iOS applications security testing cheat sheet

Oana Cornea

OWASP
The Open Web Application Security Project

About Me

- Oana Cornea

- Application Security Analyst at Electronic Arts, in Bucharest, Romania.

Agenda

OWASP
The Open Web Application Security Project

- Introduction
- iOS security model
- iOS application assessment
- Wrap up - Mobile risks

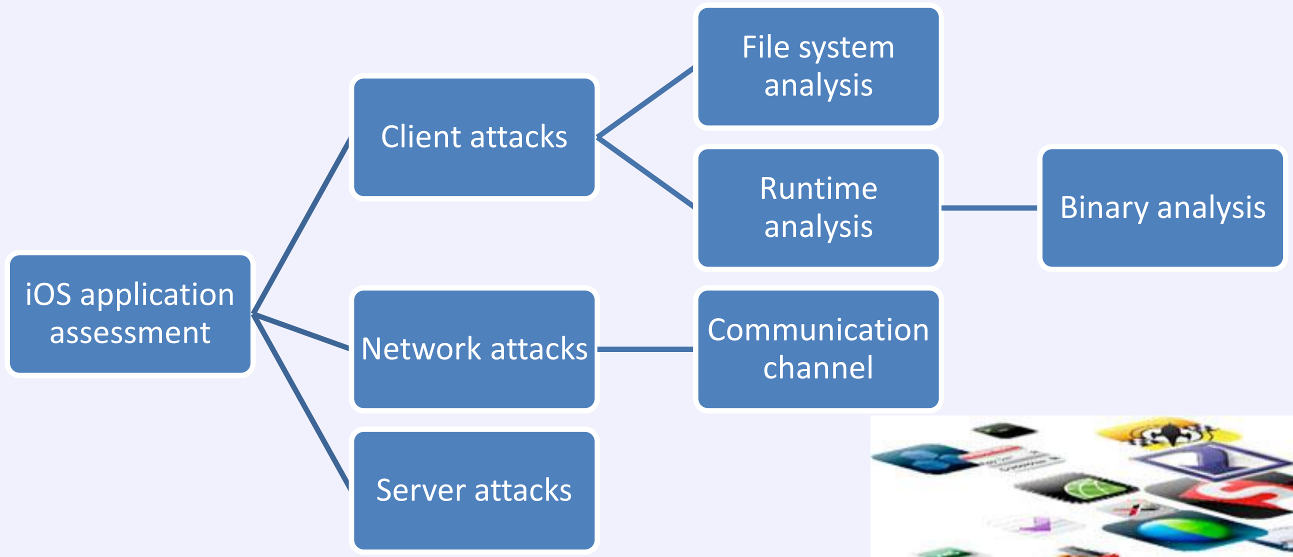
The iOS security model

OWASP
The Open Web Application Security Project

- Device security
- Data Security
- Network Security
- Application Security

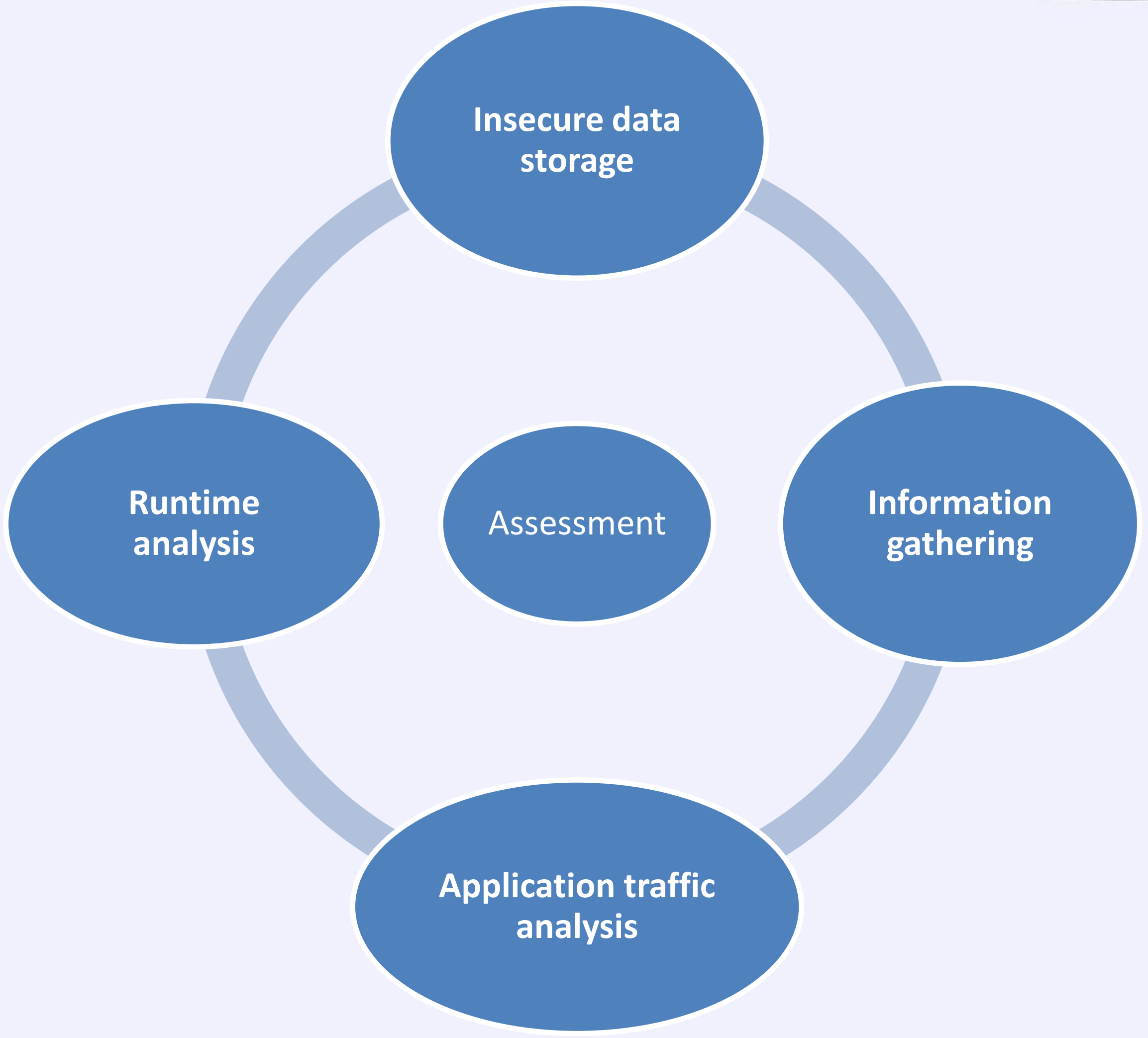
iOS application assessment

OWASP
The Open Web Application Security Project

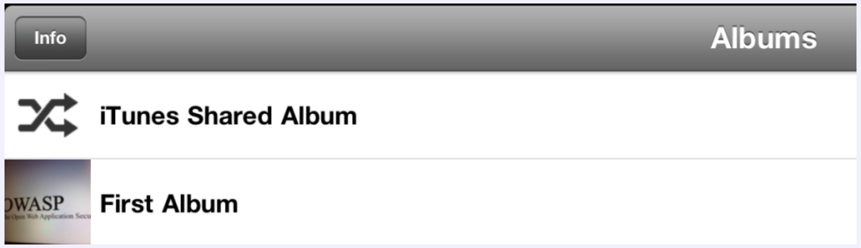


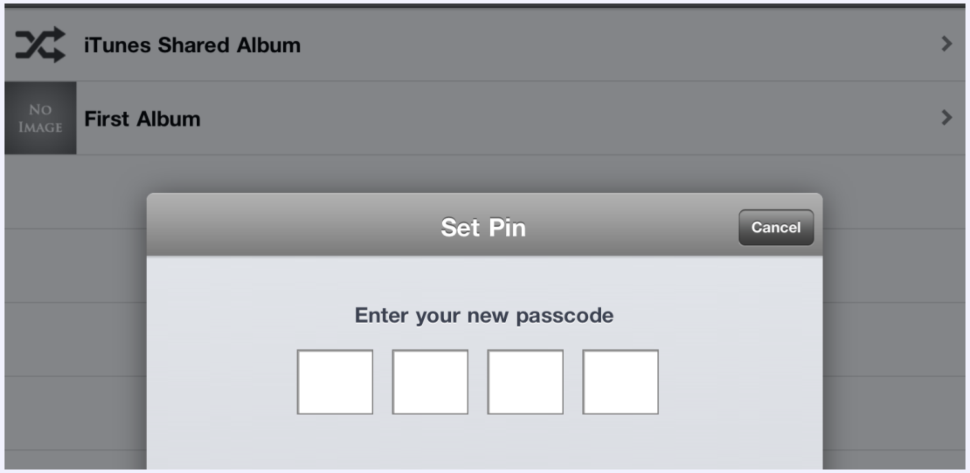
iOS application assessment

OWASP
The Open Web Application Security Project



Test application



