Damn Vulnerable IOS Application Solutions
http://damnvulnerableiosapp.com/

Jailbreak Detection – Jailbreak Test 1

Dump the class information for this application by navigating to the folder where the application binary is located and using the command *class-dump DamnVulnerableIOSApp*

```
Prateeks-iPhone:/var/mobile/Applications/A9E7B7D9-4032-44C0-952F-5636D3A0EC9A root# ls
DamnVulnerableIOSApp.app/  Documents/  Library/  tmp/
Prateeks-iPhone:/var/mobile/Applications/A9E7B7D9-4032-44C0-952F-5636D3A0EC9A root# cd DamnVulnerableIOSApp.app/
Prateeks-iPhone:/var/mobile/Applications/A9E7B7D9-4032-44C0-952F-5636D3A0EC9A/DamnVulnerableIOSApp.app root# ls
Assets.car             Info.plist                    PkgInfo              card-bg.png              en.lproj/          main-bg.png      menu-icon\@2x.png      slider-bg.png
Base.lproj/            LaunchImage-700-568h\@2x.png  ResourceRules.plist  card-bg\@2x.png          header-bg.png      main-bg\@2x.png  slider-active.png      slider-bg\@2x.png
DamnVulnerableIOSApp*  Model.momd/                   _CodeSignature/      embedded.mobileprovision  header-bg\@2x.png  menu-icon.png    slider-active\@2x.png
Prateeks-iPhone:/var/mobile/Applications/A9E7B7D9-4032-44C0-952F-5636D3A0EC9A/DamnVulnerableIOSApp.app root# class-dump DamnVulnerableIOSApp
/*
 *     Generated by class-dump 3.1.2.
 *
 *     class-dump is Copyright (C) 1997-1998, 2000-2001, 2004-2007 by Steve Nygard.
 */

struct CGPoint {
    float _field1;
    float _field2;
};

struct CGRect {
    struct CGPoint _field1;
    struct CGSize _field2;
};

struct CGSize {
    float _field1;
    float _field2;
};
```

On scrolling down a bit, we see this function – *(BOOL)isJailbroken;* in the class JailbreakDetectionVC that returns a BOOL value.

```
@interface JailbreakDetectionVC : /Users/Prateek/Desktop/DVIA/DamnVulnerableIOSApp/DamnVulnerableIOSApp/View Controllers/
{
}

- (BOOL)isJailbroken;
- (void)jailbreakTest2Tapped:(id)fp8;
- (void)jailbreakTest1Tapped:(id)fp8;
- (void)readArticleTapped:(id)fp8;
- (void)didReceiveMemoryWarning;
- (void)viewDidLoad;
- (id)initWithNibName:(id)fp8 bundle:(id)fp12;

@end
```

Looks like this method is the one that is used to check whether a device is jailbroken or not. If we modify the implementation of this method to return *NO*, then our task will be accomplished.

Let's use cycript to overwrite this method's implementation. First, lets hook into the application by finding the process ID of the application and using the command *cycript -p PID***.** Make sure the application is running in foreground or we won't be able to hook into this application.

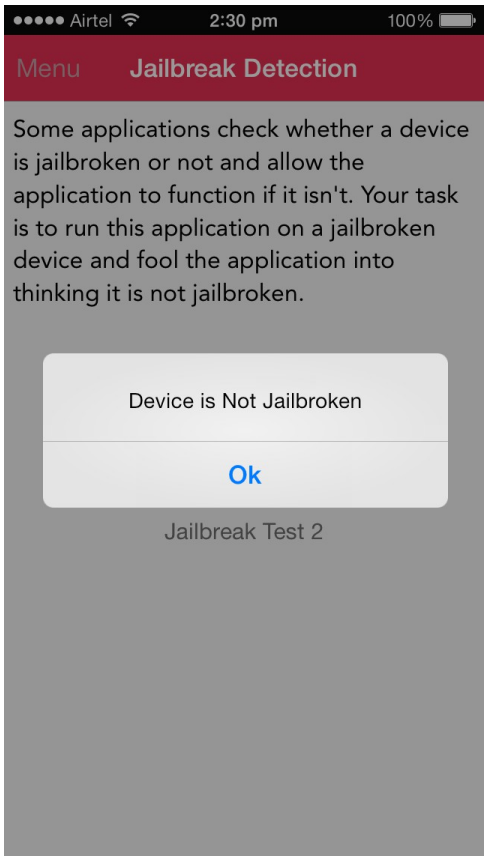
```
Prateeks-iPhone:/var/mobile/Applications root# ps aux | grep "Damn"
root      2307   0.0  0.0   338632    512 s001  S+    2:24PM   0:00.01 grep Damn
mobile    2302   0.0  1.3   397716  13164   ??  Ss    2:23PM   0:00.52 /var/mobile/Applications/A9E7B7D9-4032-44C0-952F-5636D3A0EC9A/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
Prateeks-iPhone:/var/mobile/Applications root# cycript -p 2302
cy# UIApp
@"<UIApplication: 0x16560b30>"
cy#
```

Now modify the implementation by using this command in the cycript interpreter

```
JailbreakDetectionVC.messages['isJailbroken'] = function () {return NO};
```

```
cy# JailbreakDetectionVC.messages['isJailbroken'] = function () {return NO};
{}
cy#
```

And now, if you tap on the button *Jailbreak Test 1* in the application, you will see that the alert says "Device is not jailbroken" even though the device I am currently running the application on is actually jailbroken.



And you will notice that if you tap on *Jailbreak Test 2*, you will see an alert that says "Device is jailbroken". Looks like our fix didn't work for the second test. To know how to bypass the check for jailbreak in that case, please have a look at the solutions for *Jailbreak Test 2*.