OWASP

Open Web Application Security Project

모바일기기 주요정보 보호

남대현

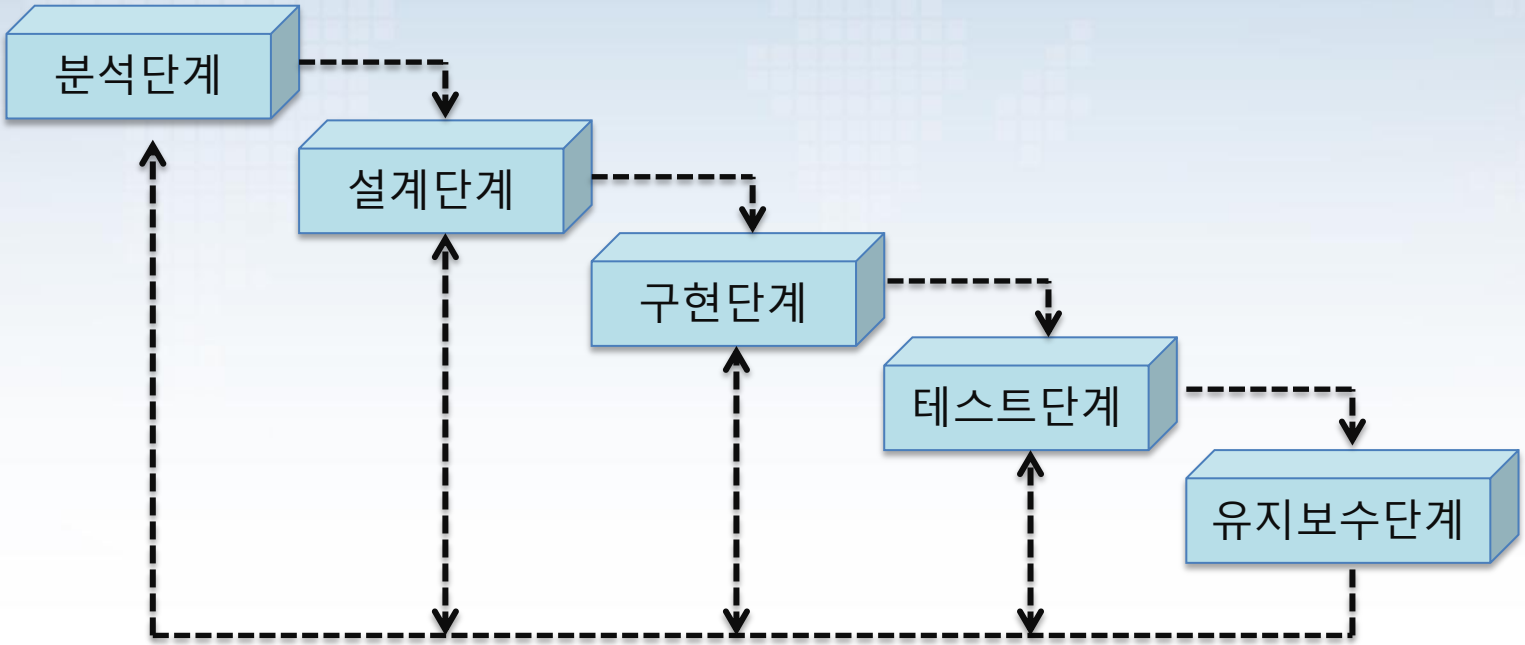
Agenda

- Software Development Life Cycle
- 공격 이해를 위한 개념 설명
  - USBmux
  - lockdownd
- Case Study
  - Log Handling
  - TestClass, AlertDialog, PlainText Handling
  - Background Image Cache
  - Encryption File & key Handling

Software Development Life Cycle

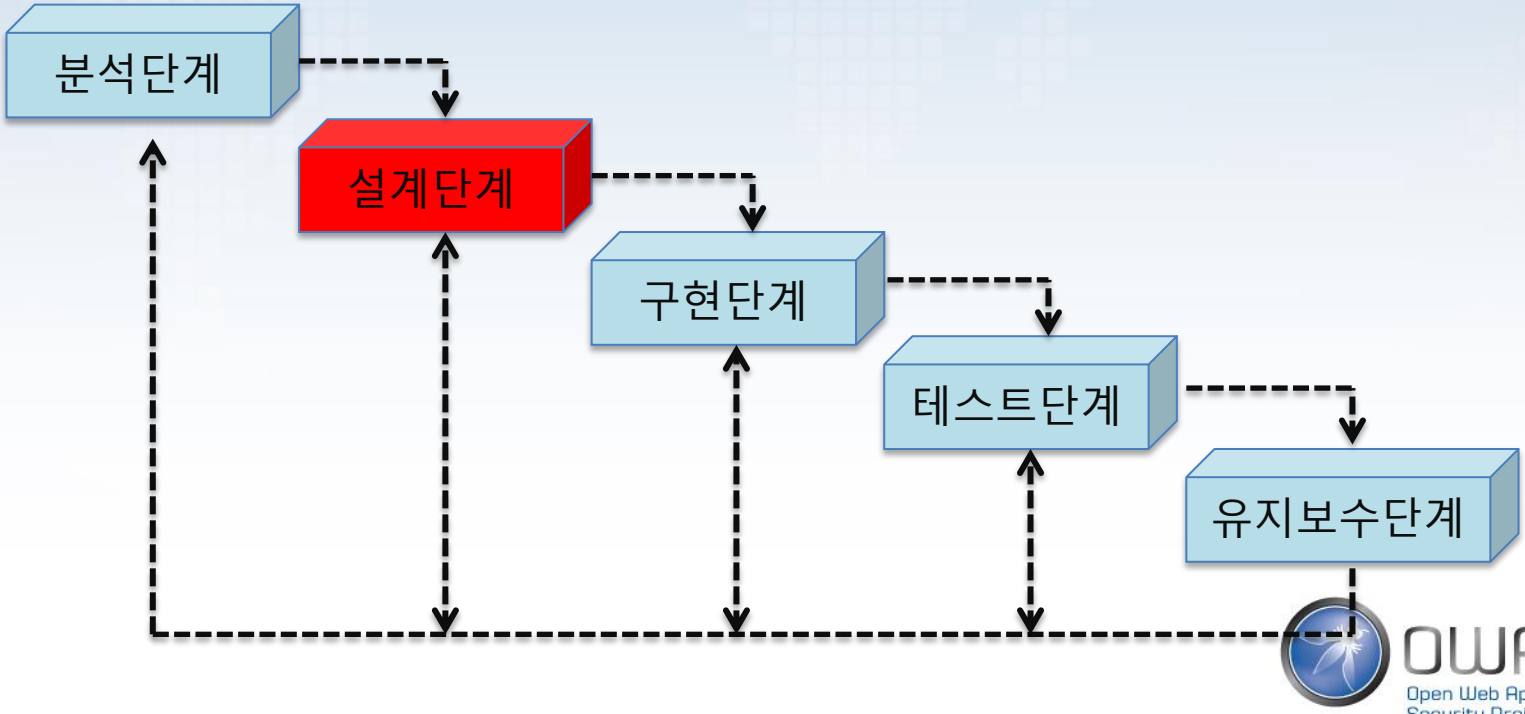
SDLC(Software Development Life Cycle)

- 각 단계 명확, 각 단계들간의 유기적 연관성
- 대부분의 개발 프로젝트가 SDLC모형을 기반으로 진행



SDLC(Software Development Life Cycle)

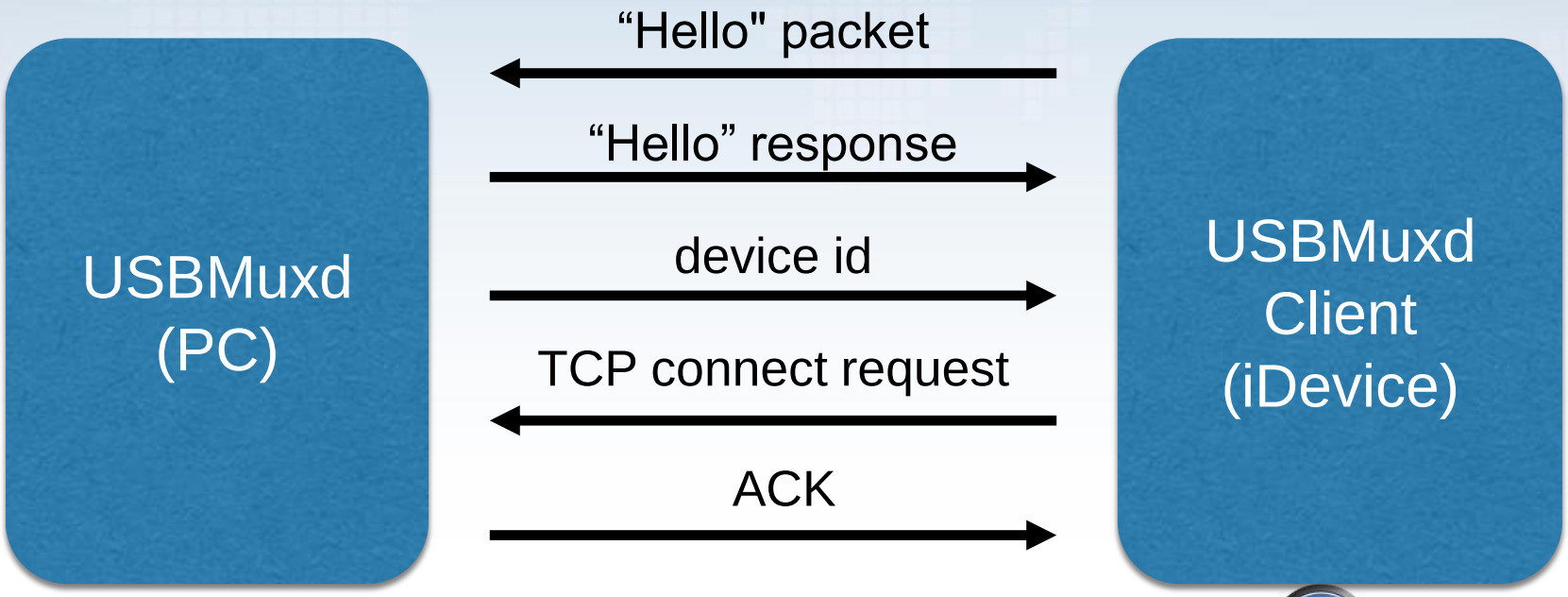
- 모바일 기기 정보보호를 위해
- 설계 단계에서 고려했으면 하는 것들을 사례별로 설명
- iOS에서 새로운 시각의 공격방법을 사례별로 설명