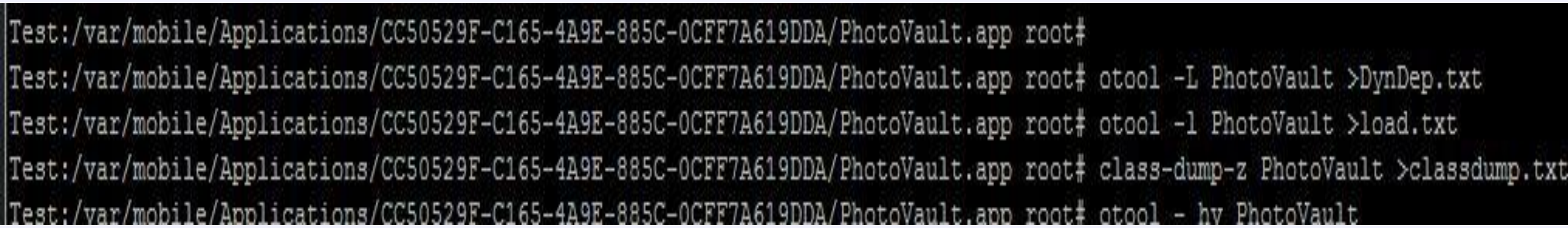


Information gathering

OWASP
The Open Web Application Security Project

- Observe application behavior
- Determine the application’s data states (at rest, in transit or on display) and sensitivity – in this case the sensitive photos are protected by a PIN.
- Identify
  - access methods
  - what frameworks are in use
  - server side APIs that are in use
  - what protocols are in use
  - other applications or services with which the application interacts

Information gathering



```
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root#
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# otool -L PhotoVault >DynDep.txt
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# otool -l PhotoVault >load.txt
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# class-dump-z PhotoVault >classdump.txt
