A mobile application vulnerability scanner, designed for DevOps process integration, that is built to protect your customers' privacy and defend their devices against modern threats.

Scan report dated **02/07/2022**

DVIA-v2.zip

App ID
com.highaltitudehacks.DVIAswiftv2

Version 2.0

Statistics

19 vulnerabilities found

High-severity vulnerabilities	Medium-severity vulnerabilities	Low-severity vulnerabilities
5	13	1
Can lead to user data being leaked and to a serious violation of the user's privacy	May affect the app's level of protection and reduce the level of security	Violate best practices
27%	67%	6%

List of vulnerabilities

	Category	Level	Amount
1	Cross-site scripting	High	1
2	Hardcoded token or password	High	1
3	Hardcoded cryptographic key	High	2
4	Possible cache substitution	High	1
5	Insecure App Transport Security configuration	Medium	1
6	Usage of deeplinks	Medium	1

7	Content injection	Medium	1
8	Use of insecure HTTP protocol	Medium	10
9	Local network address	Low	1

Vulnerabilities in the code

Cross-site scripting

Found in the file **DVIA-v2/Vulnerabilities/Client Side Injection/Controller/ClientSideInjectionDetailViewController.swift**

```
36
37  extension ClientSideInjectionDetailViewController: UITextFieldDelegate {
38      func textFieldShouldReturn(_ textField: UITextField) -> Bool {
39          guard let name: String = nameTextField.text else { return false }
40          webView.loadHTMLString("Hello \(name), I am inside a WebView!", baseURL: nil)
41          nameTextField.resignFirstResponder()
42          return true
43      }
```

Vulnerability description

XSS or Cross-site scripting is a kind of attack where malicious scripts are inserted into a WebView page. An attacker can influence a URL that is loaded, JavaScript code that is executed, or files that are stored in public directories. In this event, harmful code may be processed and executed by the built-in browser. Execution of malicious scripts might cause unintended information leakage, modification of settings on the server side via bypassed CSRF protection, and more. On iOS there is also a risk of access to built-in WebView handlers such as script messages, URL handlers or content intended for internal communications by the web page and the mobile app, thus exposing internal app logic and functionality.

Remediation

Before insertion, client data should be correctly validated and sanitized. In this case, all metacharacters will be escaped. In other cases, XSS is the result of insecure application architecture, when it trusts data received from unprotected inputs.

Links

https://www.owasp.org/index.php/Mobile_Top_10_2014-M7 https://arxiv.org/ftp/arxiv/papers/1304/1304.7451.pdf

https://cwe.mitre.org/data/definitions/79.html

Hardcoded token or password

Found in the file **Pods/couchbase-lite-ios/CouchbaseLite.framework/Headers/CBLReplication.h**

```
101  @property (nonatomic, strong, nullable) id<CBLAuthenticator> authenticator;
102
103  /** The credential (generally username+password) to use to authenticate to the remote database.
104      This can either come from the URL itself (if it's of the form "http://user:pass@example.com")
```

```
105       or be stored in the NSURLCredentialStorage, which is a wrapper around the Keychain. */
106   @property (nonatomic, strong, nullable) NSURLCredential* credential;
107
```

Vulnerability description

A token or password was found. It might be used by an attacker to access restricted services which will cause information leakage, unwanted server setting changes, or other kinds of unrestricted service accesses or modifications.

Remediation

The developer should not hardcode such sensitive data, to prevent leakages.

Links

https://cwe.mitre.org/data/definitions/312.html

Hardcoded cryptographic key

Found in the file **Pods/RealmSwift/RealmSwift/RealmConfiguration.swift**

```
82                                          The compaction will be skipped if another process is accessing it.
83           - parameter objectTypes:        The subset of `Object` subclasses persisted in the Realm.
84         */
85         public init(fileURL: URL? = URL(fileURLWithPath: RLMRealmPathForFile("default.realm"), isDirectory: false),
86                     inMemoryIdentifier: String? = nil,
87                     syncConfiguration: SyncConfiguration? = nil,
88                     encryptionKey: Data? = nil,
89                     readOnly: Bool = false,
90                     schemaVersion: UInt64 = 0,
91                     migrationBlock: MigrationBlock? = nil,
92                     deleteRealmIfMigrationNeeded: Bool = false,
```

Vulnerability description

The cryptographic key is hardcoded in the app. It can be used by an attacker to encrypt or decrypt sensitive data, substitute a different digital signature, etc., reducing this level of data security to nil.

Remediation

The developer shouldn't hardcode encryption keys. Instead, we recommend using secure key creation and storage systems such as the Keychain services API.

Links

https://blog.oversecured.com/Use-cryptography-in-mobile-apps-the-right-way/

https://developer.android.com/training/articles/keystore https://cwe.mitre.org/data/definitions/312.html

---

Hardcoded cryptographic key

Found in the file **Pods/RealmSwift/RealmSwift/RealmConfiguration.swift**

```
86                     inMemoryIdentifier: String? = nil,
87                     syncConfiguration: SyncConfiguration? = nil,
88                     encryptionKey: Data? = nil,
89                     readOnly: Bool = false,
90                     schemaVersion: UInt64 = 0,
91                     migrationBlock: MigrationBlock? = nil,
92                     deleteRealmIfMigrationNeeded: Bool = false,
```

Found in the file **Pods/RealmSwift/RealmSwift/RealmConfiguration.swift**

```
242         }
243
244         internal static func fromRLMRealmConfiguration(_ rlmConfiguration: RLMRealmConfiguration) -> Configuration {
245             var configuration = Configuration()
246             configuration._path = rlmConfiguration.fileURL?.path
247             configuration._inMemoryIdentifier = rlmConfiguration.inMemoryIdentifier
248             if let objcSyncConfig = rlmConfiguration.syncConfiguration {
```

Vulnerability description

The cryptographic key is hardcoded in the app. It can be used by an attacker to encrypt or decrypt sensitive data, substitute a different digital signature, etc., reducing this level of data security to nil.