공격 이해를 위한 개념 설명

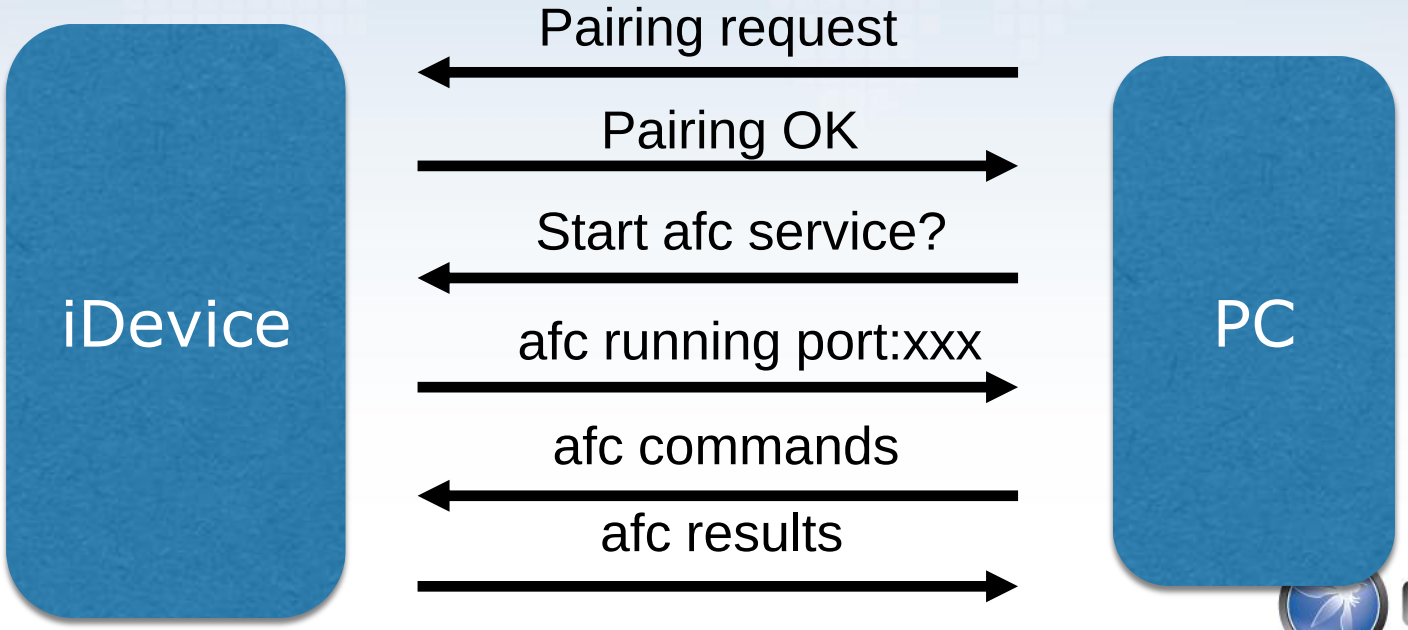
USBMux Protocol

- iTunes iDevice와 통신할 때 usbmux사용
- TCP와 유사하며 USB통하여 TCP소켓 터널링



Lockdownd

- USBMux Protocol을 사용하여 접속
- Pairing, Activation
- AFC(Apple File Conduit)사용을 위해 Pairing



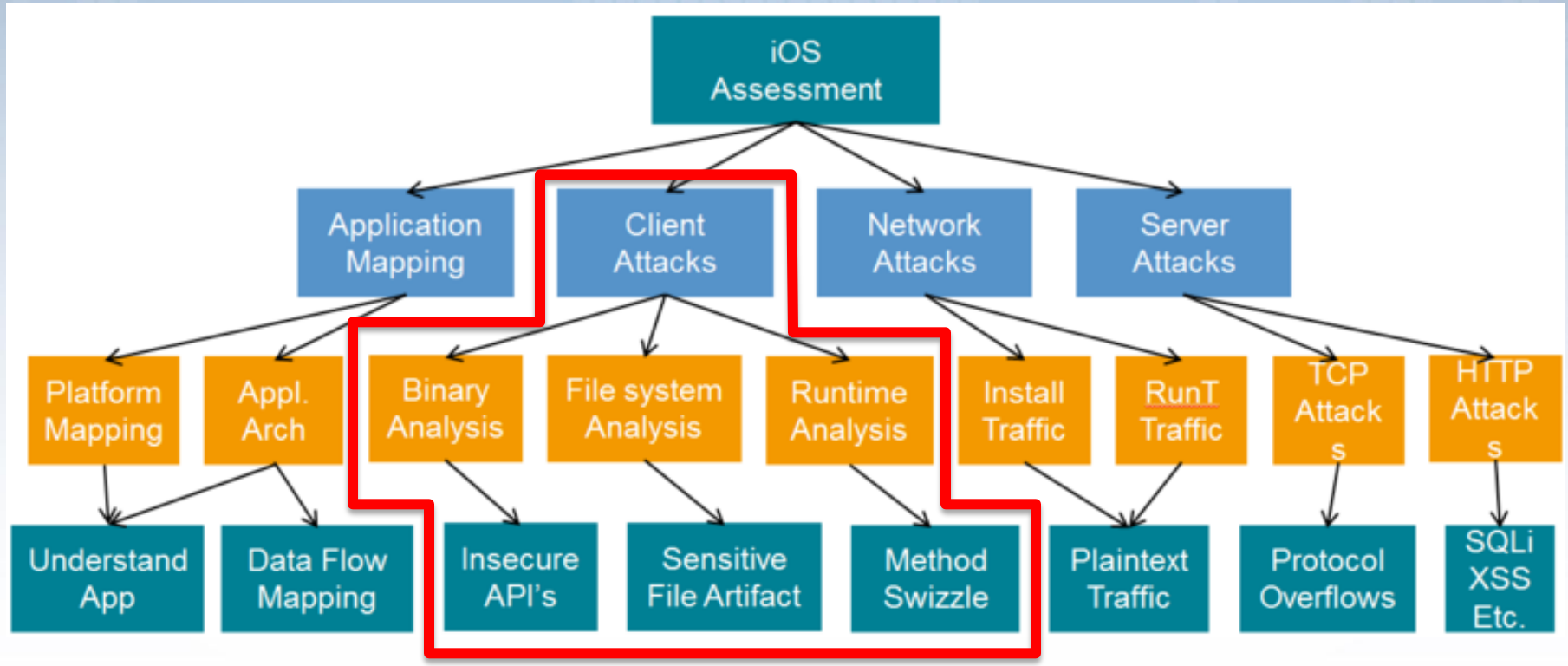
AFC(Apple File Conduit)

- iTunes와 iDevice간의 데이터 교환에 사용
- Sandbox에서 파일 추출, 삭제 등
- AFC Daemon도 Sandbox에 존재하기 때문에 특정영역으로 사용 제한
- AFC서비스를 위해서 Lockdown Pairing필요

Keychain

- Mac OS X, iOS환경의 비밀번호 관리 시스템
- Mac OS 8.6부터 사용
- PowerTalk이라는 Apple Email System 개발하면서 개발됨
- 다양한 형태의 데이터 저장가능
- Non-Jailbreak Device에서는 AFC를 통한 Keychain접근 불가
- iOS환경에서 주요한 데이터 저장에 사용

OWASP iOS App Security Test Cheat Sheet



OWASP
Open Web Application Security Project

Case Study

Log Handling

- 행자부의 모바일 개발 가이드

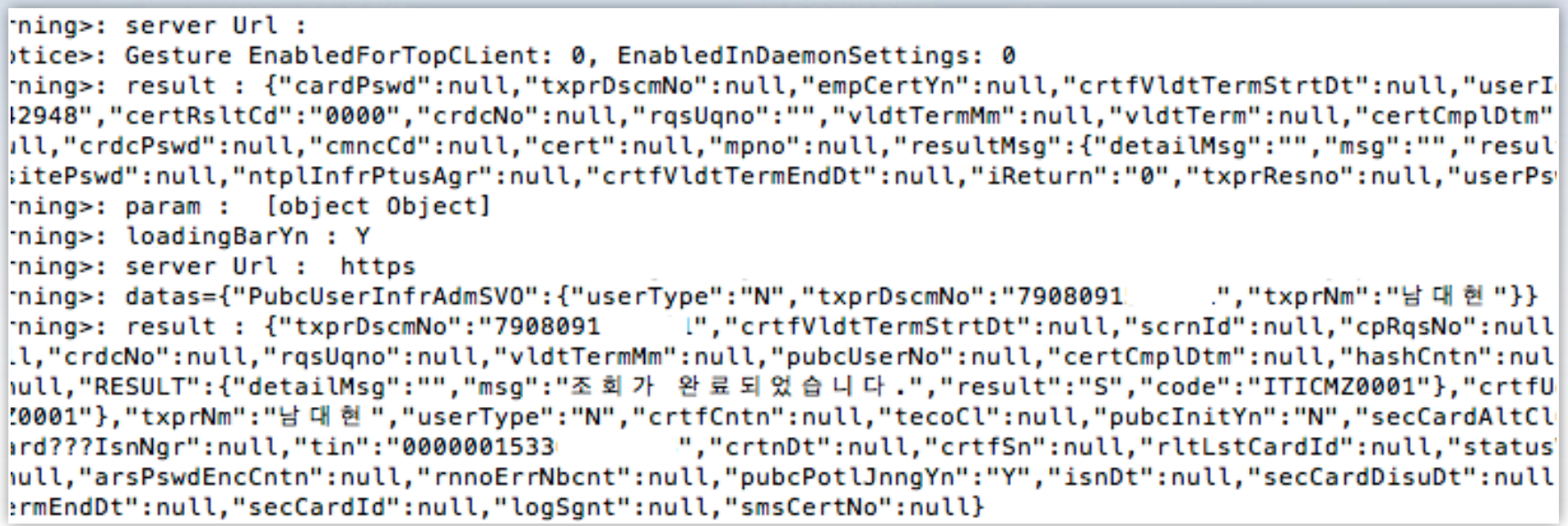
번호	보안약점	설 명
1	잘못된 세션에 의한 데이터 정보 노출	잘못된 세션에 의해 권한 없는 사용자에게 데이터 노출이 일어날 수 있는 보안약점
2	제거되지 않고 남은 디버그 코드	디버깅을 위해 작성된 코드를 통해 권한 없는 사용자 인증우회(또는 중요중보)접근이 가능해지는 보안약점
3	시스템 데이터 정보노출	사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점
4	Public 메소드부터 반환된 Private 배열	private로 선언된 배열을 public으로 선언된 메소드를 통해 반환(return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열의 수정될 수 있는 보안약점
5	Private 배열에 Public 데이터 할당	public으로 선언된 데이터 또는 메소드의 인자가 private 선언된 배열에 저장되면, private 배열을 외부에서 접근할 수 있게 되는 보안약점