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# otool - hv PhotoVault
```

- List the dynamic dependencies
- Dump the load commands for the application.
- Class dump

Application traffic analysis

- Intercept the traffic and analyze the requests and responses using a proxy: Burp, Charles, Mallory

Runtime analysis

OWASP
The Open Web Application Security Project

- Disassemble the application (gdb)
- Analyze file system interaction
- Analyze the application with a debugger (gdb): inspecting objects in memory and calling functions and methods; replacing variables and methods at runtime.

Runtime analysis

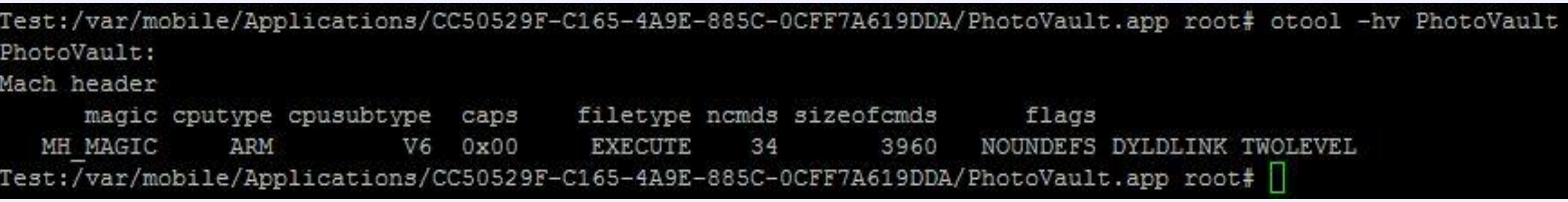
- Runtime analysis protecting features:
  - Locate the PIE (Position Independent Executable)

    Check this using the command: *otool –hv <app name>*
  - Stack smashing protection - specify the –fstack-protector-all compiler flag.

