escrypt
SECURITY. TRUST. SUCCESS.

radare2 and Frida

in the OWASP Mobile Security Testing Guide

Carlos Holguera
r2con, 06.09.2019

escrypt
SECURITY. TRUST. SUCCESS.

Carlos Holguera

@grepharder

- Security Engineer working at ESCRYPT GmbH since 2012
- Areas of expertise:
  - Mobile & Automotive Security Testing
  - Security Testing Automation
- Project Leader of the OWASP Mobile Security Testing Guide w/ Sven Schleier and Jeroen Willemsen

escrypt
SECURITY. TRUST. SUCCESS.

1 From the Standard to the Guide

2 Examples

escrypt
SECURITY. TRUST. SUCCESS.

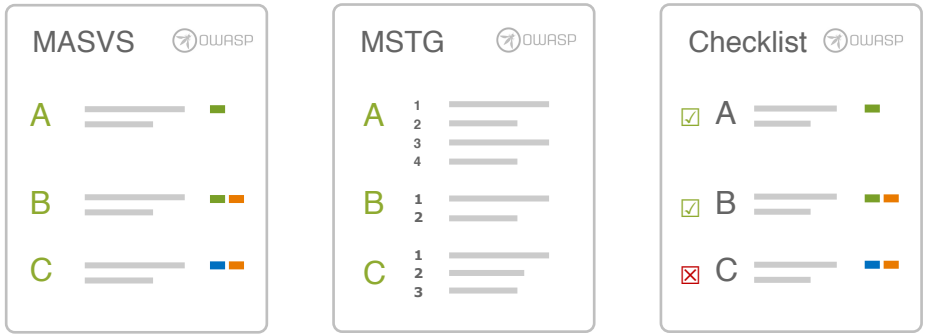
1 From the Standard to the Guide

2 Examples

escrypt
SECURITY. TRUST. SUCCESS.

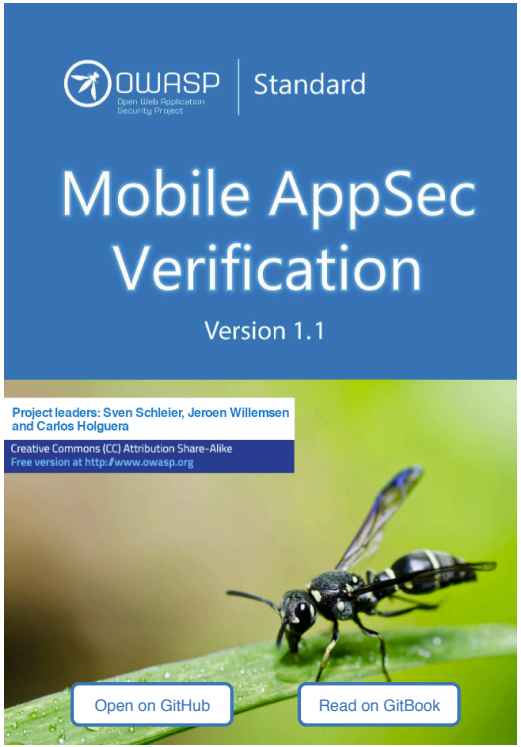
- Trustworthy sources?
- Right Methodology?
- Latest Techniques?

Online videos, articles, trainings ??



escrypt
SECURITY. TRUST. SUCCESS.

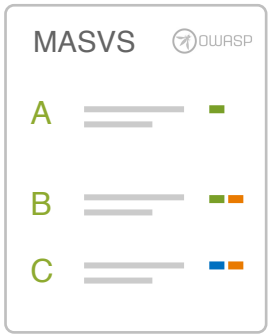
EPUB
MOBI
PDF
DOCX



The MASVS is a standard that establishes the security requirements for mobile app security.

OS agnostic

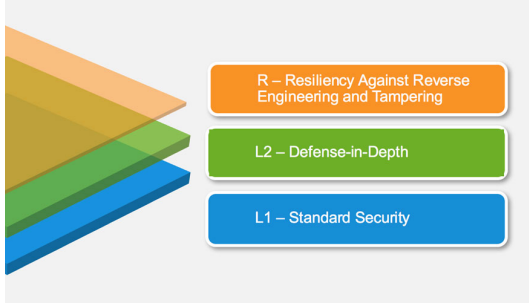
Chinese
English
German
Japanese
Russian
Spanish



escrypt
SECURITY. TRUST. SUCCESS.

Security Verification Requirements

#	MSTG-ID	Description	L1	L2
5.1	MSTG-NETWORK-1	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	✓	✓
5.2	MSTG-NETWORK-2	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	✓	✓
5.3	MSTG-NETWORK-3	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.	✓	✓
5.4	MSTG-NETWORK-4	The app either uses its own certificate store, or pins the endpoint certificate or public key, and subsequently does not establish connections with endpoints that offer a different certificate or key, even if signed by a trusted CA.		✓
5.5	MSTG-NETWORK-5	The app doesn't rely on a single insecure communication channel (email or SMS) for critical operations, such as enrollments and account recovery.		✓
5.6	MSTG-NETWORK-6	The app only depends on up-to-date connectivity and security libraries.		✓

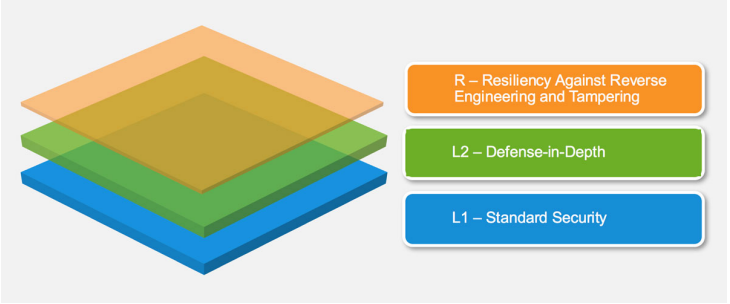


escrypt
SECURITY. TRUST. SUCCESS.