Log Handling

- 일련의 프로그램의 흐름을 알 수 있음.
- Debug용도의 Log출력으로 중요정보 노출
  - Encryption Class Log
    - PIN, Messages, etc
  - Keychain Class Log
  - Pairing Information
  - Jailbreak Check Routine

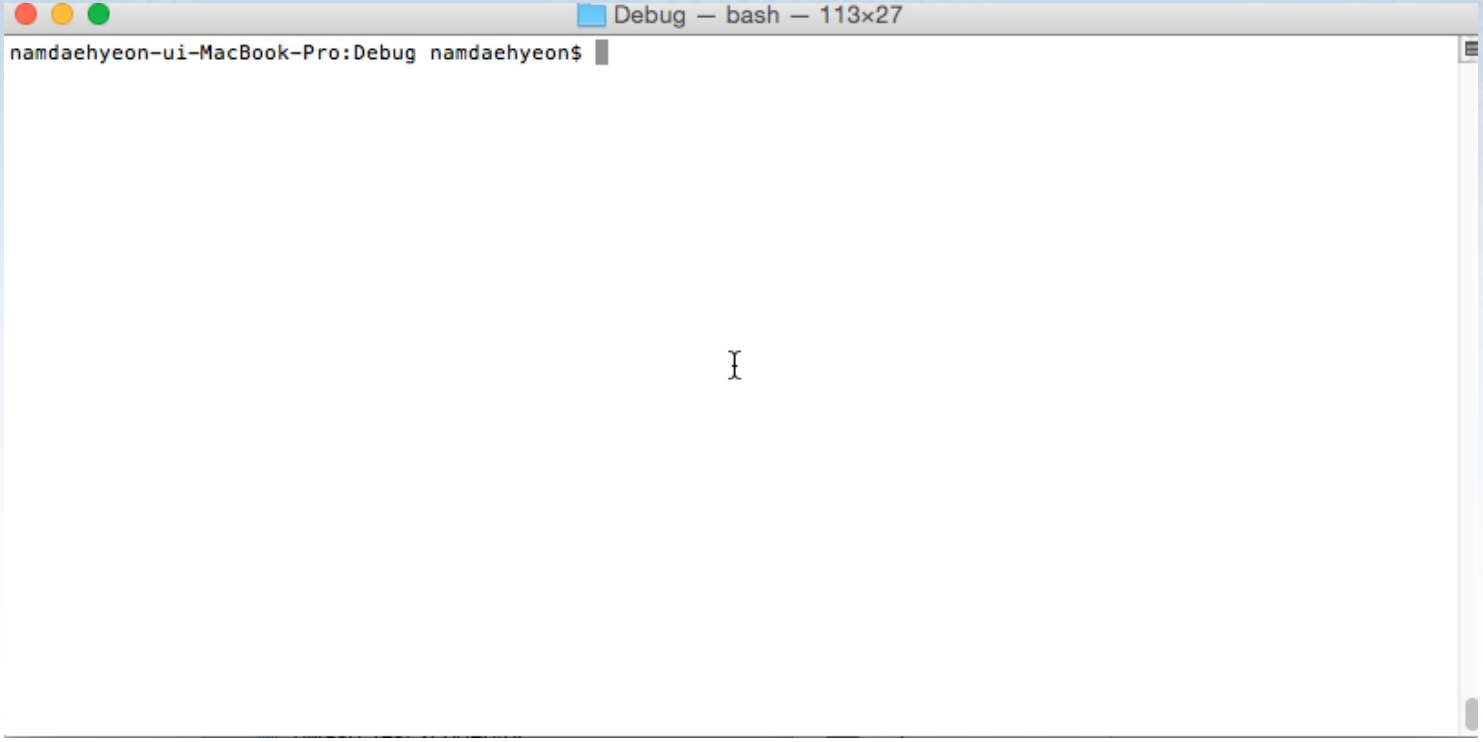
Log Handling

- Lockdown 서비스 중 “com.apple.syslog_relay”
- Xcode와 같이 Device로그 확인가능



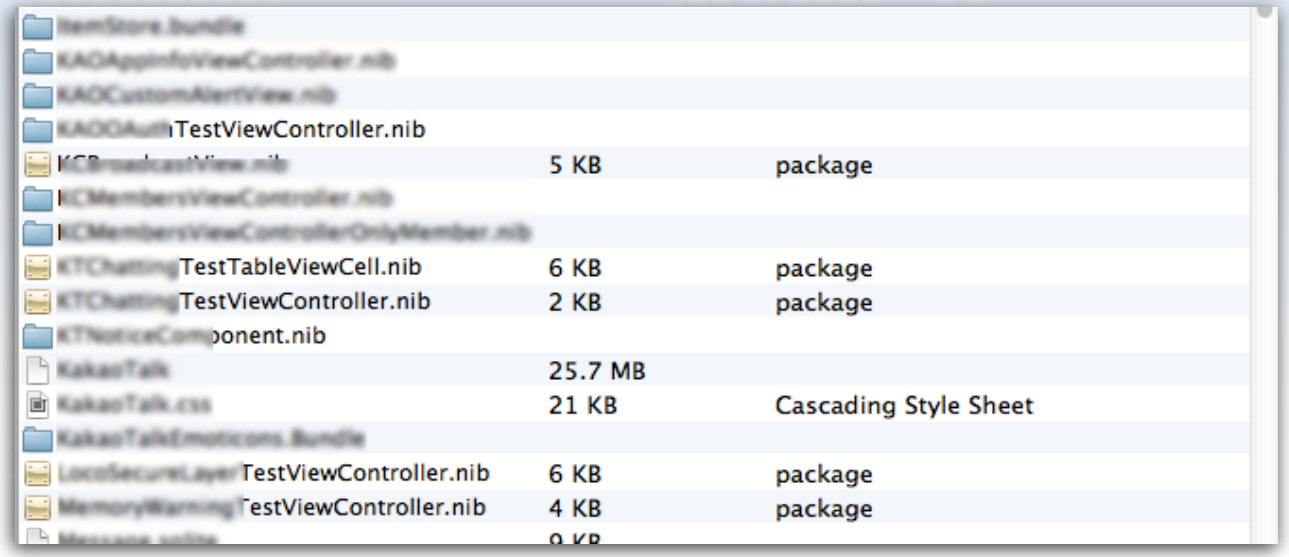
Log Handling

- DEMO



Abusing Test Class

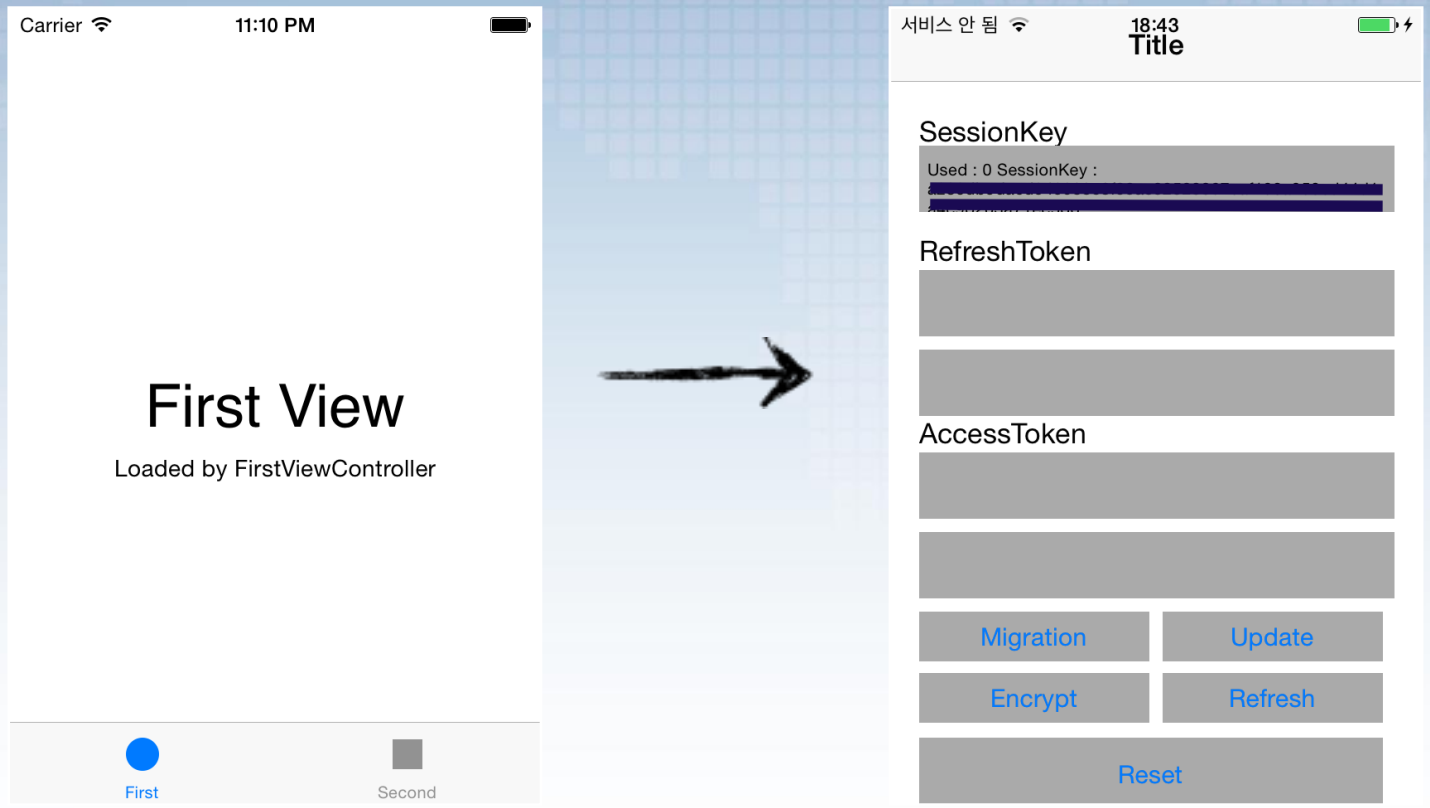
- TestViewController
  - 개발단계에서 테스트에 사용된 다양한 클래스 존재
  - 이러한 클래스는 공격자에게 주요한 정보제공



