

# iNalyzer User Guide

## Contents

iNalyzer Installation.....	2
Packaging with iNalyzer.....	8
On Your Device.....	8
On Your Laptop.....	9
Using Windows.....	11
Using Linux/Mac.....	12
Static Analysis Using iNalyzer.....	13
Summary.....	18
iNalyzer With Unencrypted Applications.....	19
Runtime Analysis Using iNalyzer.....	21
Running iNalyzer With Burp Proxy.....	24
Note.....	29
Useful Links.....	29

## iNalyzer Installation

The first step is to install iNalyzer on your jailbroken device. You can do this by visiting Cydia.

Go to Cydia → Manage (make sure the following source URL is added as shown below: <http://appsec-labs.com/cydia>)

1. Open Cydia as shown below:



2. Go to the **Manage** tab and tap **Settings** in the top menu:



3. Tap **Developer** and then **Done**:



4. Go to **Manage** and tap **Sources**:



5. Go to **Edit** → **Add**, and insert the following URL: <https://appsec-labs.com/cydia>:



6. Wait for the update to complete.

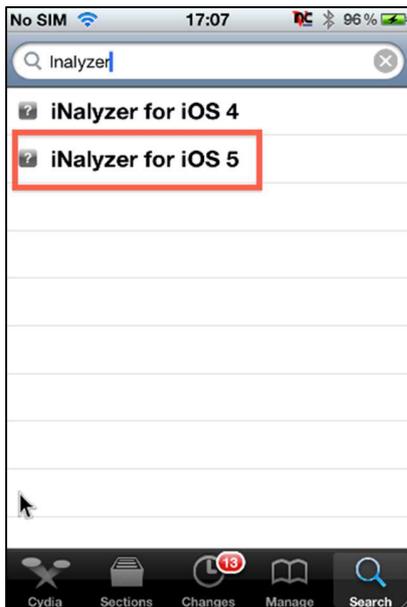


Now go to **Search** and search for iNalyzer. Depending on the iOS version that you are running, you should download the corresponding version.

7. Once **AppSec-Labs.com** is listed, tap **Search**.



8. In the **Search** bar, type **iNalyzer** and tap the version corresponding to your iOS version.



9. In the **Details** tab, tap **Install**.



10. In the **Confirmation** tab, tap **Confirm**.



11. Wait until the installation completes.



12. Tap Return to Cydia.



Once installation is complete, power-cycle your device by turning it off and then on again.

-  **NOTE:** iNalyzer is installed in the **/Applications** directory because it needs to run as a root user. See the following image:

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ ssh root@10.0.1.23
root@10.0.1.23's password:
Prateeks-iPod:~ root# cd /Applications/iNalyzer5.app/
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

## Packaging with iNalyzer

### On Your Device

1. Look for the iNalyzer5 icon:



2. Tap the icon so the iNalyzer listener will load. Once loaded, the iNalyzer listener port (:5544) will be added to the icon:



3. Alternately, SSH ((Secure Shell) make a secure connection over an insecure network) to the device and run /iNalyzer to start it. Run it again to stop the process.

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ls
CodeDirectory                dox.template*          logUI.
CodeEntitlements            doxMe.sh*              logUI.cy
CodeRequirements           footer.html*           logo.gif
CodeResources              helper_init.cy         logo.png
CodeSignature              iNalyzer*              mi.py*
Info.plist                 iNalyzer-Daemon.sh*   msgD.txt*
Menu.sh*                   iNalyzer5*            msgR.txt*
PkgInfo                    iNalyzer5.entitlements packApp.sh*
ResourceRules.plist        iNalyzer5.xcent       proc.kill
_CodeSignature/            icon.png               proc.run
com.appsec-labs.inalyzer5.Shutdown.plist libsubjc.cy           r30c5.sh
com.appsec-labs.inalyzer5.Startup.plist  listApp.sh*          utils.cy*
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

4. Locate your device's IP address by following these steps:

- 📱 Open the **Settings** application:



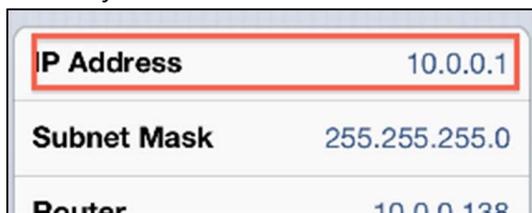
- Tap the **WiFi Settings**:



- Tap **Advanced Settings**:

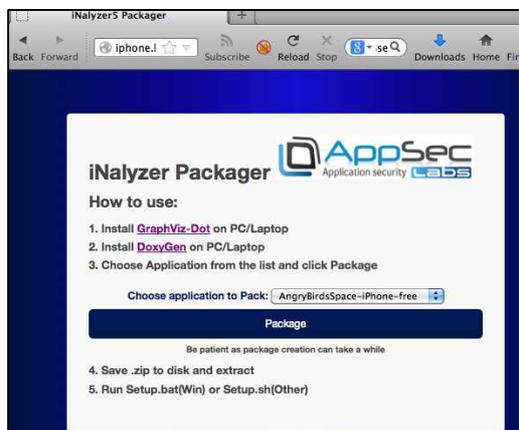


- Locate your device's IP address and write it down:



## On Your Laptop

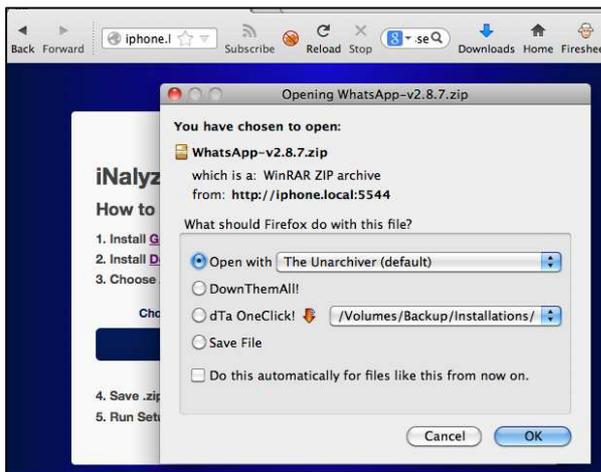
1. Verify that **GraphViz-Dot** is installed (pre-installed in the VM image).
2. Verify that **DoxyGen** is installed (pre-installed in the VM image).
3. Open **Firefox** to this address <http://<yourDeviceIP>:5544> to receive the iNalyzer Packager welcome screen:



4. Choose **Application to Package** and click the **Package** button.



5. Wait for the process to complete, at which point you will be prompted to download a ZIP file.

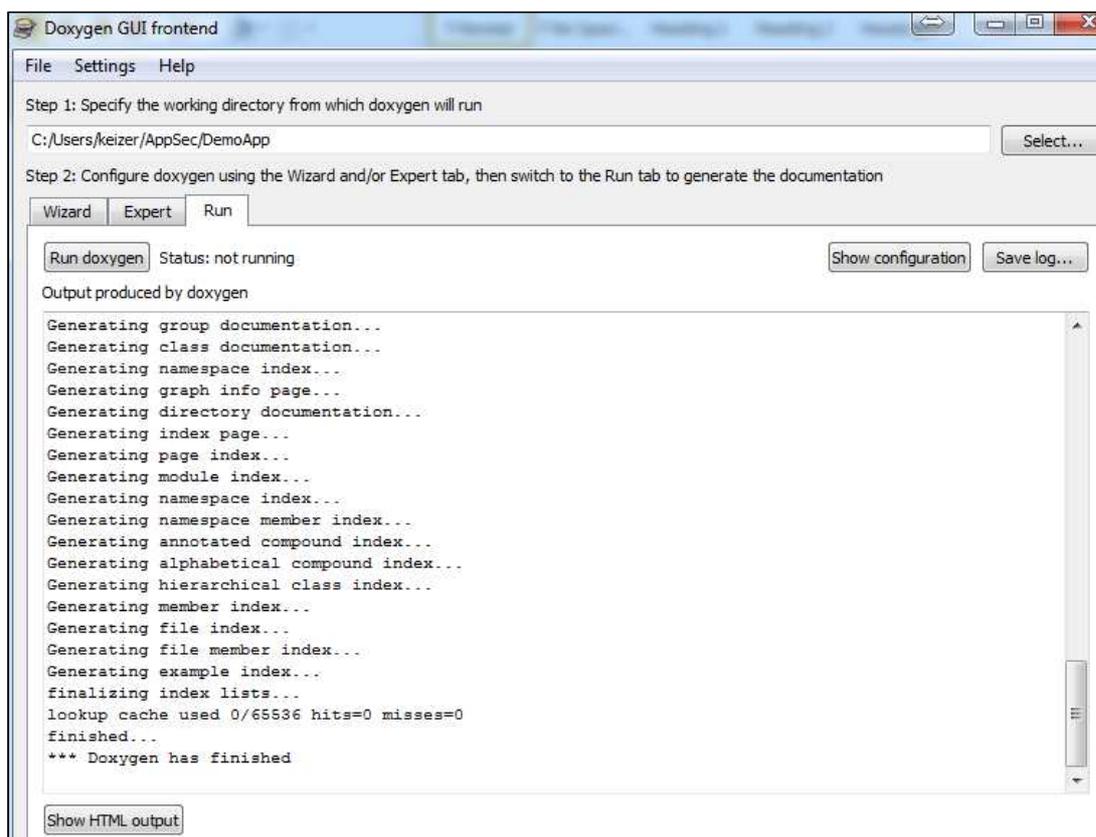


6. Extract the content to a destination folder on your desktop.

## Using Windows

1. Open the **DoxyGen** wizard and load the dox.template file form the extracted payload/DoxyGen folder:

- ☐ Switch to the **Run** sub-tab, and click **Run doxygen**.
- ☐ Wait for the process to complete and click **Show HTML output** at the bottom of the list.