MASVS-L1: all mobile apps.

MASVS-L2: apps handling **sensitive data** and/or functionality.

MASVS-R: apps handling **highly sensitive data** and may serve as a means of **protecting intellectual property** or **tamper-proofing** an app.



- MASVS L1 Alarm App
- MASVS L2 Health App
- MASVS L1+R Game App
- MASVS L2+R Banking App

escrypt
SECURITY. TRUST. SUCCESS.

PDF
ePub
Gitbook
DOCX

The MSTG is a comprehensive manual for mobile app security testing and reverse engineering.

It describes technical processes for verifying the controls listed in the MASVS.

escrypt
SECURITY. TRUST. SUCCESS.

Example: iOS Testing Guide

- Platform Overview
- iOS Basic Security Testing
- Data Storage on iOS
- iOS Cryptographic APIs
- Local Authentication on iOS
- iOS Network APIs
- iOS Platform APIs
- Code Quality and Build Settings for iOS Apps
- Tampering and Reverse Engineering on iOS
- iOS Anti-Reversing Defenses

From the Standard to the Guide

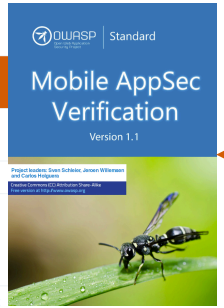
OWASP Mobile Security Testing Guide: MSTG IDs

escrypt
SECURITY. TRUST. SUCCESS.

Security Verification Requirements

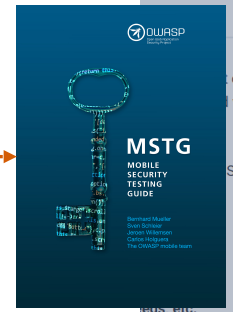
#	MSTG-ID	Description	L1	L2
6.1	MSTG-PLATFORM-1	The app only requests the minimum set of permissions necessary.	✓	✓
6.2	MSTG-PLATFORM-2	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.		
6.3	MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.		
6.4	MSTG-PLATFORM-4	The app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected.		
6.5	MSTG-PLATFORM-5	JavaScript is disabled in WebViews unless explicitly required.		
6.6	MSTG-PLATFORM-6	WebViews are configured to allow only the minimum set of protocol handlers required (ideally, only https is supported). Potentially dangerous handlers, such as file, tel and app-id, are disabled.		
6.7	MSTG-PLATFORM-7	If native methods of the app are exposed to a WebView, verify that the WebView only renders JavaScript contained within the app package.	✓	✓
6.8	MSTG-PLATFORM-8	Object deserialization, if any, is implemented using safe serialization APIs.	✓	✓
6.9	MSTG-PLATFORM-9	The app protects itself against screen overlay attacks. (Android only)		✓

GitBook



MSTG-PLATFORM-3

GitBook



MSTG-PLATFORM-3

iOS Platform APIs

Testing Custom URL Schemes (MSTG-PLATFORM-3)

Overview Custom URL schemes allow apps to communicate via a custom protocol . An app must declare support for...

References

...app only requests the minimum set of permissions necessary," MSTG-PLATFORM-3: "The app does not export sensitive functionality via custom URL...

Android Platform APIs

Testing Custom URL Schemes (MSTG-PLATFORM-3)

Overview Both Android and iOS allow inter-app communication via custom URL schemes. These custom URLs allow other applications to perform...

References

...mechanisms such as intents, custom URLs, and network sources," MSTG-PLATFORM-3: "The app does not export sensitive functionality via custom URL...

escrypt
SECURITY. TRUST. SUCCESS.

Mobile Application Security Requirements - Android

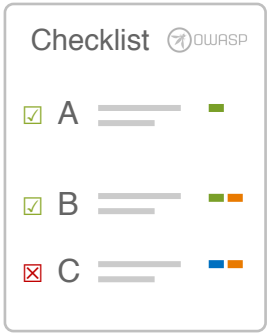
ID	MSTG-ID	Detailed Verification Requirement	Level 1	Level 2	Status	
V1		**Architecture, design and threat modelling**				
1.1	MSTG-ARCH-1	All app components are identified and known to be needed.	✓	✓		Architectural Information
1.2	MSTG-ARCH-2	Security controls are never enforced only on the client side, but on the respective remote endpoints.	✓	✓		Injection Flaws (MSTG-ARCH-2 and MSTG-PLATFORM-2)
1.3	MSTG-ARCH-3	A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.	✓	✓		Architectural Information
1.4	MSTG-ARCH-4	Data considered sensitive in the context of the mobile app is clearly identified.	✓	✓		Identifying Sensitive Data
1.5	MSTG-ARCH-5	All app components are defined in terms of the business functions and/or security functions they provide.		✓	N/A	Environmental Information
1.6	MSTG-ARCH-6	A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.		✓	N/A	Mapping the Application
1.7	MSTG-ARCH-7	All security controls have a centralized implementation.		✓	N/A	Testing for insecure Configuration of Instant Apps (MSTG-AR
1.8	MSTG-ARCH-8	There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.		✓	N/A	Cryptographic policy
1.9	MSTG-ARCH-9	A mechanism for enforcing updates of the mobile app exists.		✓	N/A	Testing enforced updating (MSTG-ARCH-9)
1.10	MSTG-ARCH-10	Security is addressed within all parts of the software development lifecycle.		✓	N/A	Security Testing and the SDLC
V2		**Data Storage and Privacy**				
2.1	MSTG-STORAGE-1	System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys.	✓	✓		Testing Local Storage for Sensitive Data (MSTG-STORAGE-1
2.2	MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.				Testing Local Storage for Sensitive Data (MSTG-STORAGE-1
2.3	MSTG-STORAGE-3	No sensitive data is written to application logs.	✓	✓		Testing Logs for Sensitive Data (MSTG-STORAGE-3)
2.4	MSTG-STORAGE-4	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	✓	✓		Determining Whether Sensitive Data is Sent to Third Parties
2.5	MSTG-STORAGE-5	The keyboard cache is disabled on text inputs that process sensitive data.	✓	✓		Determining Whether the Keyboard Cache Is Disabled for Te
2.6	MSTG-STORAGE-6	No sensitive data is exposed via IPC mechanisms.	✓	✓		Determining Whether Sensitive Stored Data Has Been Expos
2.7	MSTG-STORAGE-7	No sensitive data, such as passwords or pins, is exposed through the user interface.	✓	✓		Checking for Sensitive Data Disclosure Through the User Inte
2.8	MSTG-STORAGE-8	No sensitive data is included in backups generated by the mobile operating system.		✓	N/A	Testing Backups for Sensitive Data (MSTG-STORAGE-8)
2.9	MSTG-STORAGE-9	The app removes sensitive data from views when moved to the background.		✓	N/A	Finding Sensitive Information in Auto-Generated Screenshot
2.10	MSTG-STORAGE-10	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.		✓	N/A	Checking Memory for Sensitive Data (MSTG-STORAGE-10)
2.11	MSTG-STORAGE-11	The app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode.		✓	N/A	Testing the Device-Access-Security Policy (MSTG-STORAGE-
2.12	MSTG-STORAGE-12	The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.		✓	N/A	Testing User Education (MSTG-STORAGE-12)
V3		**Cryptography**				