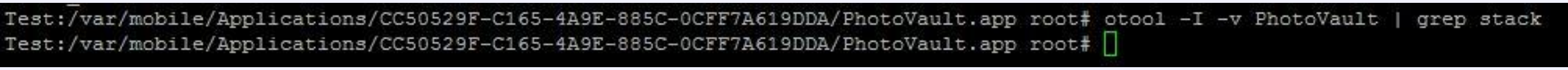
    Check this using: *otool –I –v <app name> | grep stack* .

    If the application was compiled with the stack smashing protection two undefined symbols will be present: "___stack_chk_fail" and "___stack_chk_guard".

Runtime analysis



```
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# otool -hv PhotoVault
PhotoVault:
Mach header
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
   MH_MAGIC     ARM         V6  0x00     EXECUTE    34       3960   NOUNDEFS DYLDLINK TWOLEVEL
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root#
```



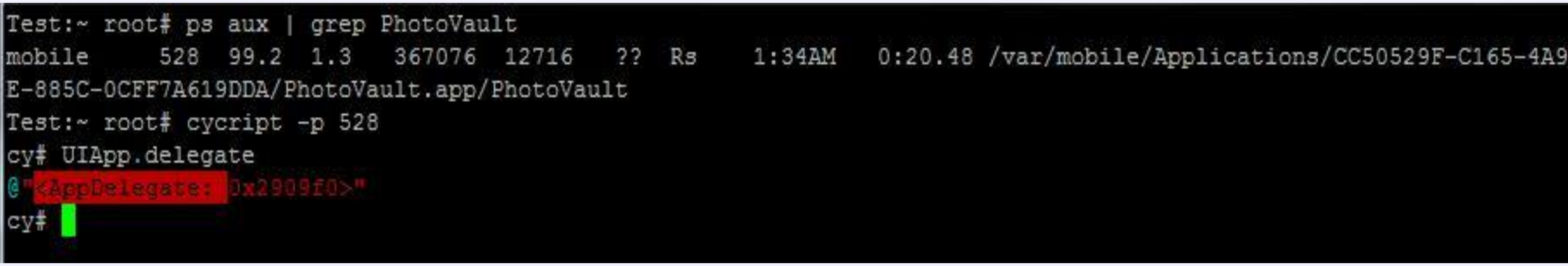
```
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root# otool -I -v PhotoVault | grep stack
Test:/var/mobile/Applications/CC50529F-C165-4A9E-885C-0CFF7A619DDA/PhotoVault.app root#
```

Runtime analysis

- Abusing the runtime with Cycript
- Abusing the runtime library – disassembling and debugging