## Using Linux/Mac

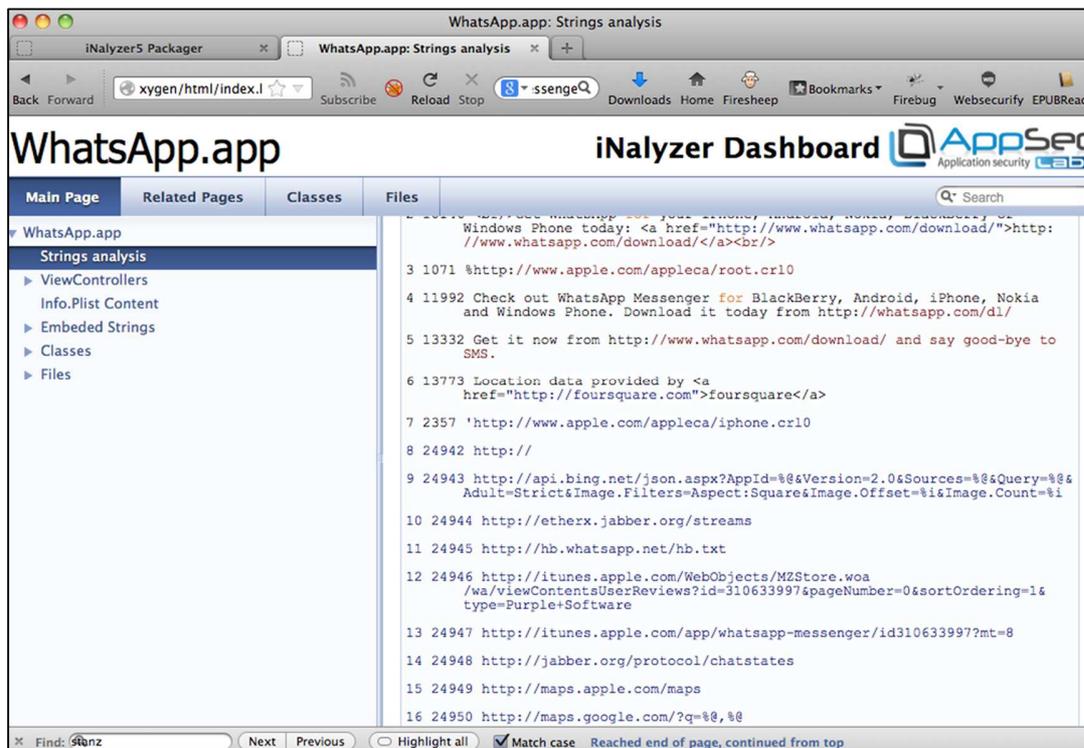
1. Execute the Doxygen engine on the /Payload/Doxygen/dox.template file. For example: `doxygen/[Path To Extracted ZIP]/Payload/Doxygen/dox.template`.

☐ Alternately, you can edit and run the doxMe.sh script.

```

coreDumps-MacBook-Pro-2:Doxygen _coredump$ ls -l
total 64
-rwxr-xr-x@ 1 _coredump  staff  11628 17:31 10 מפר dox.template
-rwxr-xr-x@ 1 _coredump  staff    103 07:33 8 מפר doxMe.sh
-rwxr-xr-x@ 1 _coredump  staff   7201 17:31 10 מפר footer.html
-rwxr-xr-x@ 1 _coredump  staff   6799 07:33 8 מפר logo.gif
coreDumps-MacBook-Pro-2:Doxygen _coredump$
  
```

2. Wait for the process to complete and open the /Doxygen/html/index.html file with Firefox.
3. Inspect the iNalyzer dashboard for the application:



## Static Analysis Using iNalyzer

In this section, we will look at the iNalyzer and how we can use it for black box assessment of iOS applications. iNalyzer allows us to view the class information, perform runtime analysis, and many other actions. iNalyzer automates the efforts for decrypting the application, dumping class information, and presents it in a much more logical way.

iNalyzer requires some dependencies be installed before use, as explained in the previous section. Please make sure to install GraphViz and DoxyGen as iNalyzer will not function without these programs available on your device.

1. Navigate inside the iNalyzer directory and run /iNalyzer5 without any arguments. You will then see the full list of applications available for analysis.

```
Prateeks-iPod:/Applications/iNalyzer5.app root# pwd
/Applications/iNalyzer5.app
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5
usage: ./iNalyzer5 [application name] [...]
Applications available: 650_379_weather AngryBirdsBlack-iPhone AngryBirdsRio
Free AngryBirdsSpace-iPhone ATN BatteryDoctorLite BookMyShow CloudReaders De
fcon GeoDoIt GmailHybrid iBooks McAfee Threat Alert NASA NASA TV Path Shazam
Snapchat Spotify TED Whiteboard Wikitude WordPress
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

2. Choose the app you want to test and pass it as an argument to iNalyzer5. iNalyzer will begin its work by decrypting the app, discovering the class information, etc. The following steps will demonstrate the stages as pertains to the Defcon application.

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5 Defcon
got params /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/ Defcon.app 11 iNalyzer is iNalyzing
Defcon...
iNalyzer:crack_binary got /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/Defcon /tmp/iNalyzer5_
327b2245/Payload/Defcon.app/Defcon Dumping ARMV7 portion...helloooo polis?
iNalyzer:Creating Snapshot into ClientFiles
iNalyzer:Snapshot Done
iNalyzer:Population Done
iNalyzer:Dumping Headers
iNalyzer:Patching Headers
Compressing decrypted application (2/2).. [=====] ] 23%
```

3. When iNalyzer5 finishes its job, it will create an .ipa file (an iOS application archive file that stores an iOS app). It will be stored in the following location:

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5 Defcon
got params /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/ Defcon.app 11 iNalyzer is iNalyzing
Defcon...
iNalyzer:crack_binary got /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/Defcon /tmp/iNalyzer5_
327b2245/Payload/Defcon.app/Defcon Dumping ARMV7 portion...helloooo polis?
iNalyzer:Creating Snapshot into ClientFiles
iNalyzer:Snapshot Done
iNalyzer:Population Done
iNalyzer:Dumping Headers
iNalyzer:Patching Headers
/var/root/Documents/iNalyzer/Defcon-v1.1.ipa
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

4. Obtain this .ipa file and download it through our system. Navigate inside the following folder: Payload/Doxygen:

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ cd Desktop/Dumped\ Apps/Defcon-v1.1/Payload/Doxygen/
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$ ls
dox.template  doxMe.sh      footer.html   logo.gif
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$
```

5. You will see a shell script named doxMe.sh. There is a running automated DoxyGen task running inside this script. DoxyGen also runs GraphViz for generating graphs. The results are stored inside a folder named HTML. iNalyzer has already stored all the class information for us inside a folder named "Reversing Files" and uses DoxyGen and GraphViz to display the information in a much more presentable format. In addition, this shell script opens up the index.html file inside the created HTML folder.