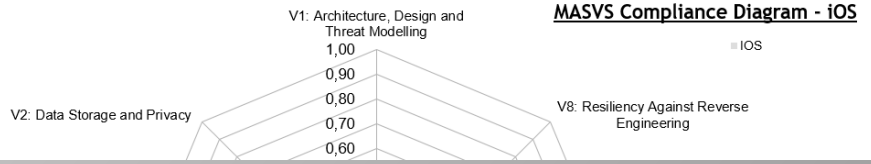
Get from GitHub

Security Requirements - Android | Anti-RE - Android | Security Requirements - iOS | Anti-RE - iOS | Version history

The Checklist is used in security assessments. It contains links to the MSTG test cases for each requirement.

English
French
Japanese
Spanish

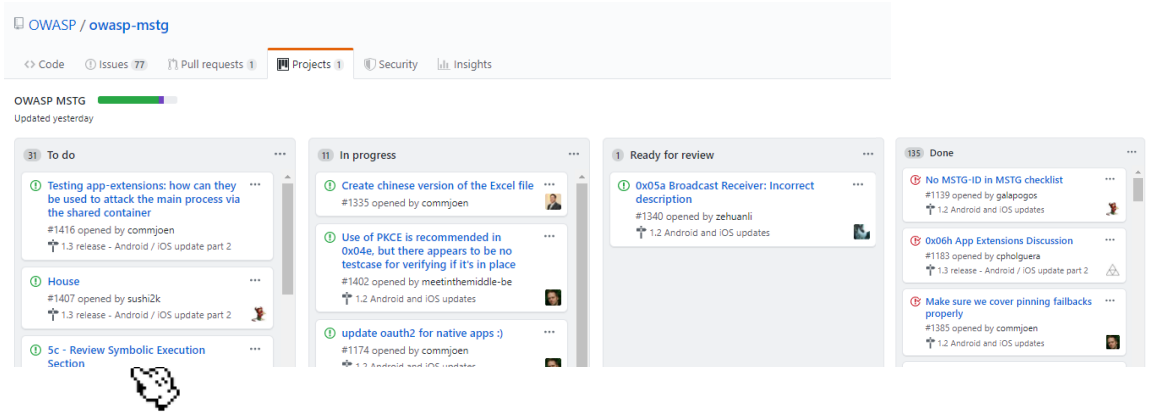




From the Standard to the Guide

Contributing: pick, PR, merge, repeat

escrypt
SECURITY. TRUST. SUCCESS.



Contact us in GitHub:

Sven sushi2k
Singapore

Jeroen Willemsen commjoen

cpholguera cpholguera

Or Slack:

#project-mobile_omtg

Follow our Style Guide

Follow our Code of Conduct

Pick Issue

We'll assign it to you

Send PR

When you're ready

Get Feedback

Time for polishing the PR

Get Approval

And we'll merge it

escrypt
SECURITY. TRUST. SUCCESS.

1 From the Standard to the Guide

2 Examples

Example A
MSTG-PLATFORM-3

escrypt
SECURITY. TRUST. SUCCESS.

#	MSTG-ID	Description	L1	L2
6.3	MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.	✓	✓

>> demo

Testing Custom URL Schemes (MSTG-PLATFORM-3)

