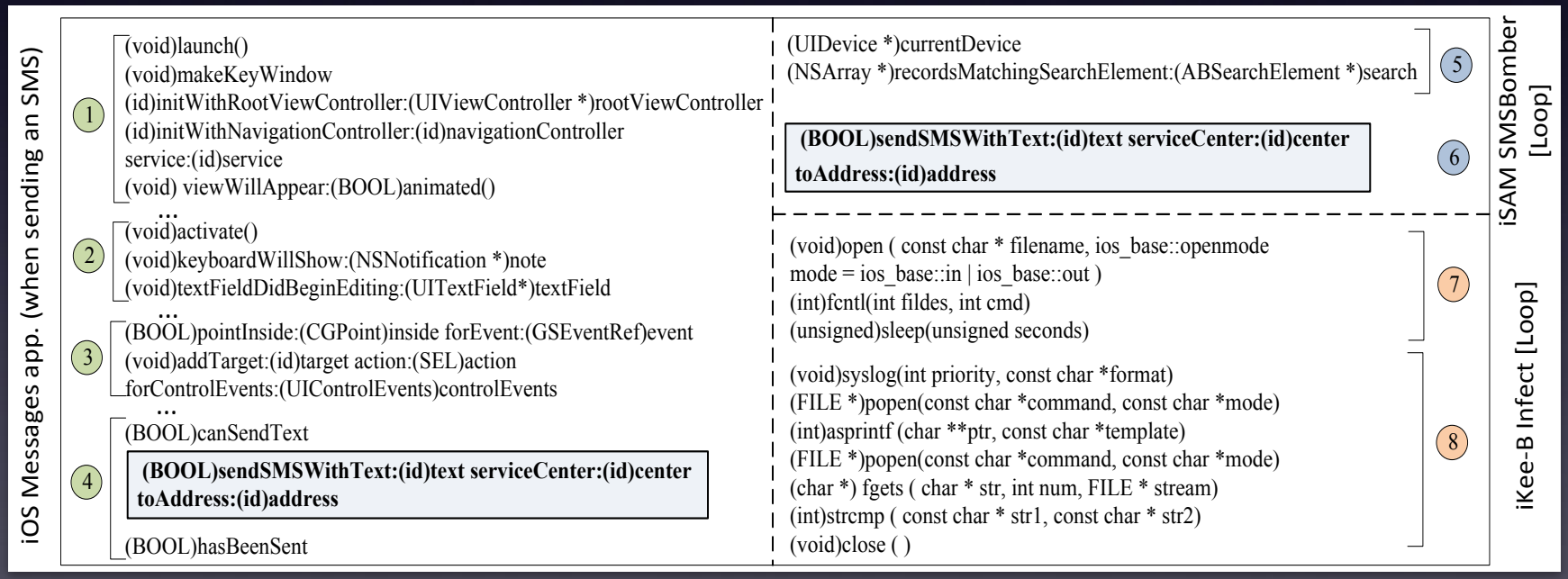
CONTENTS

Detecting iOS malwares (a real case scenario)

- **Signature-based** detection is not always enough
  - CyDetector will inform the user about the invocation of a suspicious method
  - but, a call to invoke a suspicious methods does not mean necessarily that the application is behaving malicious
    - e.g., a Buddy-finder application requires the user's location
- However, we need **behavior-based** tools able to detect malicious code in an efficient way
  - to succeed it is necessary to understand the basic properties of mobile malware
    - how it is created ?
    - what tasks it executes and in which order ?
    - what differentiates it from a legitimate application ?

Detecting iOS malwares (a real case scenario)

- An example of analyzing malicious and legitimate applications via iDMA
  - Messages: the app for communication via SMS, MMS, iMessaging
  - iKee-B : 1st iOS malware
  - iSAM (iPhone Stealth Airborne Malware) : 1st iOS rootkit



Detecting iOS malwares via machine learning

- What's more…
  - Use machine learning to create an automatic malware detection tool
  - Would allow us to create behavioral profiles and cross-evaluated them by well-known machine learning classifiers
    - need a training set to train the classifier about the normal and malicious behavior of the application of interest
    - also a test set to assess the detection rate and classify the software of interest as malicious or not

Method invocations	
iKee-B testing SSH Vulnerability	**Modified version of SMSBomber**
(void)syslog(int priority, const char *format)	(UIDevice *)currentDevice
(int)asprintf (char **ptr, const char *template)	(void)setSessionValidationData:(NSDat a *) _data
(FILE *)popen(const char *command, const char *mode)	(BOOL)sendSMSWithText:(id)text serviceCenter:(id)center toAddress:(id)address
(char *) fgets (char * str, int num, FILE * stream)	
(int)strcmp (const char * str1, const char * str2)	

CONTENTS

Malware Detection Results

- We cross-evaluated 4 well-known machine learning classifiers

Bayesian Network		RBF		KNN		Random Forest	
TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
100%	4%	92%	0%	100%	7%	100%	0%

- All the experiments present highly accurate results
  - they provide strong evidence that behavior-based classification in terms of API method invocation may be a very precise means of detecting new types of malware or variations of existing ones
  - but, more research is needed to better assess this potentiality
    - for instance, iDMA can be used to construct behavioral profile by considering a large number of legitimate applications and then…
    - use the profiles to a properly designed IDS to safeguard the device in real time

Conclusions

- The evolution of malware is a continuous race between intruders and defenders.
- Our aim: highlight smartphones weaknesses and offer in-depth information towards combating such threats.
- We design and implement iDMA
  - produce exploitable information about the behavior of the application of interest in terms of which method is invoked and in what sequential order
- Overall, designing a highly secure mobile device is still a very challenging task.

Thank You !