Abusing Test Class

- TestViewController
  - 테스트 클래스가 컴파일 되어 있는지 확인하는 방법
  - Debugger
    - Cycript –p <pid>
    - 메모리에 로딩된 클래스 정보 확인
  - App Decryption
    - Strings <app> | grep TestClass
    - Decompiler

Abusing Test Class



UIApp.keyWindow.rootViewController
= [[TestViewController alloc]init]

OWASP
Open Web Application Security Project

Abusing AlertDialog, UIAlertController

- 금융감독원

앱 위·변조 방지대책

① 앱 실행시 폰 임의개조 탐지 및 차단 여부 (실 테스트 점검)

ㅇ 임의개조 폰에서 앱 실행 시 탐지 및 서비스 접속 차단 여부 확인

② 위·변조 앱에서 접속시 탐지 및 차단 여부 (정상 앱에 악성코드를 추가한 리패키징 앱 차단여부 등) (실 테스트 점검)

ㅇ 주요 검증대상 파일을 변조 후, 앱을 실행하여 정상적으로 서비스 이용 가능 혹은 제한 여부 확인

- 적용된 앱 무결성 검증기술의 특성에 따라 변조대상을 선별하여 점검 진행

ㅇ 주요파일 무결성 검증(변조) 대상

- 안드로이드 : 설치파일, 실행파일, 리소스 파일(xml, 이미지 등) 등
- iOS : 실행파일, 리소스 파일(xml, 이미지 등) 등

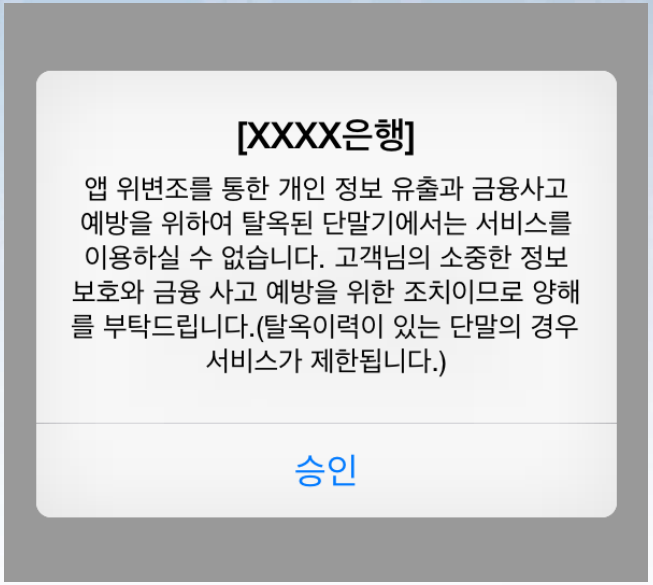
ㅇ 주요파일 무결성 검증 방법

- **(안드로이드)** 설치파일 다운로드 → 파일 변조 → 스마트폰에 주입 및 설치 → 앱 실행 후 정상적으로 서비스 이용/제한 여부 확인
- **(iOS*)** 스마트폰으로부터 변조대상 추출 → 파일 변조 → 스마트폰에 주입 → 앱 실행 후 정상적으로 서비스 이용/제한 여부 확인

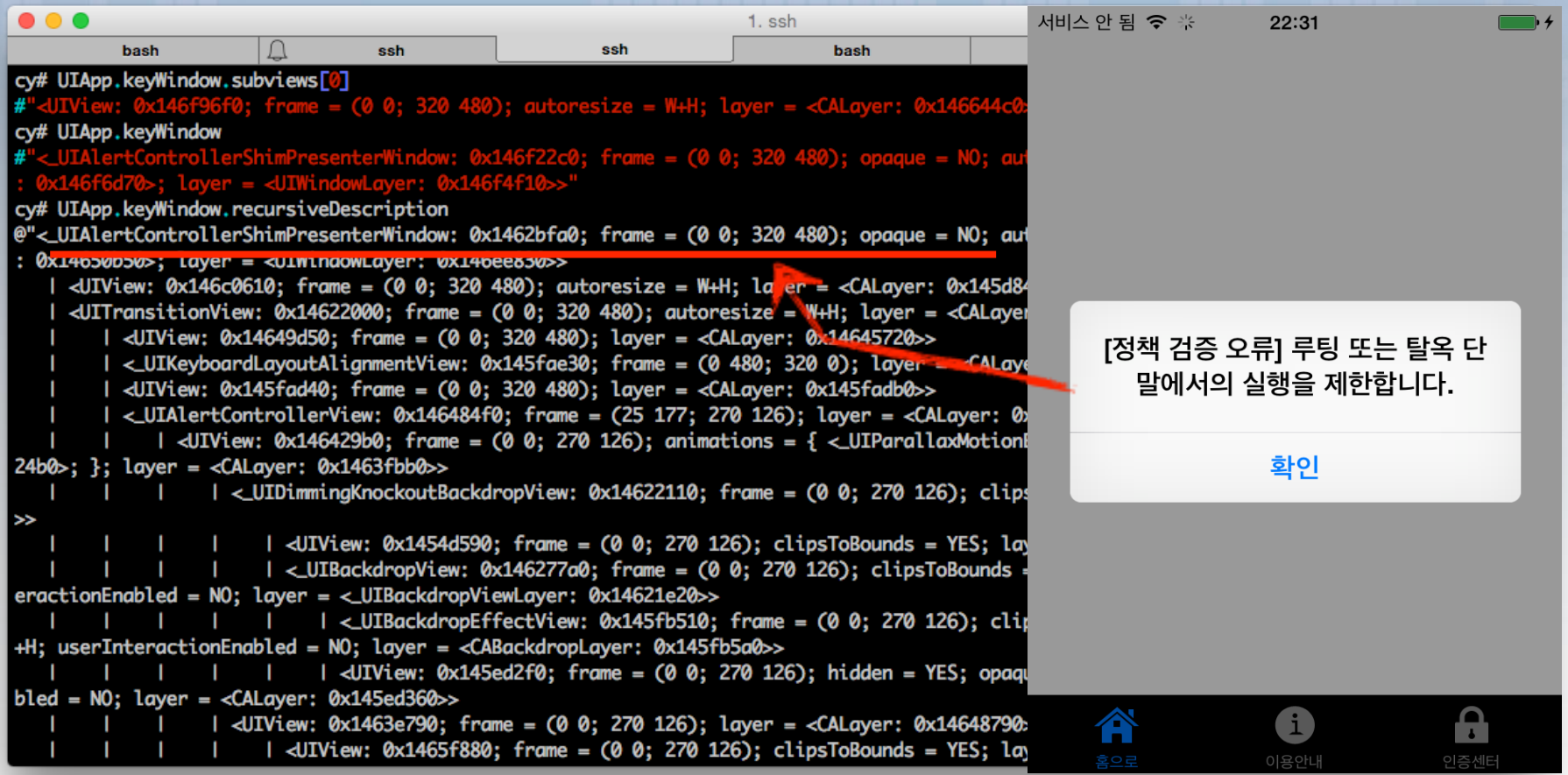
* *iOS 실행파일 변조는 임의개조 폰에서만 가능하며, 순정 iOS 최신 버전에서 파일 추출/주입 도구가 정상적으로 동작하지 않을 경우 점검 불가*

Abusing AlertDialog, UIAlertController

- 폰 임의개조 기기에서 앱 실행방지
  - 솔루션에 의한 탐지 및 종료
  - UIAlertView
  - UIAlertController
- 설계 과정에서 놓치기 쉬운 공격
  - 폰 임의개조기기임을 알릴 때
  - 서브 뷰 삭제
  - [self.window.rootViewController.view removeFormSuer..]