- Performing URL Requests

Using Frida

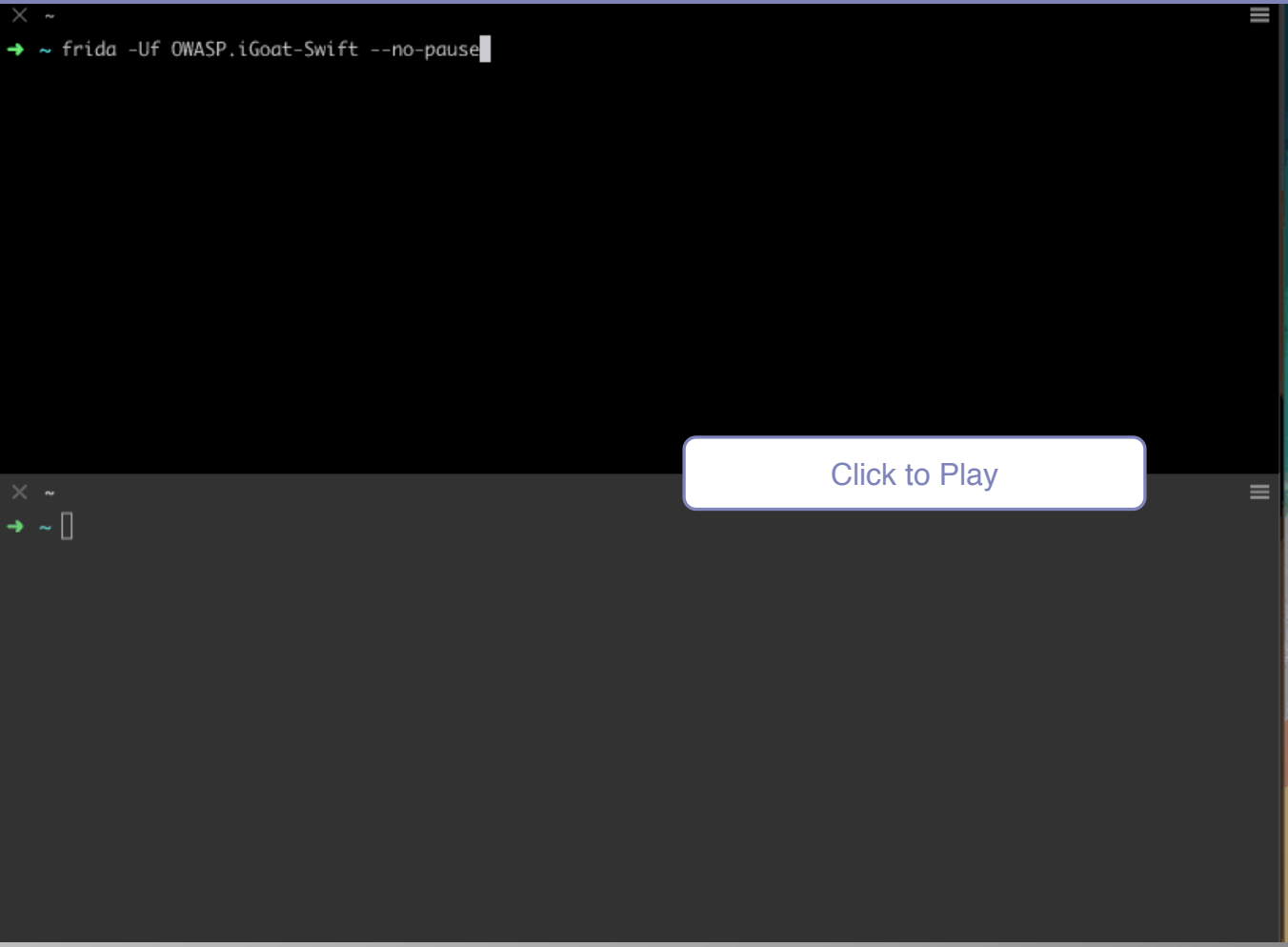
If you simply want to open the URL scheme you can do it using Frida:

```
$ frida -U iGoat-Swift

[iPhone::iGoat-Swift]-> function openURL(url) {
                            var UIApplication = ObjC.classes.UIApplication.sharedApplication();
                            var toOpen = ObjC.classes.NSURL.URLWithString_(url);
                            return UIApplication.openURL_(toOpen);
                        }
[iPhone::iGoat-Swift]-> openURL("tel://234234234")
true
```

FRIDA
CLI

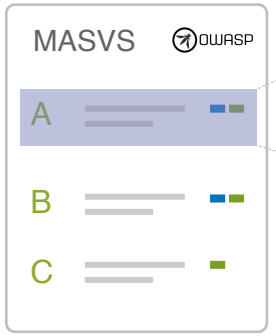
Example A (MSTG-PLATFORM-3): openURL with Frida

escrypt



Example A
MSTG-PLATFORM-3

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-PLATFORM-3

#	MSTG-ID	Description	L1	L2
6.3	MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.	✓	✓

>> demo

Testing Custom URL Schemes (MSTG-PLATFORM-3)

- Identifying and Hooking the URL Handler Method

Crafting the Link Yourself and Letting Safari Open It

For this we will use the ObjC method observer from Frida CodeShare, which is an extremely handy script that allows you to quickly observe any collection of methods or classes just by providing a simple pattern.

In this case we are interested into all methods containing "openURL", therefore our pattern will be `*[* *openURL*]`:

- The first asterisk will match all instance `-` and class `+` methods.
- The second matches all Objective-C classes.
- The third and forth allow to match any method containing the string `openURL`.

```
$ frida -U iGoat-Swift --codeshare mrmacete/objc-method-observer

[iPhone::iGoat-Swift]-> observeSomething("*[* *openURL*]");
Observing  -[_UIDICActivityItemProvider activityViewController:openURLAnnotationForActivityType:]
Observing  -[CNQuickActionsManager _openURL:]
Observing  -[SUClientController openURL:]
Observing  -[SUClientController openURL:inClientWithIdentifier:]
Observing  -[FBSSystemService openURL:application:options:clientPort:withResult:]
Observing  -[iGoat_Swift.AppDelegate application:openURL:options:]
```

FRIDA CLI

FRIDA CodeShare

Example A (MSTG-PLATFORM-3): Trace URL handler

escrypt

frida

```
[iPhone::OWASP.iGoat-Swift]-> function openURL(url) {
                              var UIApplication = ObjC.classes.UIApplication.sharedApplication();
                              var toOpen = ObjC.classes.NSURL.URLWithString_(url);
                              return UIApplication.openURL_(toOpen);
                          }
undefined
[iPhone::OWASP.iGoat-Swift]-> openURL("tel://12345")
true
[iPhone::OWASP.iGoat-Swift]->
```

Click to Play

~

```
➜  ~ frida -U iGoat-Swift --codeshare mrmacete/objc-method-observer
```

About iGoat

OWASP iGoat (Swift) v1.0

Thank you for using iGoat-Swift. This is a Swift version of original iGoat (Objective C) project. This program is a demonstration of common iOS application security weaknesses and their remediations. The exercises are intended to provide hands-on experience for iOS developers and mobile pentesters.