```
#!/bin/sh

/Applications/Doxygen.app/Contents/Resources/doxygen dox.template && open ./html/index.html
~
~
~
~
```

6. Run this shell script and let iNalyzer finish:

```
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$ ./doxMe.sh
Warning: Tag 'SYMBOL_CACHE_SIZE' at line 55 of file dox.template has become obsolete.
To avoid this warning please remove this line from your configuration file or upgrade it using "doxygen -u"
Searching for include files...
Searching for example files...
Searching for images...
Searching for dot files...
Searching for msc files...
Searching for files to exclude
Searching for files to process...
Searching for files in directory /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles
Reading and parsing tag files
Parsing files
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h...
/Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h:38: warning: reached end
_of comment while inside a @code block; check for missing @endcode tag!
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__InfoPlist__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__InfoPlist__.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__String_dump__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__String_dump__.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/Bio.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/Bio.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/DC17TableNavItem.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/DC17TableNavItem.h...
```

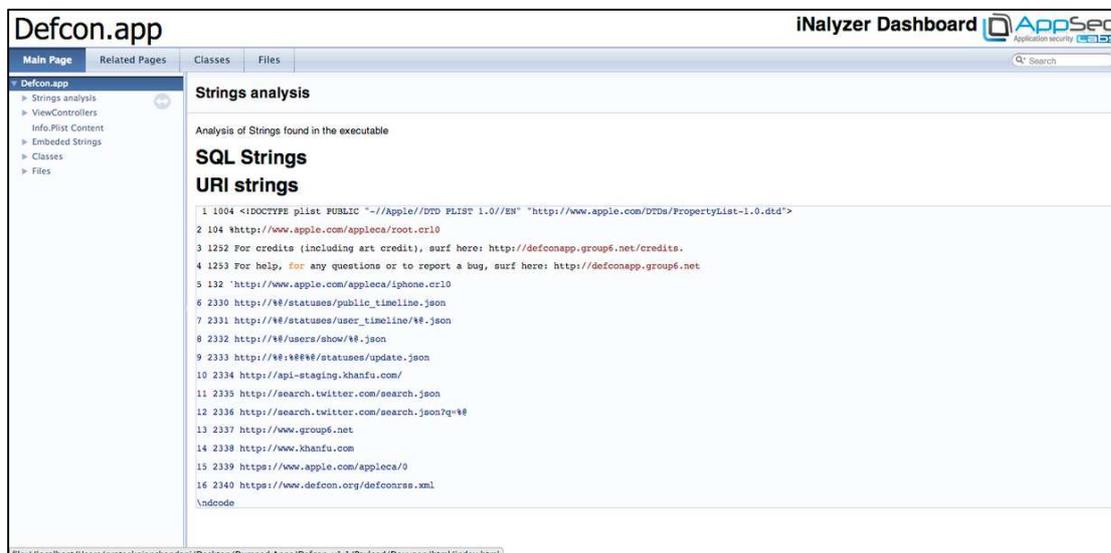
```

Patching output file 84/105
Patching output file 85/105
Patching output file 86/105
Patching output file 87/105
Patching output file 88/105
Patching output file 89/105
Patching output file 90/105
Patching output file 91/105
Patching output file 92/105
Patching output file 93/105
Patching output file 94/105
Patching output file 95/105
Patching output file 96/105
Patching output file 97/105
Patching output file 98/105
Patching output file 99/105
Patching output file 100/105
Patching output file 101/105
Patching output file 102/105
Patching output file 103/105
Patching output file 104/105
Patching output file 105/105
lookup cache used 905/65536 hits=2587 misses=932
finished...
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$

```

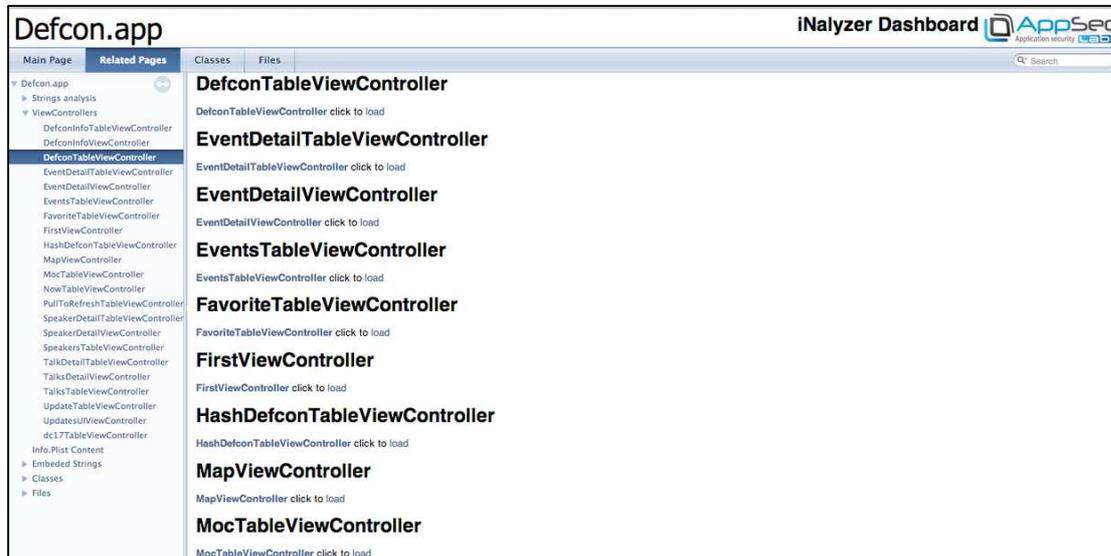
Once iNalyzer is done it will automatically open up the **index.html** file stored inside the HTML folder that was created. It is recommended to use the Firefox browser for runtime analysis as the other browsers may not support all of its features.

 The first page gives a string analysis of the entire app. It divides the strings into SQL and URL strings.



The screenshot shows the iNalyzer Dashboard interface for 'Defcon.app'. The main content area is titled 'Strings analysis' and displays a list of strings found in the executable, categorized into 'SQL Strings' and 'URI strings'. The list includes various URLs and file paths, such as 'http://www.apple.com/DRDs/PropertyList-1.0.dtd' and 'http://www.apple.com/appleca/root.cr10'. The interface also shows a sidebar with navigation options like 'Main Page', 'Related Pages', 'Classes', and 'Files'.

🔗 You can review the view controller classes used in the app.

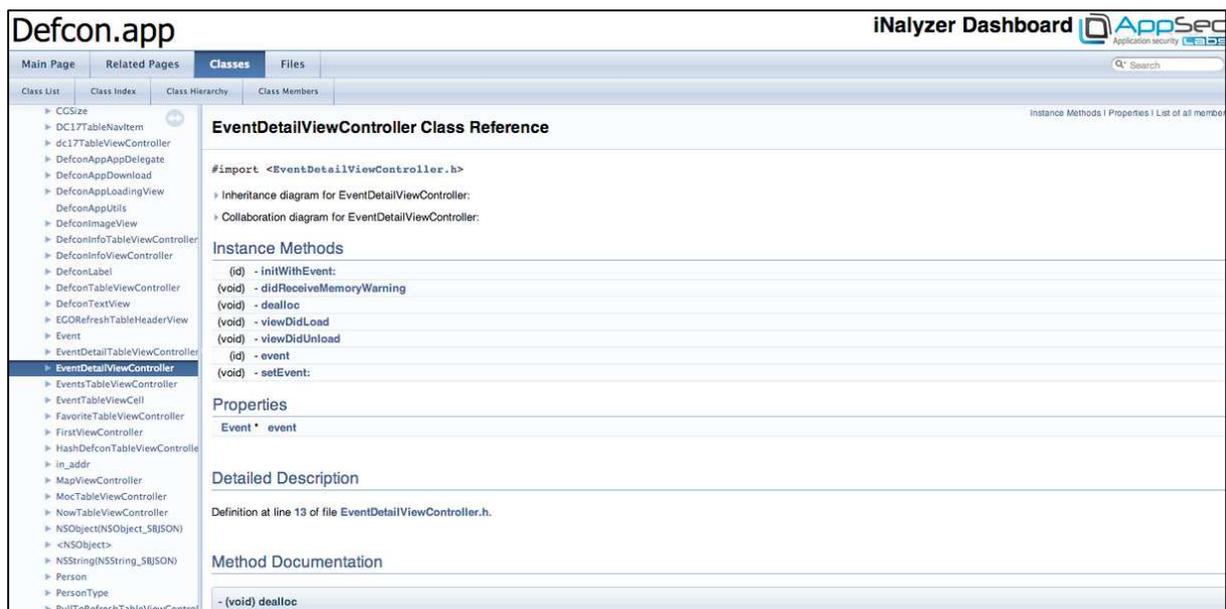


The screenshot shows the iNalyzer Dashboard interface for 'Defcon.app'. The 'Classes' tab is active, displaying a list of view controller classes. The classes listed are:

- DefconTableViewController
- EventDetailTableViewController
- EventDetailViewController
- EventsTableViewController
- FavoriteTableViewController
- FirstViewController
- HashDefconTableViewController
- MapViewController
- MocTableViewController

Each class name is followed by a 'click to load' link. The left sidebar shows a tree view of the application's structure, with 'ViewControllers' expanded to show the list of classes.

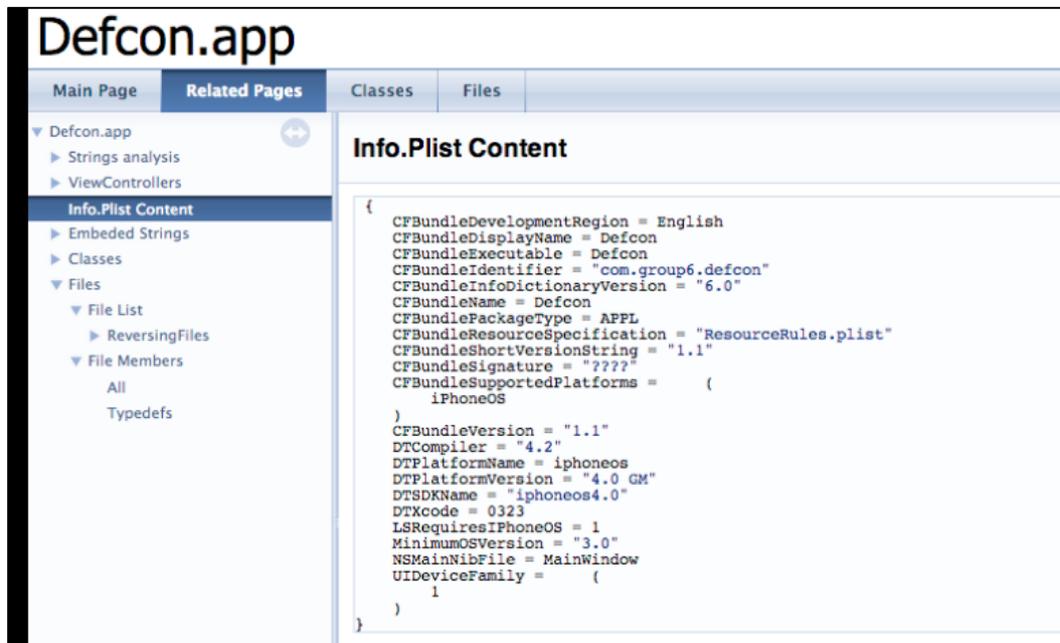
🔗 Clicking on any of the view controllers will show you its methods and properties.



The screenshot shows the iNalyzer Dashboard interface for 'Defcon.app', displaying the class reference for 'EventDetailViewController'. The 'Classes' tab is active, and the 'Class Reference' sub-tab is selected. The class reference page includes the following sections:

- Class Reference:** Shows the class name and a link to the class reference.
- Import:** Shows the import statement: `#import <EventDetailViewController.h>`
- Inheritance diagram:** Shows the inheritance diagram for EventDetailViewController.
- Collaboration diagram:** Shows the collaboration diagram for EventDetailViewController.
- Instance Methods:** Lists the instance methods:
  - (id) - initWithEvent:
  - (void) - didReceiveMemoryWarning
  - (void) - dealloc
  - (void) - viewDidLoad
  - (void) - viewDidUnload
  - (id) - event
  - (void) - setEvent:
- Properties:** Lists the properties:
  - Event \* event
- Detailed Description:** Shows the definition at line 13 of file EventDetailViewController.h.
- Method Documentation:** Shows the method documentation for the `- (void) dealloc` method.

📄 The contents of the **info.plist** file provides vulnerable information.



**Defcon.app**

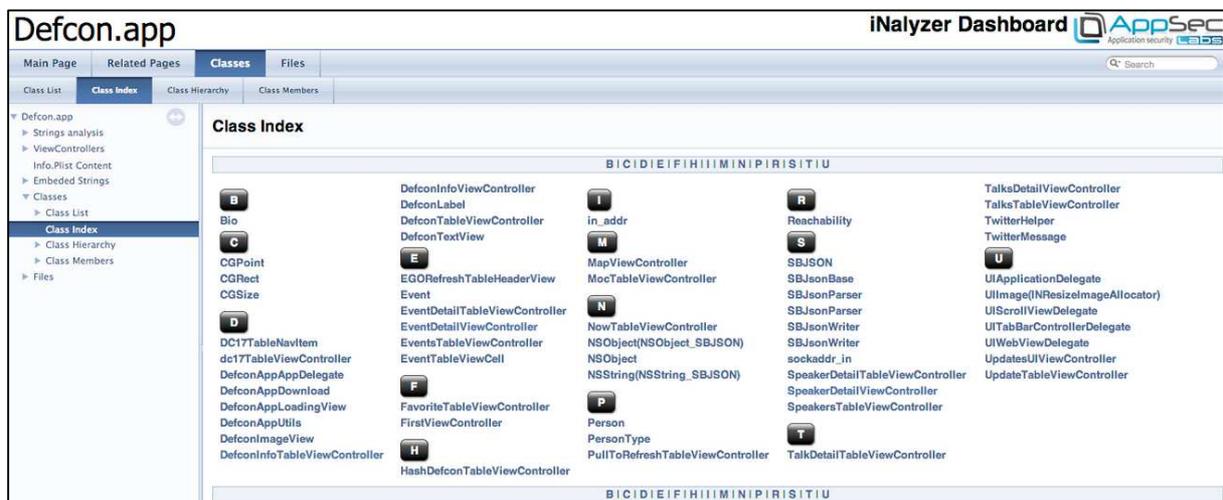
Main Page | **Related Pages** | Classes | Files

- Defcon.app
  - Strings analysis
  - ViewControllers
  - Info.Plist Content**
  - Embedded Strings
  - Classes
  - Files
    - File List
      - ReversingFiles
    - File Members
      - All
      - Typedefs

**Info.Plist Content**