Abusing AlertDialog, UIAlertController

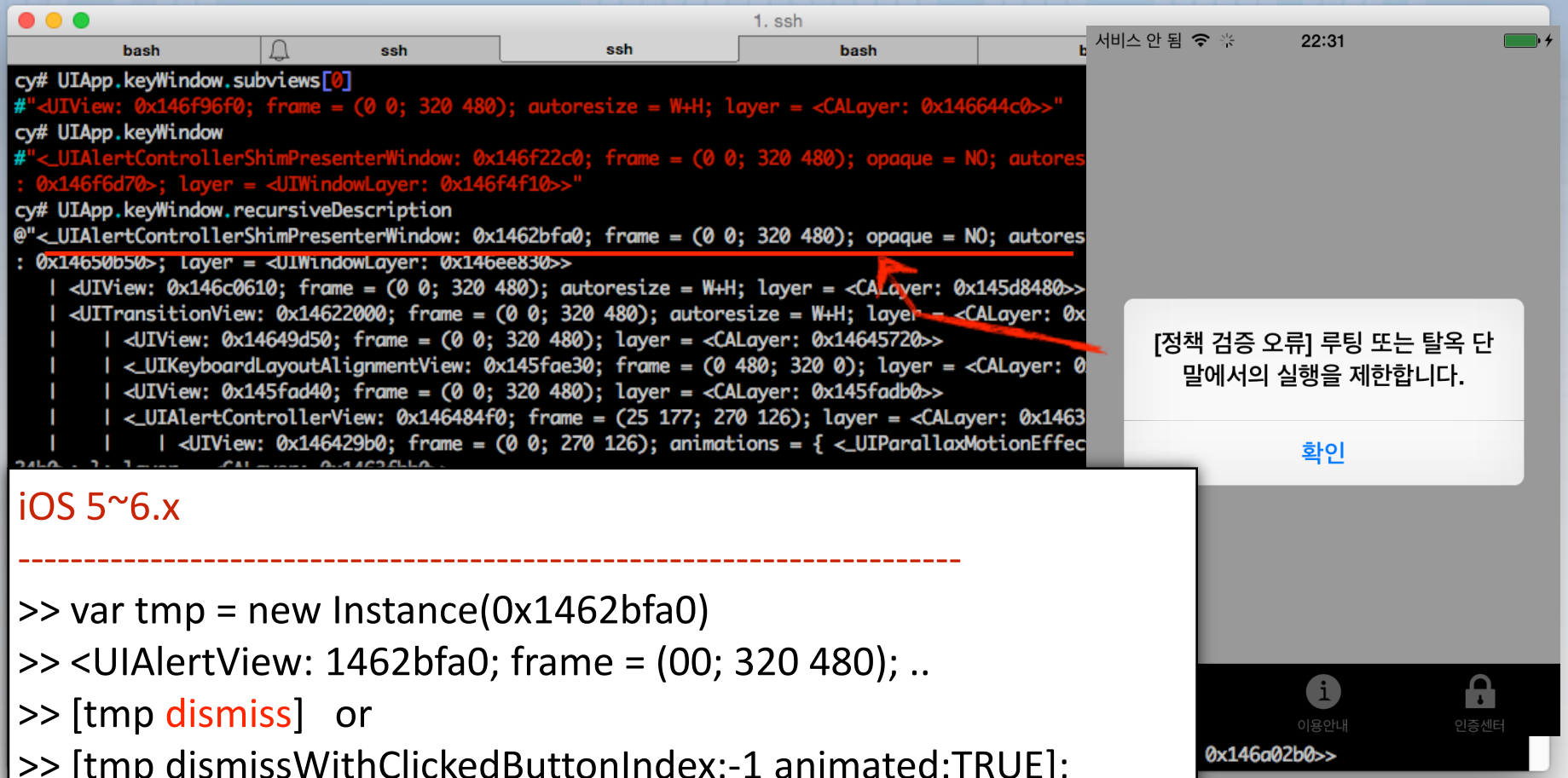


OWASP
Open Web Application Security Project

Abusing AlertDialog, UIAlertController

- Objects Hierarchy
  - 화면에 보이는 객체, 혹은 주소를 이용하여 인스턴스 생성
  - UIAlertView, UIAlertController에 메시지 전송
  - 공격은 디버거를 이용(gdb, cycript)
- Dynamic Library제작
  - ASLR이지만 Hierarchy의 첫 번째 Subview객에체 메시지 전송

Abusing AlertDialog, UIAlertController



iOS 5~6.x

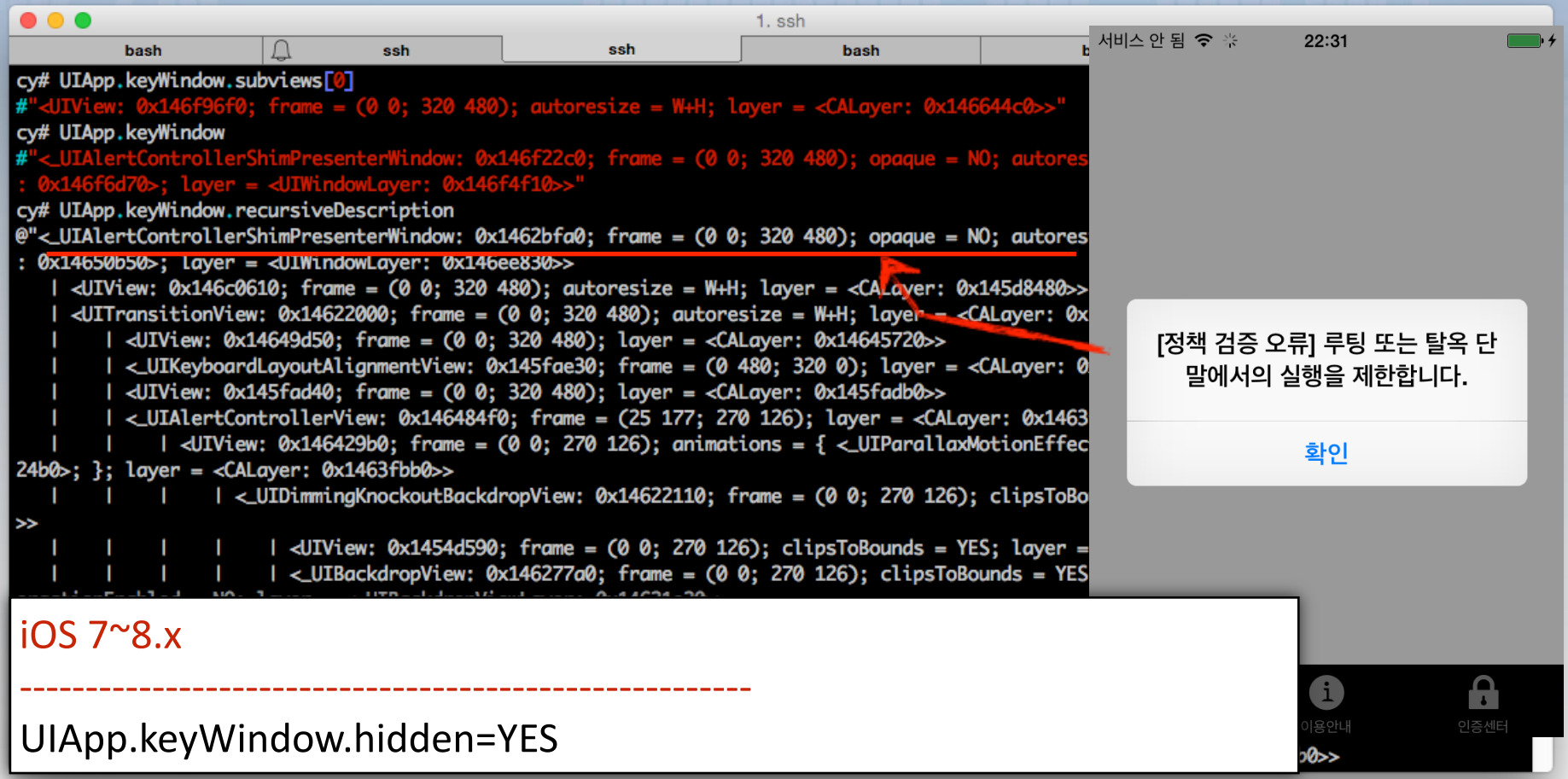
>> var tmp = new Instance(0x1462bfa0)

>> <UIAlertView: 1462bfa0; frame = (00; 320 480); ..

>> [tmp dismiss] or

>> [tmp dismissWithClickedButtonIndex:-1 animated:TRUE];