OWASP iGoat project has been presented at **AppSec USA, SEC-T, BruCON, c0c0n**

Project Code: Github

Project Lead:
Swaroop Yermalkar Follow @swaroopsy

Credits:
Anthony Gonsalves
Ken van Wyk
Jonathan Carter

To contribute to iGoat, please contact Swaroop

Example A (MSTG-PLATFORM-3): openURL from Safari

escrypt

frida

```
                        var toOpen = ObjC.classes.NSURL.URLWithString_(url);
                        return UIApplication.openURL_(toOpen);
                    }
undefined
[iPhone::OWASP.iGoat-Swift]-> openURL("tel://12345")
true
[iPhone::OWASP.iGoat-Swift]-> openURL("igoat://")
true
[iPhone::OWASP.iGoat-Swift]->
```

Click to Play

frida

```
(0x1c403ca80)  -[iGoat_Swift.AppDelegate application:openURL:options:]
application: <UIApplication: 0x102815c00> (UIApplication)
openURL: igoat:// (NSURL)
options: {
    UIApplicationOpenURLOptionsOpenInPlaceKey = 0;
    UIApplicationOpenURLOptionsSourceApplicationKey = "OWASP.iGoat-Swift";
} (__NSDictionaryM)
0x18ec9b0d8 UIKit!__58-[UIApplication _applicationOpenURLAction:payload:origin:]_block_invoke
0x18ec9aa94 UIKit!-[UIApplication _applicationOpenURLAction:payload:origin:]
0x18eca41fc UIKit!-[UIApplication _handleNonLaunchSpecificActions:forScene:withTransitionContext:completion:]
0x18f6a6480 UIKit!__82-[_UIApplicationCanvas _transitionLifecycleStateWithTransitionContext:completion:]_block_invo
ke
0x18f6a635c UIKit!-[_UIApplicationCanvas _transitionLifecycleStateWithTransitionContext:completion:]
0x18f418294 UIKit!__125-[_UICanvasLifecycleSettingsDiffAction performActionsForCanvas:withUpdatedScene:settingsDiff
:fromSettings:transitionContext:]_block_invoke
0x18f5af0ac UIKit!_performActionsWithDelayForTransitionContext
```

About iGoat

OWASP iGoat (Swift) v1.0

Thank you for using iGoat-Swift. This is a Swift version of original iGoat (Objective C) project. This program is a demonstration of common iOS application security weaknesses and their remediations. The exercises are intended to provide hands-on experience for iOS developers and mobile pentesters.

OWASP iGoat project has been presented at **AppSec USA, SEC-T, BruCON, c0c0n**

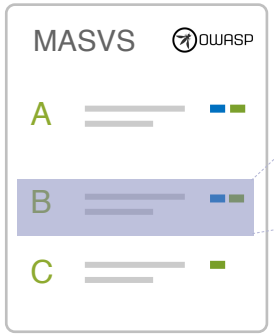
Project Code: Github

Project Lead:
Swaroop Yermalkar Follow @swaroopsy

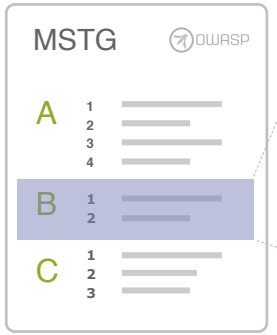
Credits:
Anthony Gonsalves
Ken van Wyk
Jonathan Carter

To contribute to iGoat, please contact Swaroop

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-PLATFORM-5



#	MSTG-ID	Description	L1	L2
6.5	MSTG-PLATFORM-5	JavaScript is disabled in WebViews unless explicitly required.	✓	✓

Testing iOS WebViews (MSTG-PLATFORM-5)

- Identifying WebView Usage

Look out for usages of the above mentioned WebView classes by searching in Xcode.

In the compiled binary you can search in its symbols or strings like this:

UIWebView

```
$ rabin2 -zz ./WheresMyBrowser | egrep "UIWebView$"
489 0x0002fee9 0x10002fee9   9  10 (5.__TEXT.__cstring) ascii UIWebView
896 0x0003c813 0x0003c813  24  25 () ascii _OBJC_CLASS_$_UIWebView
1754 0x00059599 0x00059599  23  24 () ascii _OBJC_CLASS_$_UIWebView
```

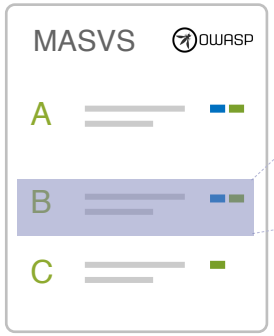
WKWebView

```
$ rabin2 -zz ./WheresMyBrowser | egrep "WKWebView$"
490 0x0002fef3 0x10002fef3   9  10 (5.__TEXT.__cstring) ascii WKWebView
625 0x00031670 0x100031670  17  18 (5.__TEXT.__cstring) ascii unwindToWKWebView
904 0x0003c960 0x0003c960  24  25 () ascii _OBJC_CLASS_$_WKWebView
1757 0x000595e4 0x000595e4  23  24 () ascii _OBJC_CLASS_$_WKWebView
```

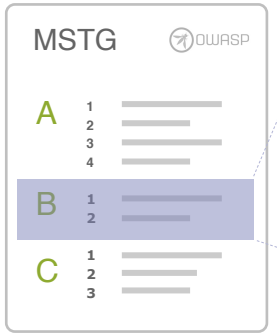
rabin2

Example B
MSTG-PLATFORM-5

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-PLATFORM-5



#	MSTG-ID	Description	L1	L2
6.5	MSTG-PLATFORM-5	JavaScript is disabled in WebViews unless explicitly required.	✓	✓

>> demo

Testing iOS WebViews (MSTG-PLATFORM-5)

- Enumerating WebView Instances

Once you've identified a WebView in the app, you may inspect the heap in order to find instances of one or several of the WebViews that we have seen above.