Steps

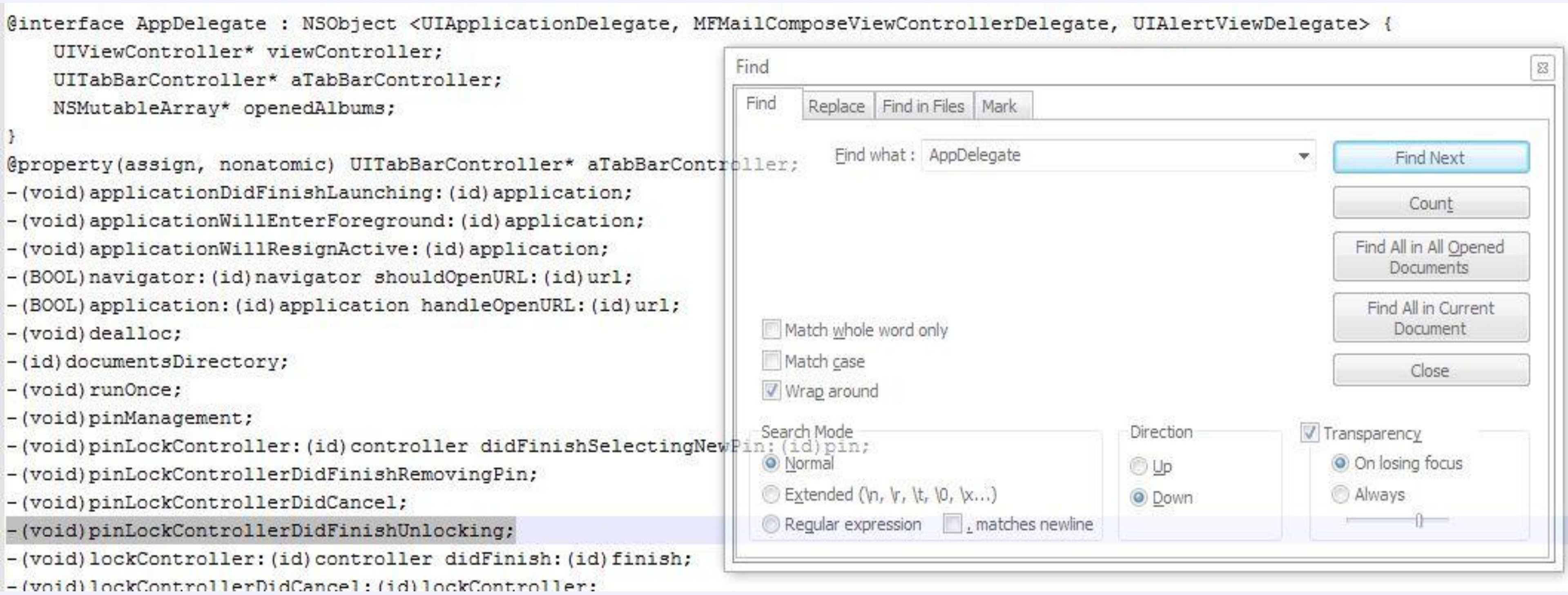
- Hook into the application process using *cycript –p [PID]* command.
- Grab the application delegate instance using *UIApp.delegate* command.



```
Test:~ root# ps aux | grep PhotoVault
mobile     528  99.2  1.3   367076  12716   ??  Rs    1:34AM   0:20.48 /var/mobile/Applications/CC50529F-C165-4A9
E-885C-0CFF7A619DDA/PhotoVault.app/PhotoVault
Test:~ root# cycript -p 528
cy# UIApp.delegate
@"<AppDelegate: 0x2909f0>"
cy# 
```

Steps

Search the class dump for AppDelegate and look for its interface.



```
@interface AppDelegate : NSObject <UIApplicationDelegate, MFMailComposeViewControllerDelegate, UIAlertViewDelegate> {
    UIViewController* viewController;
    UITabBarController* aTabBarController;
    NSMutableArray* openedAlbums;
}
@property(assign, nonatomic) UITabBarController* aTabBarController;
-(void)applicationDidFinishLaunching:(id)application;
-(void)applicationWillEnterForeground:(id)application;
-(void)applicationWillResignActive:(id)application;
-(BOOL)navigator:(id)navigator shouldOpenURL:(id)url;
-(BOOL)application:(id)application handleOpenURL:(id)url;
-(void)dealloc;
-(id)documentsDirectory;
-(void)runOnce;
-(void)pinManagement;
-(void)pinLockController:(id)controller didFinishSelectingNewPin:(id)pin;
-(void)pinLockControllerDidFinishRemovingPin;
-(void)pinLockControllerDidCancel;
-(void)pinLockControllerDidFinishUnlocking;
-(void)lockController:(id)controller didFinish:(id)finish;
-(void)lockControllerDidCancel:(id)lockController;
```