AlertDialog, UIAlertController Handling



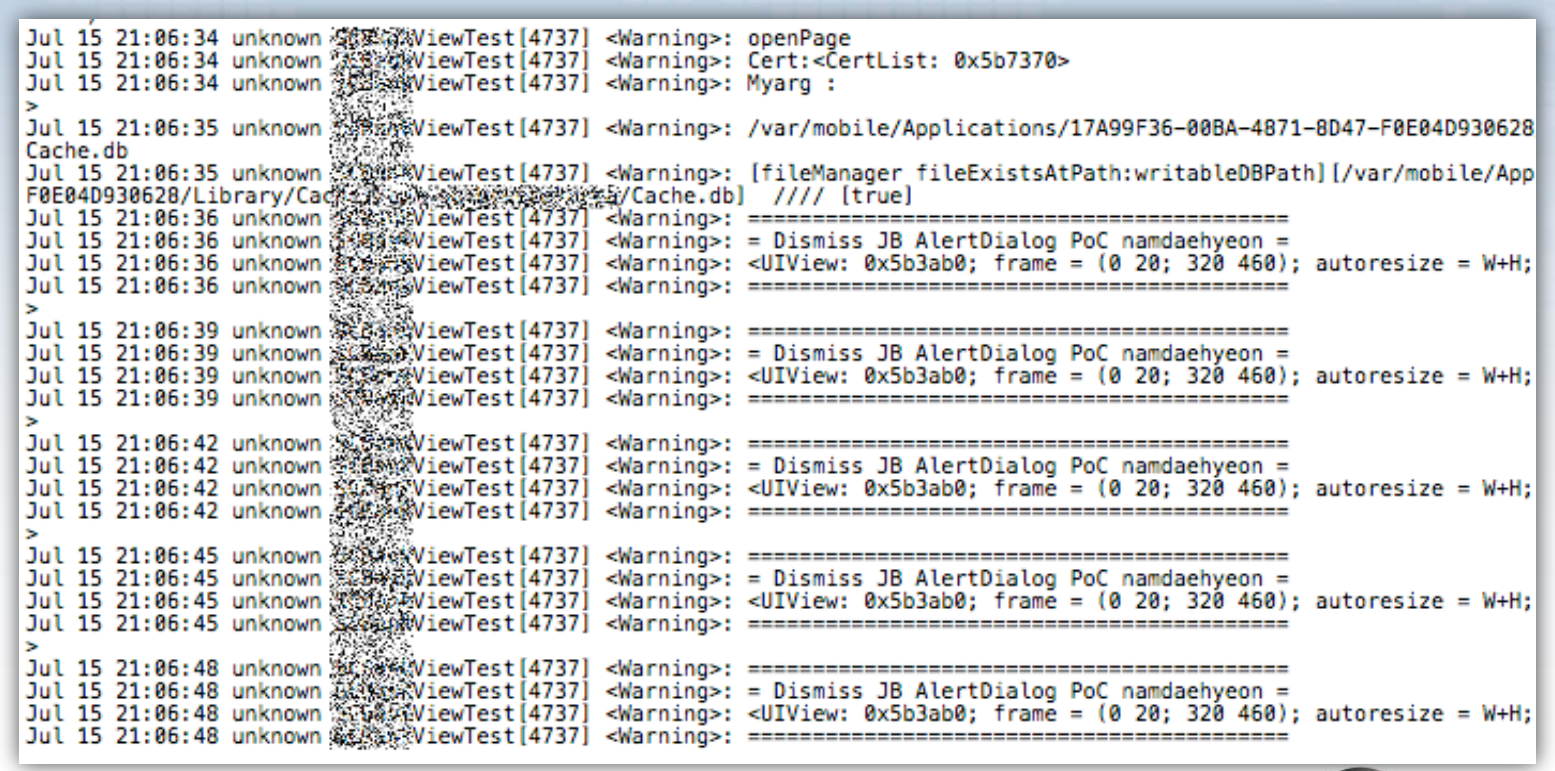
iOS 7~8.x

UIApp.keyWindow.hidden=YES

OWASP
Open Web Application Security Project

Abusing AlertDialog, UIAlertController

- Dynamic Library PoC



```
Jul 15 21:06:34 unknown [redacted]ViewTest[4737] <Warning>: openPage
Jul 15 21:06:34 unknown [redacted]ViewTest[4737] <Warning>: Cert:<CertList: 0x5b7370>
Jul 15 21:06:34 unknown [redacted]ViewTest[4737] <Warning>: Myarg :
>
Jul 15 21:06:35 unknown [redacted]ViewTest[4737] <Warning>: /var/mobile/Applications/17A99F36-00BA-4871-8D47-F0E04D930628
Cache.db
Jul 15 21:06:35 unknown [redacted]ViewTest[4737] <Warning>: [fileManager fileExistsAtPath:writableDBPath][/var/mobile/App
F0E04D930628/Library/Cac[redacted]/Cache.db]  //// [true]
Jul 15 21:06:36 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
Jul 15 21:06:36 unknown [redacted]ViewTest[4737] <Warning>: = Dismiss JB AlertDialog PoC namdaehyeon =
Jul 15 21:06:36 unknown [redacted]ViewTest[4737] <Warning>: <UIView: 0x5b3ab0; frame = (0 20; 320 460); autoresize = W+H;
Jul 15 21:06:36 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
>
Jul 15 21:06:39 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
Jul 15 21:06:39 unknown [redacted]ViewTest[4737] <Warning>: = Dismiss JB AlertDialog PoC namdaehyeon =
Jul 15 21:06:39 unknown [redacted]ViewTest[4737] <Warning>: <UIView: 0x5b3ab0; frame = (0 20; 320 460); autoresize = W+H;
Jul 15 21:06:39 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
>
Jul 15 21:06:42 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
Jul 15 21:06:42 unknown [redacted]ViewTest[4737] <Warning>: = Dismiss JB AlertDialog PoC namdaehyeon =
Jul 15 21:06:42 unknown [redacted]ViewTest[4737] <Warning>: <UIView: 0x5b3ab0; frame = (0 20; 320 460); autoresize = W+H;
Jul 15 21:06:42 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
>
Jul 15 21:06:45 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
Jul 15 21:06:45 unknown [redacted]ViewTest[4737] <Warning>: = Dismiss JB AlertDialog PoC namdaehyeon =
Jul 15 21:06:45 unknown [redacted]ViewTest[4737] <Warning>: <UIView: 0x5b3ab0; frame = (0 20; 320 460); autoresize = W+H;
Jul 15 21:06:45 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
>
Jul 15 21:06:48 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
Jul 15 21:06:48 unknown [redacted]ViewTest[4737] <Warning>: = Dismiss JB AlertDialog PoC namdaehyeon =
Jul 15 21:06:48 unknown [redacted]ViewTest[4737] <Warning>: <UIView: 0x5b3ab0; frame = (0 20; 320 460); autoresize = W+H;
Jul 15 21:06:48 unknown [redacted]ViewTest[4737] <Warning>: ==========================================
```

Plain Text in the Objects Hierarchy

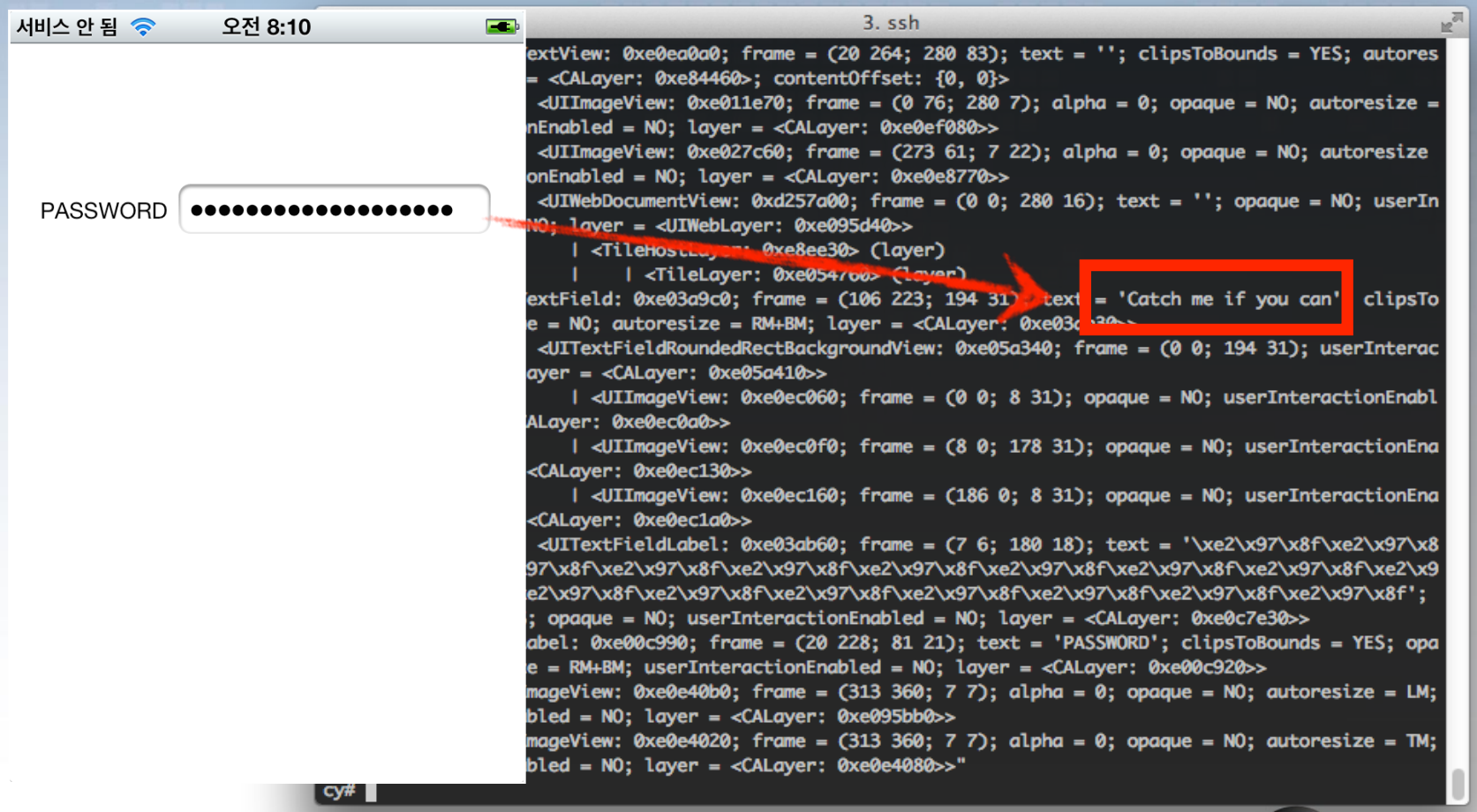
- 금융감독원

영역	점검항목	세부내용	적정 여부
모바일 금융 보안대책	백신프로그램 적용	앱 실행시 백신프로그램 구동 여부	
		백신프로그램 최신버전 업데이트 여부	
	입력정보 보호대책 적용 여부	주요정보 입력시 가상 보안키패드 등 입력정보 보호 대책 적용 여부	
	금융정보 종단간 암호화 적용 여부	스마트폰 앱과 금융회사 전자금융 서버간의 종단간 암호화(End-to-End) 적용 여부	
	거래전문 무결성 검증기법 적용	거래전문 무결성 정보생성 및 검증 여부	
		표준 통신규약 적용 여부	

Plain Text in the Objects Hierarchy

- 주요정보를 입력하는 구간에서 객체 계층에는 여전히 평문으로 존재할 때 공격자가 새로운 시각에서 공격하는 방법
- 주민등록번호 및 텍스트 입력창에 Secure옵션을 부여했음에도 평문으로 존재할 수 있고, 이 값을 획득하여 악의적으로 사용가능