For example, if you use Frida you can do so by inspecting the heap via "ObjC.choose()"

```
ObjC.choose(ObjC.classes['UIWebView'], {
  onMatch: function (ui) {
    console.log('onMatch: ', ui);
    console.log('URL: ', ui.request().toString());
  },
  onComplete: function () {
    console.log('done for UIWebView!');
  }
});

ObjC.choose(ObjC.classes['WKWebView'], {
  onMatch: function (wk) {
    console.log('onMatch: ', wk);
    console.log('URL: ', wk.URL().toString());
  },
```

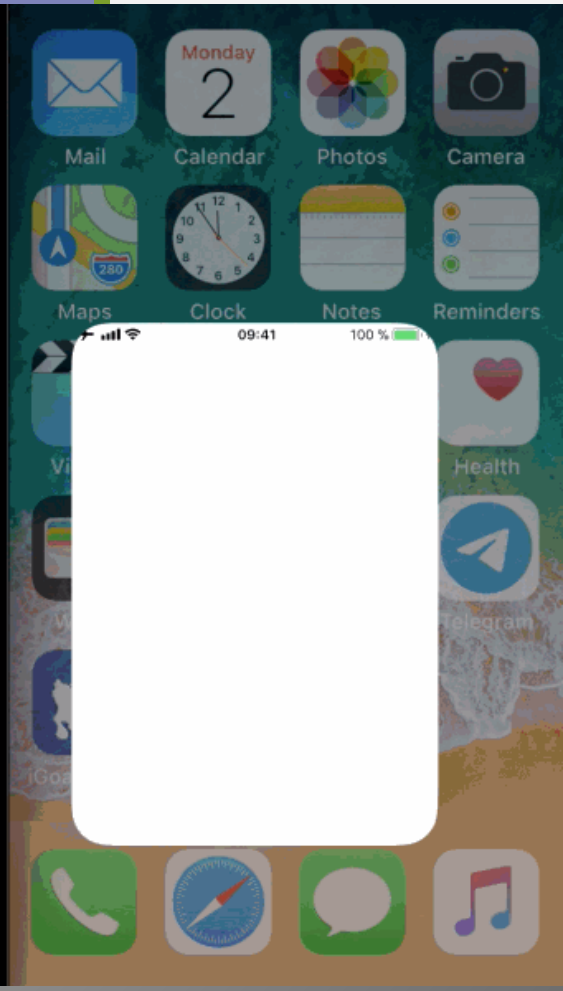
FRIDA
CLI

Example B (MSTG-PLATFORM-5): Enum. WebViews

escrypt

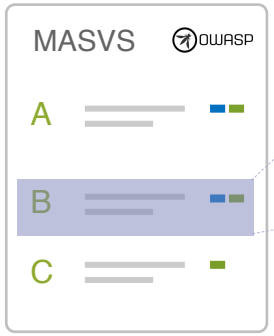
```
➜ WheresMyBrowser.app frida -Uf com.authenticationfailure.WheresMyBrowser

     ____
    / _  |   Frida 12.6.16 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Failed to spawn: the connection is closed
➜ WheresMyBrowser.app frida -Uf com.authenticationfailure.WheresMyBrowser

     ____
    / _  |   Frida 12.6.16 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Spawning `com.authenticationfailure.WheresMyBrowser`...
```

Click to Play

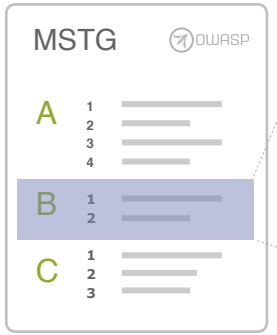


Example B
MSTG-PLATFORM-5

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-PLATFORM-5



#	MSTG-ID	Description	L1	L2
6.5	MSTG-PLATFORM-5	JavaScript is disabled in WebViews unless explicitly required.	✓	✓

Testing iOS WebViews (MSTG-PLATFORM-5)

- Checking if JavaScript is Enabled

Remember that if a `UIWebView` is being used, JavaScript is enabled by default and there's no possibility to disable it.

For `WKWebView`, you should verify if JavaScript is enabled. Use `javaScriptEnabled` from `WKPreferences` for this.