```
{
  CFBundleDevelopmentRegion = English;
  CFBundleDisplayName = Defcon;
  CFBundleExecutable = Defcon;
  CFBundleIdentifier = "com.group6.defcon";
  CFBundleInfoDictionaryVersion = "6.0";
  CFBundleName = Defcon;
  CFBundlePackageType = APPL;
  CFBundleResourceSpecification = "ResourceRules.plist";
  CFBundleShortVersionString = "1.1";
  CFBundleSignature = "????";
  CFBundleSupportedPlatforms = (
    iPhoneOS
  );
  CFBundleVersion = "1.1";
  DTCompiler = "4.2";
  DTPlatformName = iphoneos;
  DTPlatformVersion = "4.0 GM";
  DTSDKName = "iphoneos4.0";
  DTXCcode = 0323;
  LSRequiresIPhoneOS = 1;
  MinimumOSVersion = "3.0";
  NSMainNibFile = MainWindow;
  UIDeviceFamily = (
    1
  );
}
```

📄 In the **Class Hierarchy** tab you will see the class information and relationship in a graphical format. This gives us a good understanding of how this application works. These graphs are generated by the Graphvis tool.



**Defcon.app** | iNalyzer Dashboard | AppSec Labs

Main Page | Related Pages | **Classes** | Files

Class List | **Class Index** | Class Hierarchy | Class Members

Defcon.app

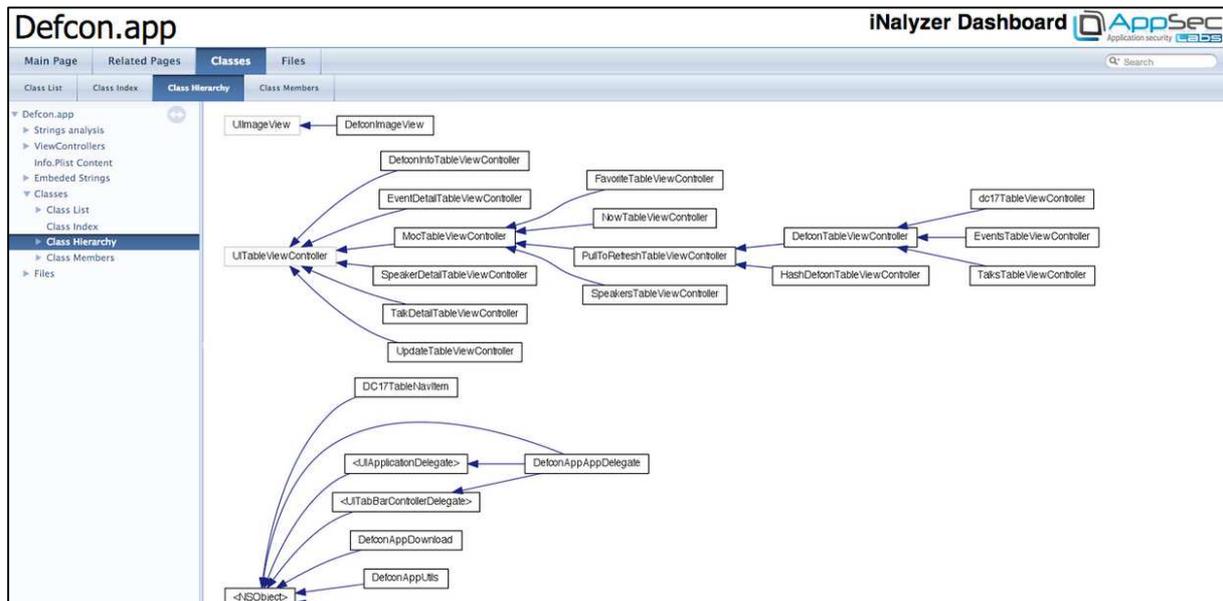
- Strings analysis
- ViewControllers
- Info.Plist Content
- Embedded Strings
- Classes
  - Class List
  - Class Index**
  - Class Hierarchy
  - Class Members
- Files

**Class Index**

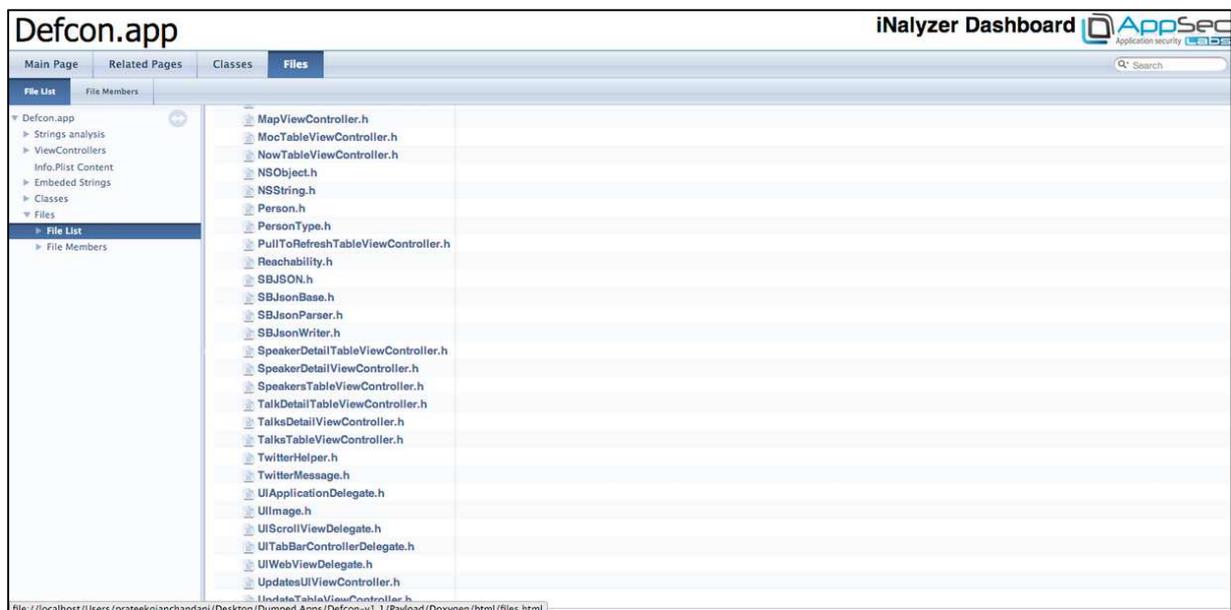
B I C I D I E I F I H I I I M I N I P I R I S I T I U

<b>B</b> Bio	DefconInfoViewController DefconLabel DefconTableViewController DefconTextView	<b>I</b> in_addr	<b>R</b> Reachability	TalksDetailViewController TalksTableViewController TwitterHelper TwitterMessage
<b>C</b> CGPoint CGRect CGSize	EGORefreshTableHeaderView Event EventDetailTableViewController EventDetailViewController EventsTableViewController EventTableViewCell	<b>M</b> MapViewController MocTableViewController	<b>S</b> SBJSON SBJsonBase SBJsonParser SBJsonParser SBJsonWriter SBJsonWriter sockaddr_in	<b>U</b> UIApplicationDelegate UIImage(INResizemageAlllocator) UIScrollViewDelegate UITabBarControllerDelegate UIWebViewDelegate UpdatesUIViewController UpdateTableViewController
<b>D</b> DC17TableNavItem dc17TableViewController DefconAppAppDelegate DefconAppDownload DefconAppLoadingView DefconAppUtils DefconImageView DefconInfoTableViewController	<b>E</b> EGORefreshTableHeaderView Event EventDetailTableViewController EventDetailViewController EventsTableViewController EventTableViewCell	<b>N</b> NewTableViewController NSObject(NSObject_SBJSON) NSObject NSString(NSString_SBJSON)	<b>P</b> Person PersonType PullToRefreshTableViewController	<b>T</b> TalkDetailTableViewController
<b>F</b> FavoriteTableViewController FirstViewController	<b>H</b> HashDefconTableViewController			

B I C I D I E I F I H I I I M I N I P I R I S I T I U



The **Files** tab shows the interface files that iNalyzer has generated.



## Summary

In this section, we discussed the static analysis of iOS application using iNalyzer and how it eases our tasks. In the next section we will look at how we can use iNalyzer further for runtime analysis of iOS applications.

## iNalyzer With Unencrypted Applications

Unencrypted applications, such as system applications (e.g. Calendar, SpringBoard, Photos, Music, etc.), must be dealt with differently when using iNalyzer.

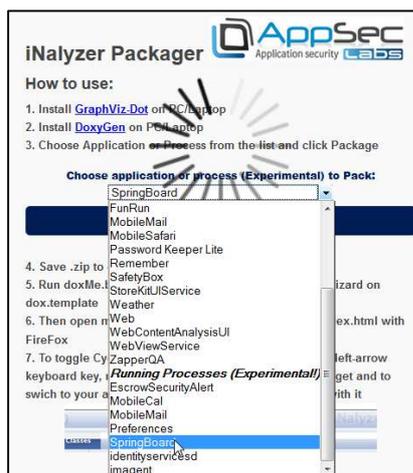
These applications should be managed in iNalyzer with the '-direct' option. In order to find the relevant path for these applications, you can locate them by looking for processes that are running on a system using the 'ps -ef' option:

```
coreDumps-iPhone:/Applications/iNalyzer5.app root# ps -ef | grep -i spring
501 20441 1 0 0:00.00 ?? 0:46.85 /System/Library/CoreServices/SpringBoard.app/SpringBoard
0 20810 490 0 0:00.00 ttys000 0:00.01 grep -i spring
```

1. After we found the desired application (SpringBoard), we copy its folder location:  
*/System/Library/CoreServices/SpringBoard.app*
2. We can employ iNalyzer by using:  
*#!/iNalyzer5 - -direct/System/Library/CoreServices/SpringBoard.app*  
When this completes, the package file (.ipa) will be waiting for you here:  
*/var/root/Documents/iNalyzer*

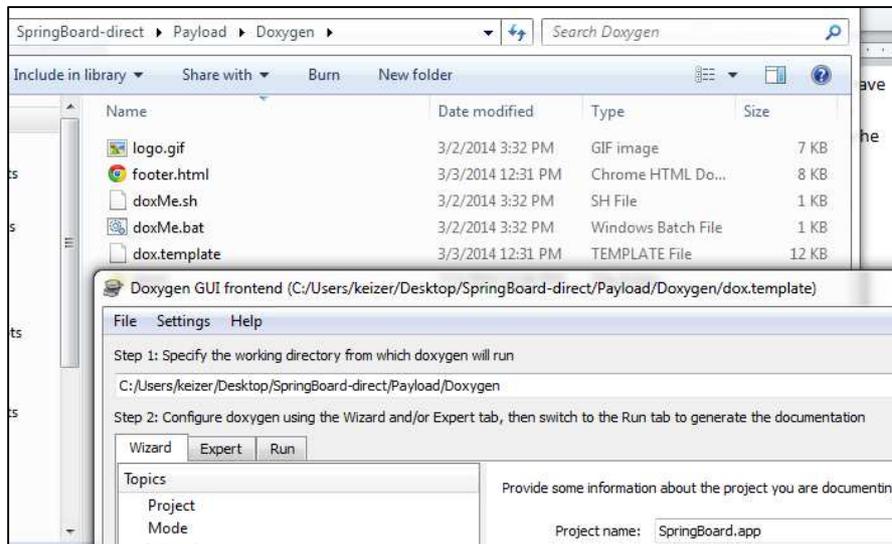
```
coreDumps-iPhone:/Applications/iNalyzer5.app root# ./iNalyzer5 --direct /System/Library/CoreServices/SpringBoard.app/
iNalyzer is iNalyzing SpringBoard.app...
Application is not encrypted
iNalyzer [1/9] Dumping Headers:Done
iNalyzer [2/9] Creating SnapShot into ClientFiles:
iNalyzer [3/9] Dumping SQLite files:Done
iNalyzer [4/9] Dumping Entitlements:Done
iNalyzer [5/9] Dumping plist files:Done
iNalyzer [6/9] Dumping binary cookies:failed
iNalyzer [7/9] Dumping keychain data:Done
iNalyzer [8/9] Dumping File Protection Class:Done
iNalyzer [9/9] Patching Headers:Done
iNalyzer done, file saved at:/var/root/Documents/iNalyzer/SpringBoard-direct.ipa
```

3. Alternately, you can use the iNalyzer interface to download the process' package file from your device. This is done by using the (experimental) section **Running Processes**

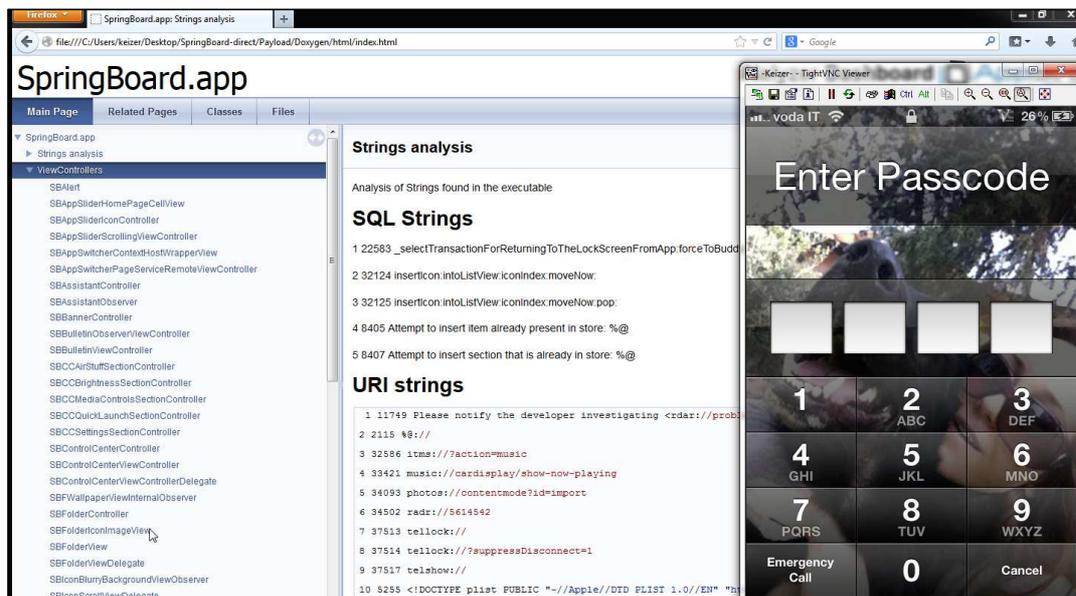


4. Copy the .ipa file to your PC, change the .ipa extension to .zip and unzip it into a folder. Once you have the folder ready, repeat the same procedure explained earlier to run DoxyGen.

- You can use the script mentioned earlier, or DoxyWizard if you run on your PC, by selecting the dox.template files within the Doxygen folder in the app.



- After you have completed step 5, you should have the index.html file, exactly as it would work on an installed and encrypted application.



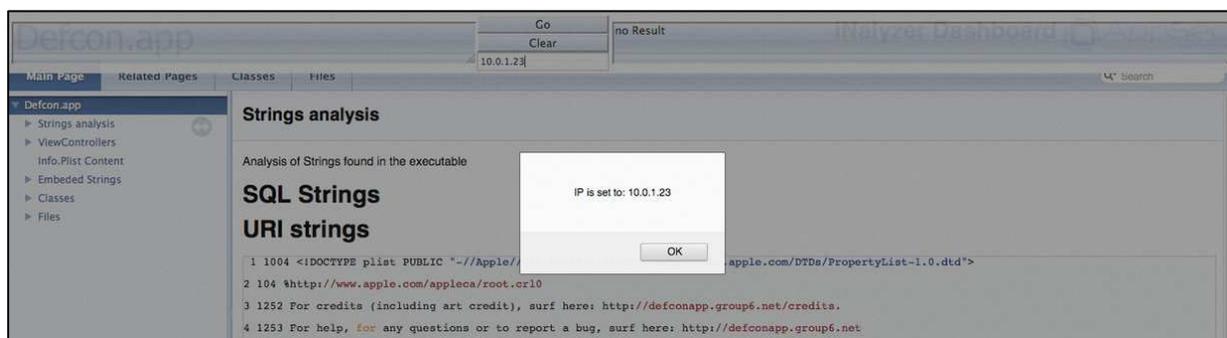
## Runtime Analysis Using iNalyzer

In the previous section, we looked at how we can perform static analysis of iOS Applications using iNalyzer. In this section, we will look at how we can use iNalyzer to perform runtime analysis of iOS applications. We can invoke methods during runtime, find the value of a particular instance variable at a particular time in the app, and essentially do anything that we can do with Cycript.

1. To open the runtime interpreter, first open the `indx.html` file generated by DoxyGen for the app you want to analyze, and then double click the left arrow key.



2. You will see a console come up on the top (as shown below), where we can type commands. Enter your device's IP address in the box in the middle-top and click **Enter**.



3. Make sure the app you want to analyze is open (i.e. in the foreground) on your device and that your device is not in sleep mode. This is important because if your app is in the background or the device is in sleep mode, your app will be temporarily paused by the OS - hence it is not possible to perform any kind of runtime analysis on it.
4. Once the app is open, you can type any command on the left side of the console, just like you would type on (such as: `UIApp`). The response will be shown in the right side of the console:



5. If you want to hide the status bar from the app, you can use the following command:  
`[[UIApplication sharedApplication] setHidden:YES animated:YES];`  
 In this case there will be no response, since the response type of this method is void.