Find what : AppDelegate

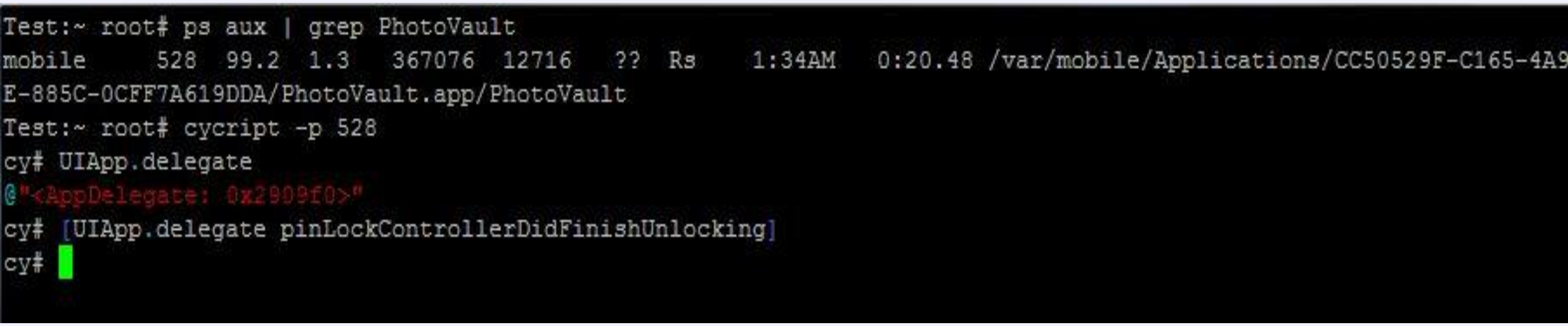
[ ] Match whole word only
[ ] Match case
[✓] Wrap around

Search Mode: [✓ Normal] Extended (\n, \r, \t, \0, \x...) Regular expression [ ] . matches newline

Direction: Up [✓ Down]

[✓] Transparency: [✓ On losing focus] Always

Steps



```
Test:~ root# ps aux | grep PhotoVault
mobile      528  99.2  1.3   367076  12716   ??  Rs    1:34AM   0:20.48 /var/mobile/Applications/CC50529F-C165-4A9
E-885C-0CFF7A619DDA/PhotoVault.app/PhotoVault
Test:~ root# cycript -p 528
cy# UIApp.delegate
@"<AppDelegate: 0x2909f0>"
cy# [UIApp.delegate pinLockControllerDidFinishUnlocking]
cy# 
```