Extend the previous script with the following line:

```
ObjC.choose(ObjC.classes['WKWebView'], {
  onMatch: function (wk) {
    console.log('onMatch: ', wk);
    console.log('javaScriptEnabled:', wk.configuration().preferences().javaScriptEnabled());
//...
  }
});
```

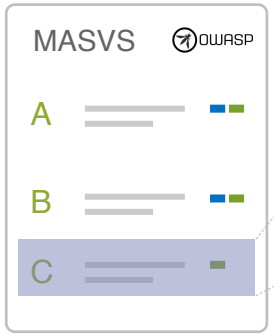
The output shows now that, in fact, JavaScript is enabled:

```
$ frida -U com.authenticationfailure.WheresMyBrowser -l webviews_inspector.js

onMatch:  <WKWebView: 0x1508b1200; frame = (0 0; 320 393); layer = <CALayer: 0x1c4238f20>>

javaScriptEnabled:  true
```

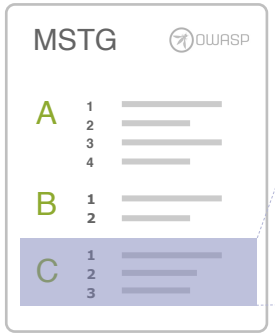
FRIDA CLI

Example C
MSTG-STORAGE-10

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-STORAGE-10



#	MSTG-ID	Description	L1	L2
2.10	MSTG-STORAGE-10	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.	✓	

Testing Memory for Sensitive Data (MSTG-STORAGE-10)

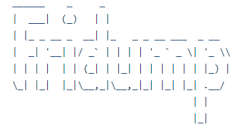
- Retrieving and Analyzing a Memory Dump (c.f. 0x6c Memory Dump)

```
$ python3 fridump.py -u Gadget -s

Current Directory: /Users/foo/PentestTools/iOS/fridump
Output directory is set to: /Users/foo/PentestTools/iOS/fridump/dump
Creating directory...
Starting Memory dump...
Progress: [##################################################] 100.0% Complete

Running strings on all files:
Progress: [##################################################] 100.0% Complete

Finished! Press Ctrl+C
```

When you add the `-s` flag, all strings are extracted from the dumped raw memory files and added to the file `strings.txt`, which is stored in Fridump's dump directory.

In both cases, if you open the file in radare2 you can use its search command (`/`). Note that first we do a standard string search which doesn't succeed and next we search for a wide string, which successfully finds our string "owasp-mstg".

```
$ r2 memory_ios
[0x00000000]> / owasp-mstg
Searching 10 bytes in [0x0-0x628c000]
hits: 0
[0x00000000]> /w owasp-mstg
Searching 20 bytes in [0x0-0x628c000]
hits: 1
0x0036f800 hit4_0 6f007700610073007000 2d006d0073004006700
```



escrypt
SECURITY. TRUST. SUCCESS.

Testing Memory for Sensitive Data (MSTG-STORAGE-10)

- Retrieving and Analyzing a Memory Dump (c.f. 0x6c Memory Dump)

```
$ python3 fridump.py -u Gadget -s

Current Directory: /Users/foo/PentestTools/iOS/fridump
Output directory is set to: /Users/foo/PentestTools/iOS/fridump/dump
Creating directory...
Starting Memory dump...
Progress: [##################################################] 100.0% Complete

Running strings on all files:
Progress: [##################################################] 100.0% Complete

Finished! Press Ctrl+C
```

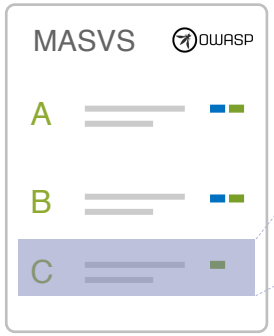
When you add the `-s` flag, all strings are extracted from the dumped raw memory files and added to the file `strings.txt`, which is stored in Fridump's dump directory.

In both cases, if you open the file in radare2 you can use its search command (`/`). Note that first we do a standard string search which doesn't succeed and next we search for a wide string, which successfully finds our string "owasp-mstg".

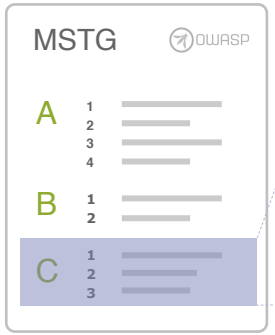
```
$ r2 memory_ios
[0x00000000]> / owasp-mstg
Searching 10 bytes in [0x0-0x628c000]
hits: 0
[0x00000000]> /w owasp-mstg
Searching 20 bytes in [0x0-0x628c000]
hits: 1
0x0036f800 hit4_0 6f007700610073007000 2d006d00730074006700
```

Example C
MSTG-STORAGE-10

escrypt
SECURITY. TRUST. SUCCESS.



MSTG-STORAGE-10



#	MSTG-ID	Description	L1	L2
2.10	MSTG-STORAGE-10	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.	✓	

>> demo

Testing Memory for Sensitive Data (MSTG-STORAGE-10)

- Runtime Memory Analysis (c.f. 0x6c In-Memory Search)

  Now, for this second example, you can search for something that's not in the app binary nor in any loaded library, typically user input. Open the iGoat-Swift app and navigate in the menu to Authentication -> Remote Authentication -> Start. There you'll find a password field that you can overwrite. Write the string "owasp-mstg" but do not click on Login just yet. Perform the following two steps.

```
[0x00000000]> \/ owasp-mstg
hits: 1
0x1c06619c0 hit3_0 owasp-mstg
```

  In fact, the string could be found at address `0x1c06619c0`. Seek `s` to there and retrieve the current memory region with `\dm.`.

```
[0x100d7d332]> s 0x1c06619c0
[0x1c06619c0]> \dm.
0x00000001c0000000 - 0x00000001c8000000 rw-
```

  Now you know that the string is located in a rw- (read and write) region of the memory map.