```
[[UIApplication sharedApplication] setHidden:YES animated:YES];
```

Go  
Clear  
10.0.1.23

6. The status bar has been hidden in the app and we no longer see the time on the top.



Similarly, we can find the delegate class of this app:

```
UIApplication.delegate
```

Go  
Clear  
@\*<DefconAppAppDelegate: 0x1a2590>\*<br>  
10.0.1.23

7. We can set the application icon badge number. In this case, let's set it to 9000:

```
[[UIApp setApplicationIconBadgeNumber:9000];
```

Go  
Clear  
10.0.1.23

8. The application icon on your device will now show the icon badge number:



Since this is exactly the same as having a Cypri console, we can enter the Javascript code as well, or any other command from Cypri's documentation.

 The following image is a command taken from the **Cycript Tricks** page:

```

[[] for (i in *UIApp)]
Go
Clear
10.0.1.23
["isa", "_delegate", "_touchMap", "_exclusiveTouchWindows", "_event", "_touchesEvent", "_motionEvent", "_remoteControlEvent", "_remoteControlEventsObservers", "_topLevelNibObjects", "_networkResourcesCurrentlyLoadingCount", "_hideNetworkActivityIndicatorTimer", "_editAlertView", "_statusBar", "_statusBarWindow", "_observerBlocks", "_mainSt
  
```

9. You can create a function using both Objective-C and Javascript syntax. This method can be used whenever you choose. The following images demonstrate two different methods:

```

function printMethods(className) {
  var count = new new Type("I");
  var methods = class_copyMethodList(objc_getClass(className), count);
  var methodsArray = [];
  for(var i = 0; i < *count; i++) {
    var method = methods[i];
    methodsArray.push(selector.method_getName(method),
    implementation.method_getImplementation(method));
  }
  free(methods);
  free(count);
  return methodsArray;
}
  
```

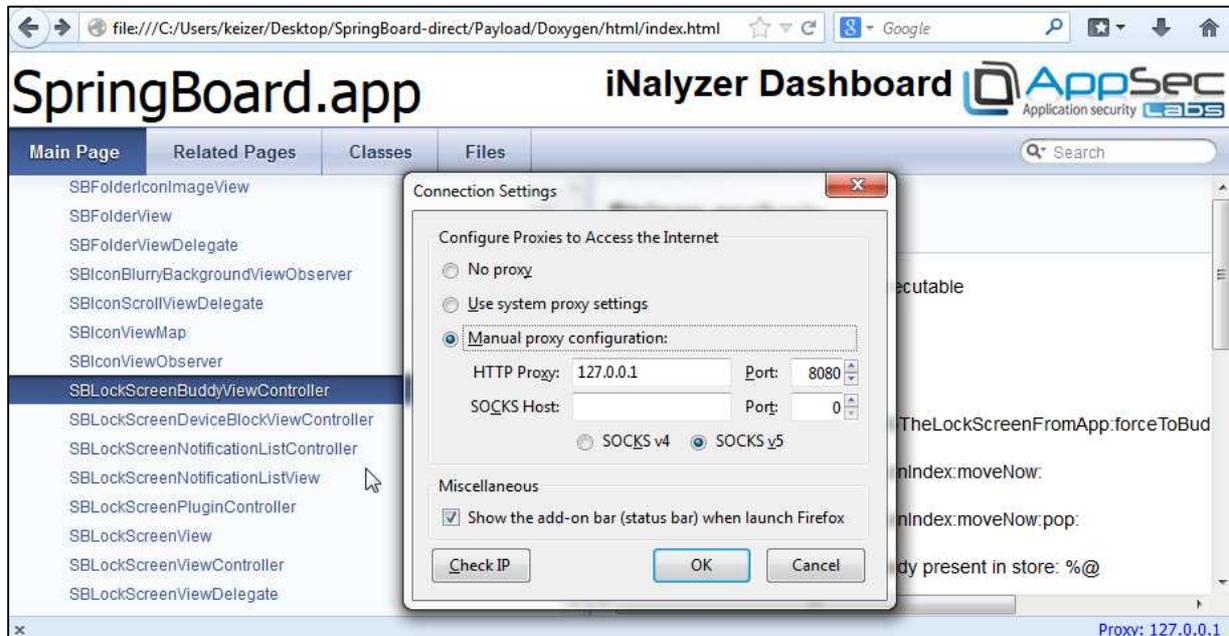
```