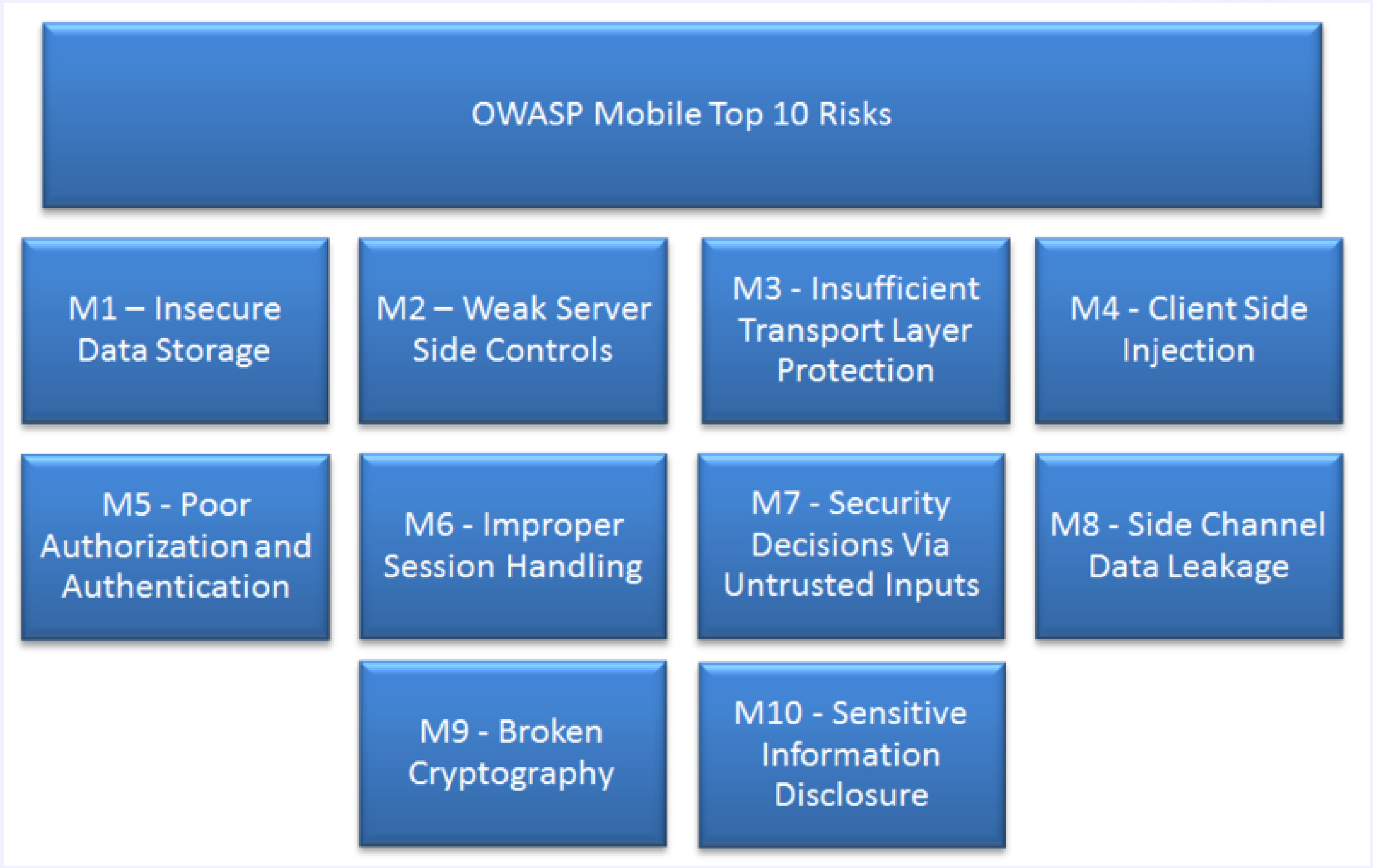
Insecure data storage

OWASP
The Open Web Application Security Project

- Log files
- Data storage in application folder
- SqlLite database
- Property list files
- File caching
- Keyboard cache
- Cookies.binarycookies
- iOS keychain
- Sensitive information in snapshots

Wrap up - Mobile risks

OWASP
The Open Web Application Security Project



OWASP Mobile Top 10 Risks

- M1 – Insecure Data Storage
- M2 – Weak Server Side Controls
- M3 - Insufficient Transport Layer Protection
- M4 - Client Side Injection
- M5 - Poor Authorization and Authentication
- M6 - Improper Session Handling
- M7 - Security Decisions Via Untrusted Inputs
- M8 - Side Channel Data Leakage
- M9 - Broken Cryptography
- M10 - Sensitive Information Disclosure

Wrap up - Mobile risks

- Insecure data storage
  - Avoid storing sensitive data on the device because any data stored locally could be compromised.
- Weak server side controls
  - Harden servers against malicious attacks
- Insufficient server side protection
  - Secure the communication

Wrap up - Mobile risks

OWASP
The Open Web Application Security Project

- Client side injection
  - Implement proper input validation
- Poor authorization and authentication
  - Avoid query string for sensitive data, institute local session timeout
- Improper session handling
  - Review the session management mechanism

Wrap up - Mobile risks

OWASP
The Open Web Application Security Project

- **Security decisions via untrusted inputs**
- The combination of input validation, output escaping, and authorization controls can be used against these weaknesses.

- **Side channel data leakage**
- Avoid crash logs, debug logs and caching app data.

- **Broken cryptography**
- Take advantage of what your platform already provides

- **Sensitive information disclosure**
- Anything that must truly remain private should not reside on the mobile device; keep private information (e.g., algorithms, proprietary information) on the server.

OWASP resources

OWASP
The Open Web Application Security Project

- iGoat
- MobiSec
- iMas
- Mobile Testing Guide

OWASP
The Open Web Application Security Project

Thank you!

- https://www.owasp.org/index.php/IOS_Application_Security_Testing_Cheat_Sheet
- https://www.owasp.org/index.php/OWASP_Mobile_Security_Project