r2frida

Example C (MSTG-STORAGE-10): Search before login

escrypt

```
➜  ~ r2 frida://usb//iGoat-Swift
```

Click to Play

Example C (MSTG-STORAGE-10): Search after login

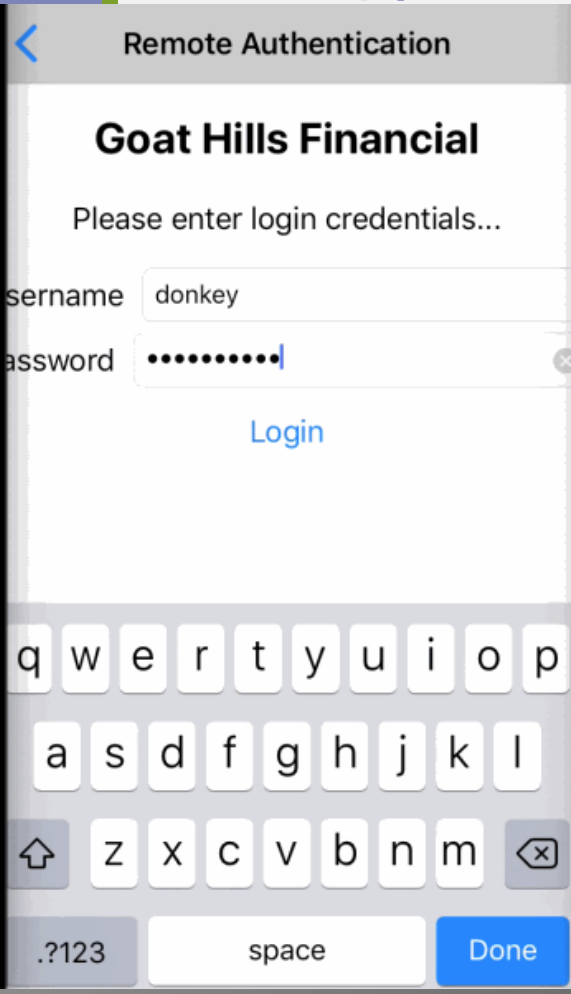
escrypt

Example C (MSTG-STORAGE-10): Search locations in mem. map

escrypt

```
Searching 10 bytes in [0x00000001a0000000-0x00000001acaf0000]
Searching 10 bytes in [0x00000001aeaf0000-0x00000001aec00000]
Searching 10 bytes in [0x00000001aec00000-0x00000001aee00000]
Searching 10 bytes in [0x00000001aee00000-0x00000001af400000]
Searching 10 bytes in [0x00000001af400000-0x00000001afc00000]
Searching 10 bytes in [0x00000001afc00000-0x00000001afe00000]
Searching 10 bytes in [0x00000001afe00000-0x00000001b0000000]
Searching 10 bytes in [0x00000001b0000000-0x00000001b0a00000]
Searching 10 bytes in [0x00000001b0a00000-0x00000001b4600000]
Searching 10 bytes in [0x00000001b4600000-0x00000001b4800000]
Searching 10 bytes in [0x00000001b4800000-0x00000001b4a00000]
Searching 10 bytes in [0x00000001b4a00000-0x00000001b4c00000]
Searching 10 bytes in [0x00000001b4c00000-0x00000001b4e00000]
Searching 10 bytes in [0x00000001b4e00000-0x00000001b5000000]
Searching 10 bytes in [0x00000001b5000000-0x00000001b5200000]
Searching 10 bytes in [0x00000001b5200000-0x00000001b5400000]
Searching 10 bytes in [0x00000001b5400000-0x00000001b5600000]
Searching 10 bytes in [0x00000001b5600000-0x00000001b5800000]
Searching 10 bytes in [0x00000001b5800000-0x00000001b5a00000]
Searching 10 bytes in [0x00000001b5a00000-0x00000001b5c00000]
Searching 10 bytes in [0x00000001b5c00000-0x00000001b5e00000]
Searching 10 bytes in [0x00000001b5e00000-0x00000001b6000000]
Searching 10 bytes in [0x00000001b6000000-0x00000001b6200000]
Searching 10 bytes in [0x00000001b6200000-0x00000001b62dc000]
Searching 10 bytes in [0x00000001b82dc000-0x00000001be510000]
Searching 10 bytes in [0x00000001c0000000-0x00000001c8000000]
Searching 10 bytes in [0x00000001c8000000-0x00000001d0000000]
Searching 10 bytes in [0x00000001d0000000-0x00000001d8000000]
Searching 10 bytes in [0x00000001d8000000-0x00000001e0000000]
hits: 3
0x1c0276dc0 hit1_0 owasp-mstg
0x1c4017870 hit1_1 owasp-mstg
0x1c4017c10 hit1_2 owasp-mstg

[0x1c0276dc0]>
```

Click to Play



escrypt
SECURITY. TRUST. SUCCESS.

- ✓ Reverse Engineer with Cutter
- ✓ Radare2 + swift demangling, what about the new `\swiD` ??
- ✓ Inspecting iOS Frameworks with passionfruit
- ✓ **Your** next contributions
- ✓ …

Random r2 grep tips ~

```
Usage: [command]~[modifier][word,word][endmodifier][[column]][:line]

      ~?<br>
      ~+
      ij~{core}, ij~{core.file}, ij~{core}~{}
      ?*~...
```


Takeaways

escrypt
SECURITY. TRUST. SUCCESS.

- ✓ Use the MASVS
- ✓ Read the MSTG
- ✓ grepharder
- ✓ Learn FRIDA
- ✓ Learn 
- ✓ Contribute!
- ✓ Have fun :)

Thank you, any questions?

@OWASP_MSTG

@grepharder

escrypt
SECURITY. TRUST. SUCCESS.

OWASP MSTG and MASVS

https://github.com/OWASP/owasp-mstg
https://github.com/OWASP/owasp-masvs

Apps

https://github.com/OWASP/owasp-mstg/tree/master/Crackmes
https://github.com/OWASP/MSTG-Hacking-Playground
https://github.com/OWASP/iGoat-Swift

Tools

https://frida.re/
https://github.com/nowsecure/r2frida
https://www.radare.org/r/
https://github.com/sensepost/objection
https://github.com/chaitin/passionfruit

ESCRYPT GmbH
Headquarters
Wittener Straße 45
44879 Bochum
Germany

Phone: +49 234 43870-200
Fax: +49 234 43870-211

info@escrypt.com
www.escrypt.com