Plain Text in the Objects Hierarchy



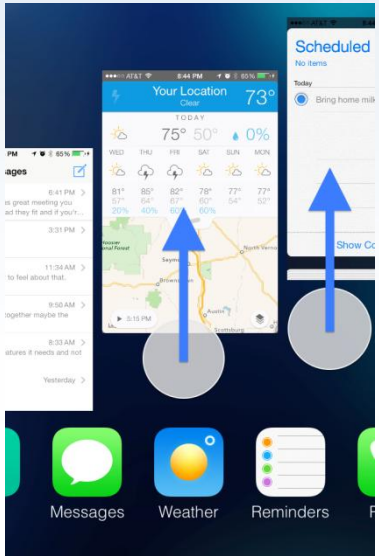
Plain Text in the Objects Hierarchy

- Dynamic Library PoC

```
NSLog(@"=========================================");
NSLog(@"= PlainText PoC namdaehyeon =");

id passLocation = [[[[[[[[UIApplication sharedApplication].keyWindow
                          subviews] objectAtIndex:0]
                        subviews] objectAtIndex:0]
                      subviews] objectAtIndex:5];

//        NSLog(@"%@",[tmp text]);
NSArray *array = [[NSString stringWithFormat:@"%@",passLocation]
                                    componentsSeparatedByString:@"; "];

//        NSLog(@"%@",array);

NSString *tmpStr = [NSString stringWithFormat:@"%@",[array objectAtIndex:3]];
tmpStr = [tmpStr stringByReplacingOccurrencesOfString:@"text = " withString:@""];
tmpStr = [tmpStr stringByReplacingOccurrencesOfString:@"'" withString:@""];

NSLog(@"%@",tmpStr);
NSLog(@"=========================================");
```

Background Image Cache

Remove sensitive information from views before moving to the background. When an app transitions to the background, the system takes a snapshot of the app's main window, which it then presents briefly when transitioning your app back to the foreground. Before returning from your applicationDidEnterBackground:method, you should hide or obscure passwords and other sensitive personal information that might be captured as part of the snapshot.



OWASP
Open Web Application Security Project

Background Image Cache

- AFC
  - /Library/Caches/Snapshots/<Identifier>
- Dynamic Library
- applicationDidEnterBackgound:
  - 주요 객체에 Hidden속성 부여
  - Self.backgroundImage = 이미지객체
  - [self.window addSubview:이미지객체]

Background Image Cache

- DEMO (Jailbreak환경)

Encryption File & Key Handling

- https://www.owasp.org/index.php/iOS_Developer_Cheat_Sheet

Broken Cryptography (M9)

Although the vast majority of cryptographic weaknesses in software result from poor key management, all aspects of a crypto system should be carefully designed and implemented. Mobile apps are no different in that regard.

Remediations

Never "hard code" or store cryptographic keys where an attacker can trivially recover them. This includes plaintext data files, properties files, and compiled binaries. Use secure containers for storing crypto keys; alternately, build a secure key exchange system where the key is controlled by a secure server, and never stored locally on the mobile device. Use only strong crypto algorithms and implementations, including key generation tools, hashes, etc. Use platform crypto APIs when feasible; use trusted third party code when not. Consumer-grade sensitive data should be stored in secure containers using Apple-provided APIs.

- Small amounts of data, such as user authentication credentials, session tokens, etc., can be securely stored in the device's Keychain (see Keychain Services Reference in Apple's iOS Developer Library).
- For larger, or more general types of data, Apple's File Protection mechanism can safely be used (see NSData Class Reference for protection options).

To more securely protect static data, consider using a third party encryption API that is not encumbered by the inherent weaknesses in Apple's encryption (e.g., keying tied to user's device passcode, which is often a 4-digit PIN). Freely available examples include SQLcipher (see http://sqlcipher.net).

Encryption File & Key Handling

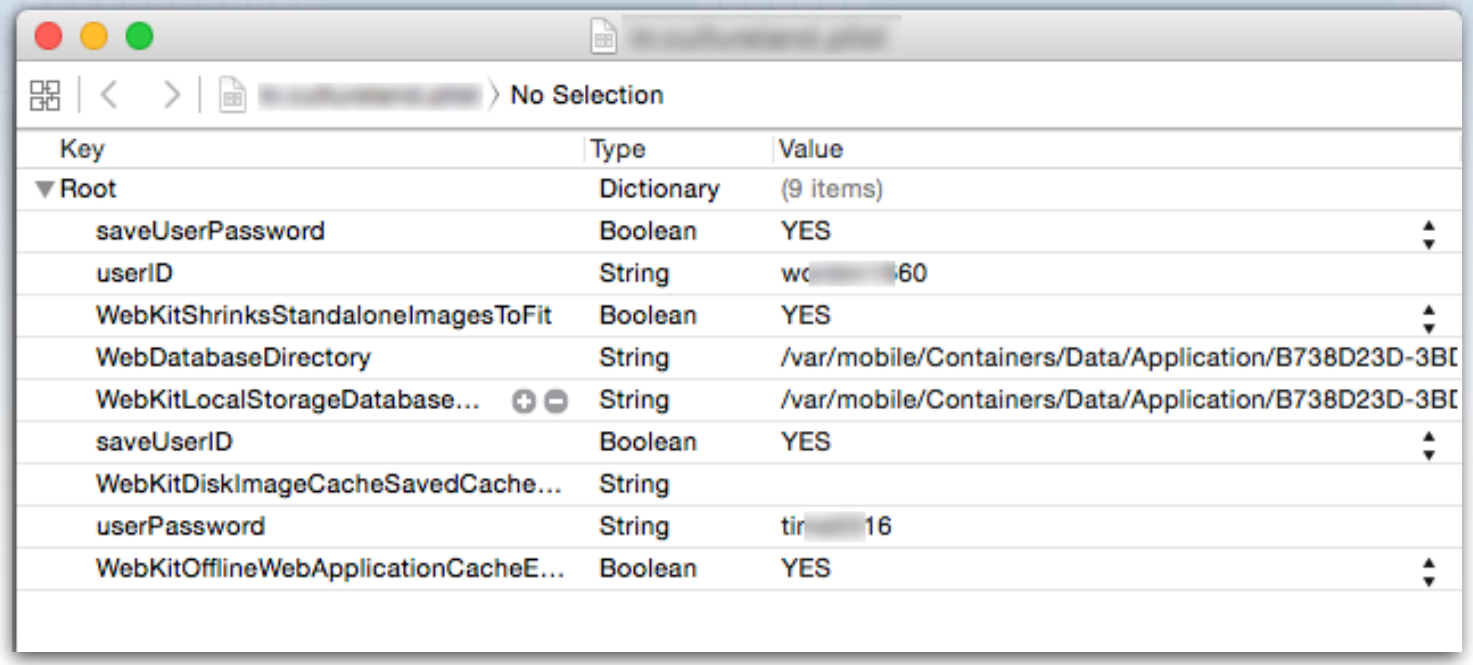
- 모바일 내 주요 정보저장
  - 평문 저장
  - 암호화 저장(암호화, 복호화 키 하드코딩)
  - 클라이언트에 암호화, 복호화 키 생성 로직
    - 사용자 고유정보, DUID정보 이용

```
= 0xa9d56;
= [[r0 encryptECB3DES:r6 key:@"20BCA802AB94AD928589164916AD9DC1"] retain
= 0xc3440;
0 = r8;
```

```
*(r7 + 0xfffffffffffffe0) = ATCrypto;
r0 = [ATCrypto cryptoWithKey:@"EncriptKeyASDFHD"];
*(r7 + 0xfffffffffffffb8) = r0;
```

Encryption File & Key Handling

- 환경설정(.plist, DB)
  - Login에 사용되는 ID, Pass가 여전히 평문으로 저장되는 앱들 존재

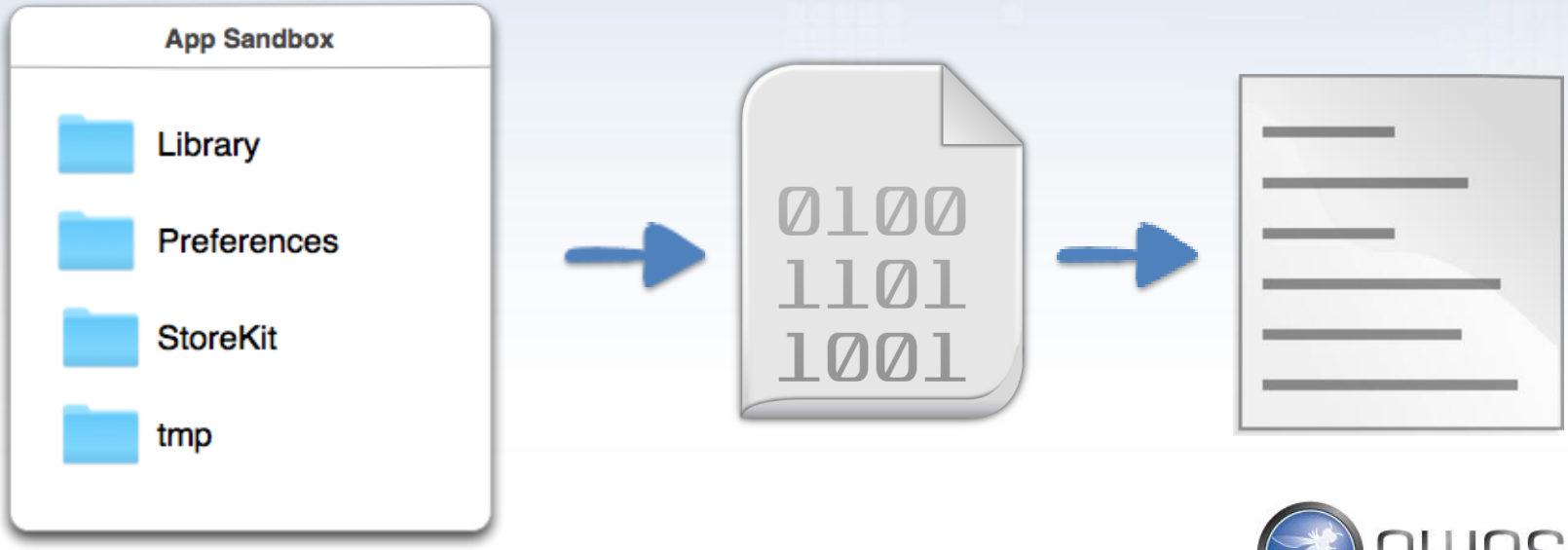


Encryption File & Key Handling

- 환경설정(.plist, DB)
  - 많은 경우 환경설정 파일에 암호화 되어 저장
    - 별도의 파일에 저장하기 전 환경설정 파일에 저장
    - 환경설정에 저장한 내용 미 삭제
  - Sandbox내 여러 형태의 정보 암호화 되어 저장
    - 페어링이 완료된 iDevice에서 AFC를 이용한 주요정보 탈취 문제점 야기