printMethods(DefconAppAppDelegate)
Go
Clear
10.0.1.23
[[selector:@selector(setTabBarController).implementation:0x2e95],
[selector:@selector(setSplashView).implementation:0x2b41],
[selector:@selector(applicationDocumentsDirectory).implementation:0x38dd],
[selector:@selector(fetchResourceWithURL).implementation:0x2c0d],
[selector:@selector(initializeDataStore).implementation:0x30b1],
[selector:@selector(initializeUI).implementation:0x339d],
[selector:@selector(splashView).implementation:0x2b35],
[selector:@selector(window).implementation:0x2b59],
[selector:@selector(applicationWillTerminate).implementation:0x3689],
[selector:@selector(applicationDidFinishLaunching).implementation:0x2f6d],
[selector:@selector(setWindow).implementation:0x2e6d],
[selector:@selector(tabBarController).implementation:0x2b4d],
[selector:@selector(connection:didFailWithError).implementation:0x3b75],
[selector:@selector(connection:didReceiveData).implementation:0x380d],
[selector:@selector(connection:didFinishLoading).implementation:0x390d],
[selector:@selector(connection:didReceiveResponse).implementation:0x2b31],
[selector:@selector(activeDownloads).implementation:0x2e1d],
[selector:@selector(setActiveDownloads).implementation:0x2ebd],
[selector:@selector(idealLoc).implementation:0x243d],
[selector:@selector(persistentStoreCoordinator).implementation:0x3239],
[selector:@selector(managedObjectModel).implementation:0x2b65],
[selector:@selector(managedObjectContext).implementation:0x2ba1]<br>
  
```

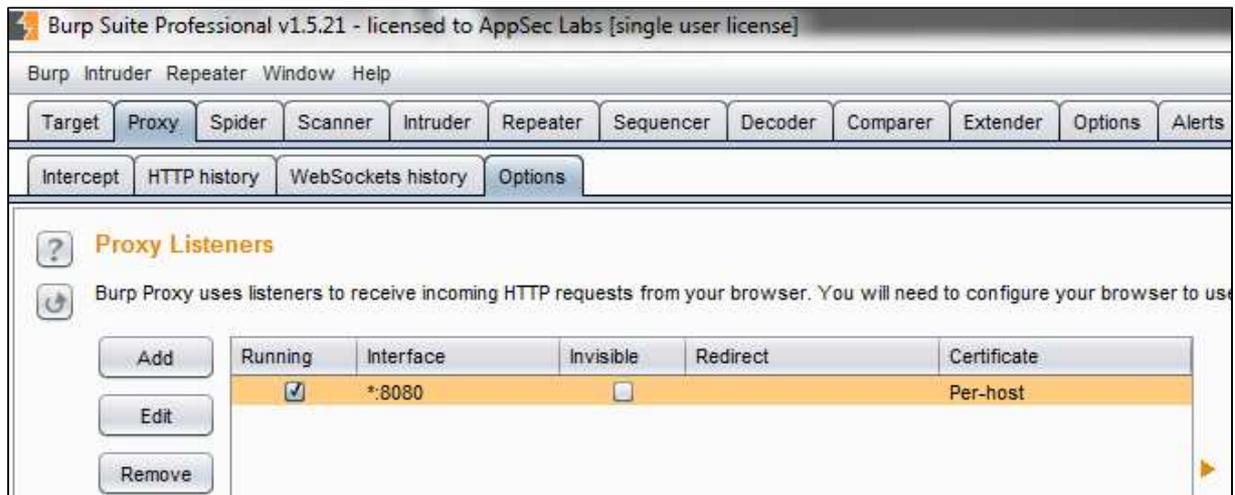
## Running iNalyzer With Burp Proxy

Since iNalyzer uses the browser as its interface, interactive with Burp becomes intuitive. Set your browser to go through Burp by setting the local host and port configured in Burp:

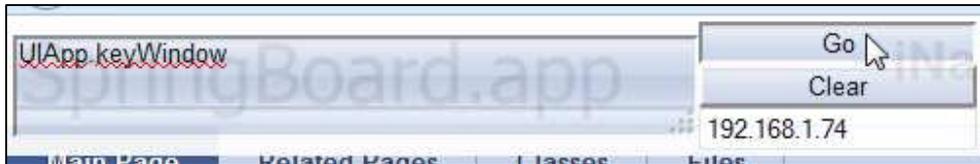
📌 Configuring in browser:



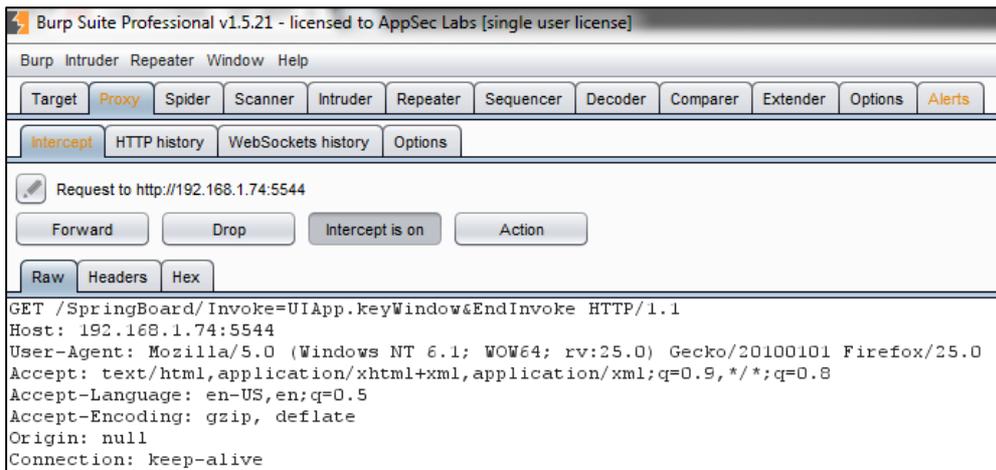
📌 Configuring in Burp:



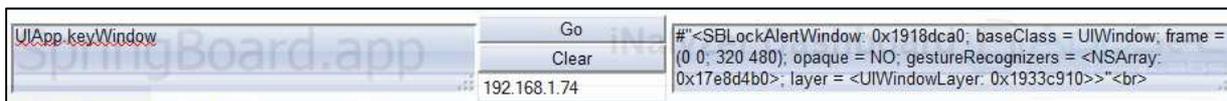
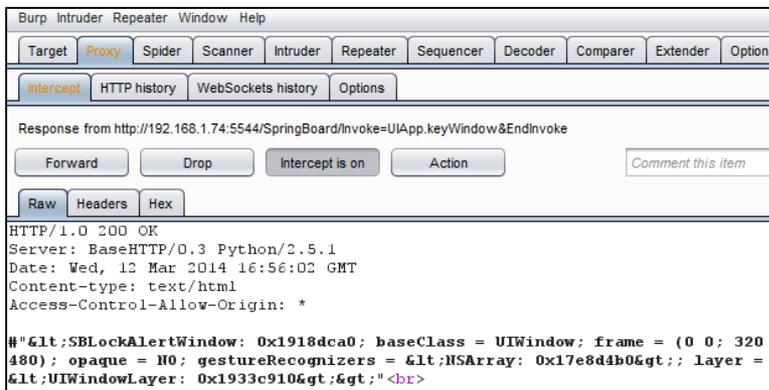
- Any action performed within iNalyzer is sent to your device. This can include, for example, querying the device which is the UI's key window:



- This will send a request to the device, which can be intercepted by Burp:



- The response can be intercepted by Burp, as shown in the following images:



In this way, all features available in Burp can become useful when using the iNalyzer. One known feature of Burp is the brute-force tool, the Intruder, which can come in handy when you want to test your mobile app against brute-force attacks.

We'll use SpringBoard for our demo. We already know that the device is locked and we need to provide a 4 digit passcode in order to open the device desktop. But what if we could brute-force our way into it? Integrating iNalyzer with Burp can help us achieve this task.

First, we need to find the method that sends the passcode into the device.

1. Let's look at the **Class List**, and search for a class that can manage and work with passcodes:



Here we have found the **SBPasscodeController**. When we review the method list, we can see that it has a method that is called: **\_passwordEntered**. This may be exactly what we're looking for.

2. Let's set "pass" as an instance of this class, we can use Cycript's chosen method:

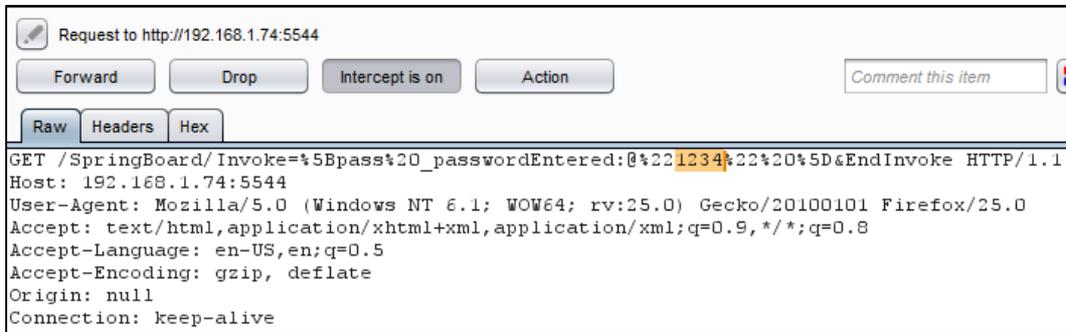


3. Now we can call the method **\_passwordEntered** with the 4 digit parameter:



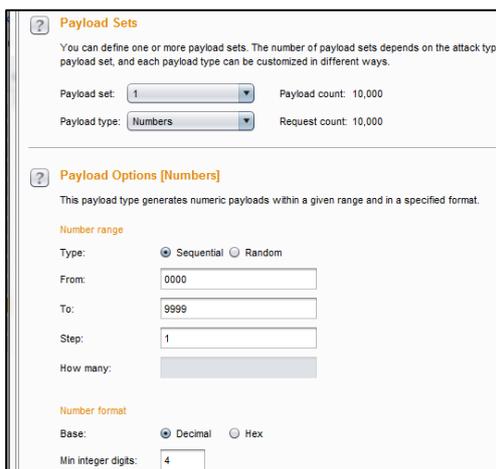
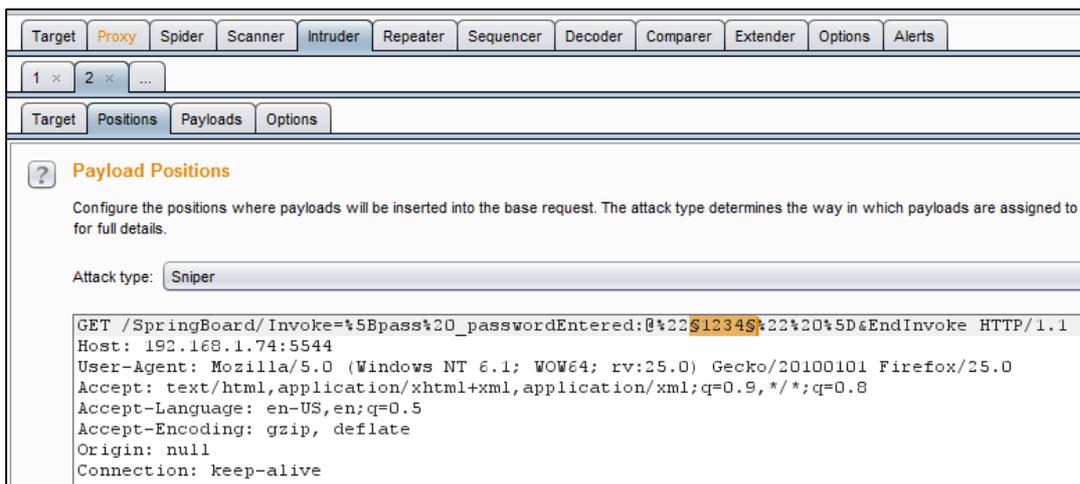
An empty response will be returned.

4. Since we do not know the correct passcode, let's brute-force it. First, catch the request in Burp:

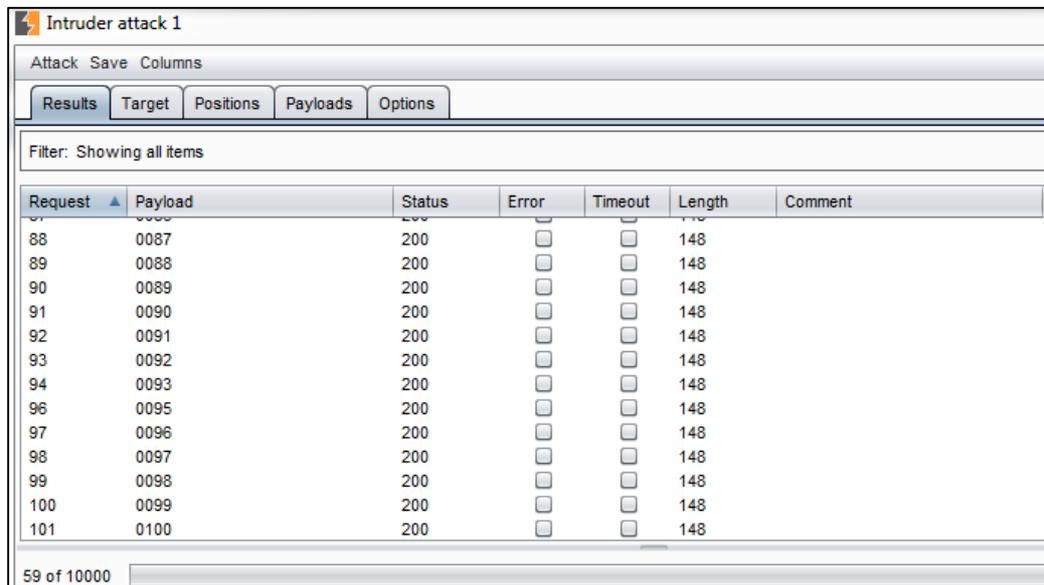


Notice the passcode parameter in the GET request.

5. Now, let's use the **Intruder** as you would use it with a regular web-application:



6. The Intruder will generate the attack as a normal web-application:



The screenshot shows a web application interface for an intruder attack. The title is "Intruder attack 1". Below the title, there are tabs for "Results", "Target", "Positions", "Payloads", and "Options", with "Results" selected. A filter bar indicates "Showing all items". The main content is a table with the following columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The table contains 13 rows of data, all with a status of 200 and a length of 148. The bottom of the interface shows "59 of 10000" items.

Request	Payload	Status	Error	Timeout	Length	Comment
88	0087	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
89	0088	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
90	0089	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
91	0090	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
92	0091	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
93	0092	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
94	0093	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
96	0095	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
97	0096	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
98	0097	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
99	0098	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
100	0099	200	<input type="checkbox"/>	<input type="checkbox"/>	148	
101	0100	200	<input type="checkbox"/>	<input type="checkbox"/>	148	

## Note

Performing these tests on a Mac OS X Mountain Lion 10.8.4, may create a problem with the latest version of GraphViz and it may cause it to hang each time. We recommend downloading an older version of GraphViz (v 2.30.1). You can find older versions of GraphViz for Mac OS [here](#).

## Useful Links

- [AppSec Labs Home page](#)
- [iNalyzer Official page](#)
- [iNalyzer Github page](#)
- [Graphviz Download page](#)
- [Doxygen Download page](#)
- [Attacking iPhone Applications, OWASP Israel 2012](#)
- [iNalyzer Vs. iSafePlay Video](#)
- [No More iOS Blackbox Assessments Presentation, HITB Amsterdam 2013](#)
- [No More iOS Blackbox Assessments Video, HITB Amsterdam 2013](#)
- [Game of Pwns: Advanced iPhone pen-testing, OWASP Israel 2013](#)
- [Cycrypt Tricks page](#)

We hope this user guide has been informative and useful! If you have any further questions please get in touch with us at: [iNalyzer@AppSec-Labs.Com](mailto:iNalyzer@AppSec-Labs.Com)