Encryption File & Key Handling

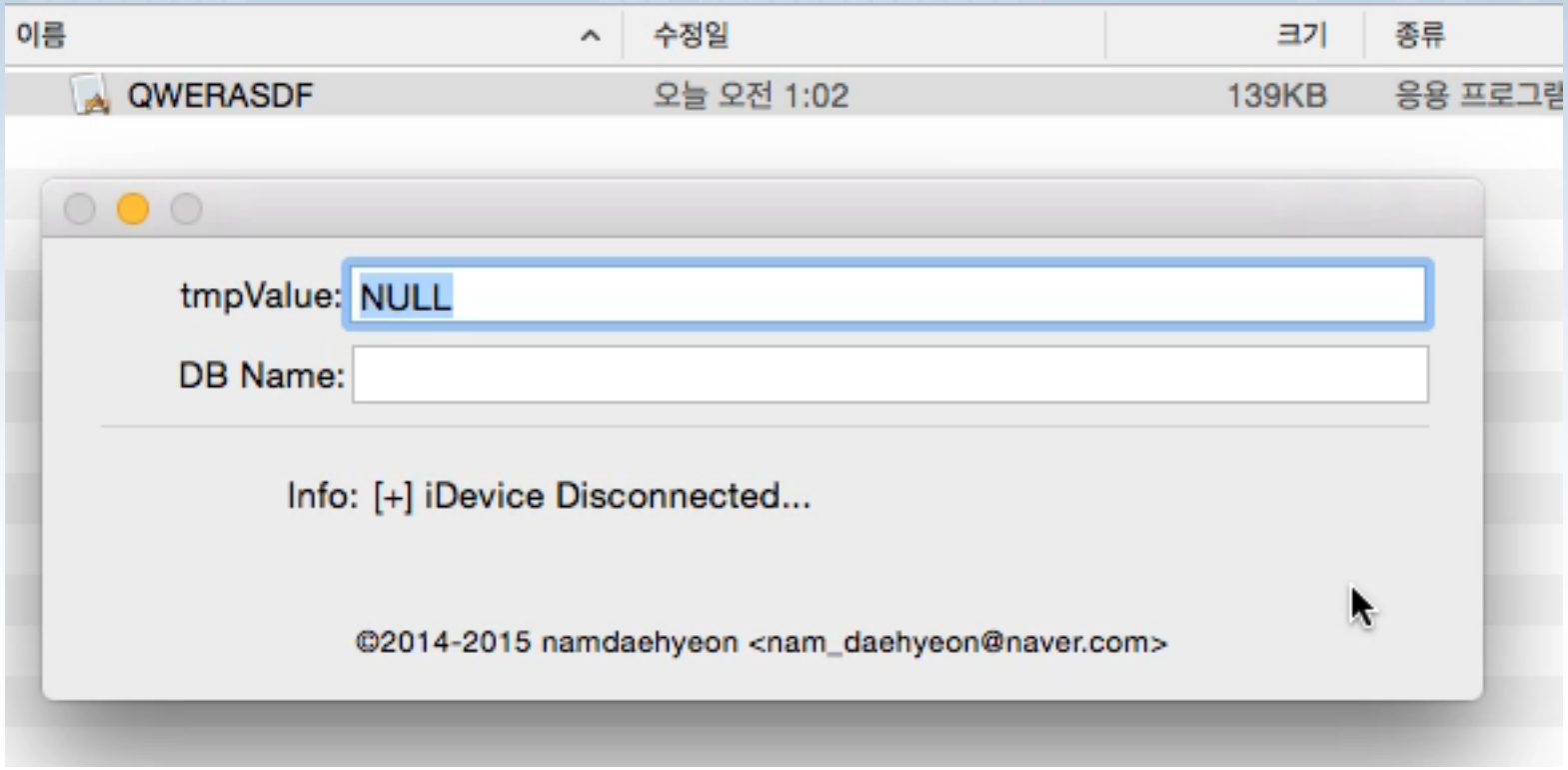
- Bundle identifier
- 암호화 되어 있는 파일만을 추출
- Runtime Abusing, Hooking, Reversing



OWASP
Open Web Application Security Project

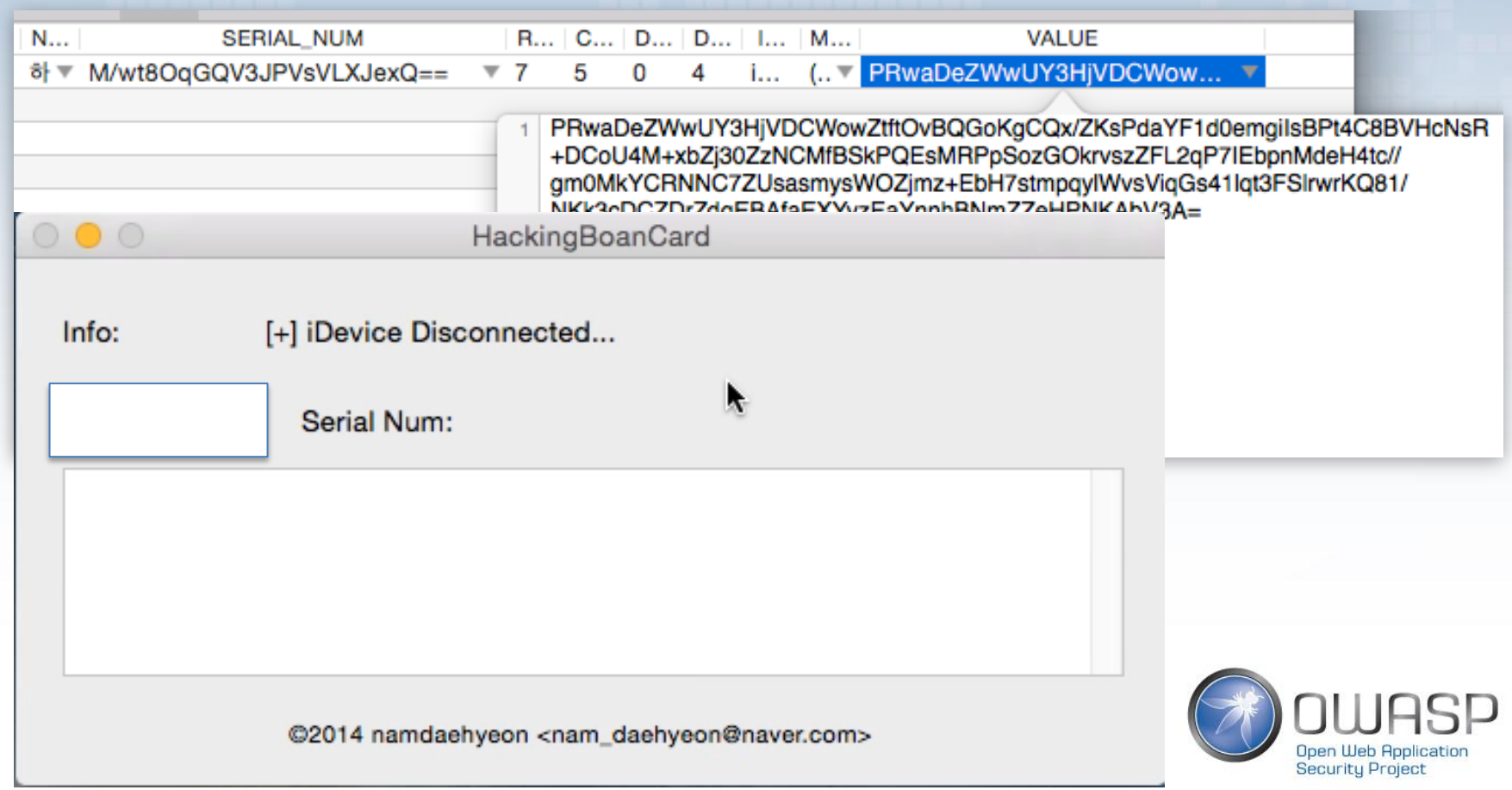
Encryption File & Key Handling

- DEMO



Encryption File & Key Handling

- DEMO



Encryption File & Key Handling

- Non-Jailbreak환경 (순정 iDevice)
  - 복호화에 사용되는 키 값 보다 Sandbox 내 암호화 되어 저장되는 파일이 문제
  - 애플정책 변경으로 iOS 8.3부터 AFC를 이용한 APP Sandbox 접근 불가 (5.x ~ 8.2는 접근가능)
  - APP실행 시 https통신을 통해 서버에 저장된 Key값을 받아 Keychain에 저장 후 주요 로직 암호화
  - 주요 정보는 반드시 Keychain에 저장

Q & A

References

- https://www.owasp.org
- https://www.theiphonewiki.com/wiki/Usbmux
- https://www.theiphonewiki.com/wiki/AFC
- Hacking apple accessories to pown iDevices – Wake up Neo! Your phone got pwnd ! by Mathieu GoToHack RENARD.pdf