Remediation

The developer shouldn't hardcode encryption keys. Instead, we recommend using secure key creation and storage systems such as the Keychain services API.

Links

https://blog.oversecured.com/Use-cryptography-in-mobile-apps-the-right-way/

https://developer.android.com/training/articles/keystore https://cwe.mitre.org/data/definitions/312.html

---

Possible cache substitution

Found in the file **DVIA-v2/Vulnerabilities/Transport Layer Protection/Controller/TransportLayerProtectionViewController.swift**

```
99          var request = URLRequest(url: url.standardized)
100         request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
101         request.httpMethod = "POST"
102         let postDictionary = [
103             "card_number" : cardNumberTextField.text,
104             "card_name" : nameOnCardTextField.text,
105             "card_cvv" : CVVTextField.text
106         ]
107         let jsonData = try? JSONSerialization.data(withJSONObject: postDictionary, options: .prettyPrinted)
108         request.setValue("application/x-www-form-urlencoded; charset=utf-8", forHTTPHeaderField: "Content-Type")
109         request.httpBody = jsonData
110
111         let task = URLSession.shared.dataTask(with: request) { data, response, error in
112             guard let data = data, error == nil else {                                                  // check for
    fundamental networking error
113                 print("error=\(String(describing: error))")
114                 return
```

Vulnerability description

An attacker has the ability to substitute the cache of HTTP requests, which can lead to a whole series of possible attacks. For example, an attacker may replace the cache content for any URL, meaning that when the user tries to visit that address they will see content prepared by the attacker in place of the original. In its turn, this can lead to the user being defrauded or to other attacks being prepared against the device.

Remediation

Restrict the ability of third-parties to affect the cache.

Links

https://capec.mitre.org/data/definitions/141.html

Insecure App Transport Security configuration

Found in the file **DVIA-v2/Info.plist**

```
32          <string>1</string>
33          <key>LSRequiresIPhoneOS</key>
34          <true/>
35          <key>NSAppTransportSecurity</key>
36          <dict>
37                  <key>NSAllowsArbitraryLoads</key>
38                  <true/>
39          </dict>
40          <key>NSCameraUsageDescription</key>
41          <string>To demonstrate the misuse of Camera, please grant it permission once.</string>
```

Vulnerability description

The app uses ATS settings that permit an insecure internet connection over HTTP or old versions of SSL. This may lead to Man-in-the-Middle attacks resulting in sensitive user data being leaked and/or modified.

Remediation

You should update the server settings to work via secure data transfer protocols, then be sure to remove the insecure ATS settings.

Links

https://cwe.mitre.org/data/definitions/319.html https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication

https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

---

Usage of deeplinks

Found in the file **DVIA-v2/Info.plist**

```
16          <string>APPL</string>
17          <key>CFBundleShortVersionString</key>
18          <string>2.0</string>
19          <key>CFBundleURLTypes</key>
20          <array>
21                  <dict>
22                          <key>CFBundleURLName</key>
23                          <string>com.highaltitudehacks.DVIAswiftv2</string>
24                          <key>CFBundleURLSchemes</key>
25                          <array>
26                                  <string>dvia</string>
27                                  <string>dviaswift</string>
28                          </array>
29                  </dict>
30          </array>
31          <key>CFBundleVersion</key>
32          <string>1</string>
33          <key>LSRequiresIPhoneOS</key>
```

Vulnerability description

Remediation

Vulnerability description

The app can receive and process files and URLs from external sources, which can lead to a wide range of security problems: injections of various kinds, path traversal, leaking of user sessions and even arbitrary code execution.

Remediation

It is important not to trust data obtained from external sources. Such data should be subjected to the maximum possible verification. It is worth refraining from passing URL data via deeplinks: instead, describe the set of various actions that the user can perform.

Links

https://cwe.mitre.org/data/definitions/939.html

Content injection

Found in the file **DVIA-v2/AppDelegate.swift**

```
34          return true
35      }
36
37      func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey : Any] = [:]) -> Bool {
38
39          let splitUrl = url.absoluteString.components(separatedBy: "/phone/call_number/")
40          if ((Int(splitUrl[1])) != nil){
41              //Valid URL, since the argument is a number
42              let alertController = UIAlertController(title: "Success!", message: "Calling \(splitUrl[1]). Ring Ring !!!",
preferredStyle: .alert)
43              let okAction = UIAlertAction(title: "OK", style: UIAlertActionStyle.default, handler: nil)
44              alertController.addAction(okAction)
45              window?.rootViewController?.present(alertController, animated: true, completion: nil)
```

Vulnerability description

An attacker has the ability to install arbitrary text within the app, which may lead to phishing attacks: the user will be inclined to trust messages from the app.

Remediation

It is recommended that you add text that is controlled externally into fields like UITextField, rather than UITextView, or limit the possibility of adding such content at all. The user needs to be sure it is not a system message.

Links

https://developer.apple.com/design/human-interface-guidelines/ios/controls/text-fields/ https://wiki.owasp.org/index.php/Phishing

https://cwe.mitre.org/data/definitions/74.html

Use of insecure HTTP protocol

Found in the file **Pods/RealmSwift/RealmSwift/Sync.swift**

```
400       completion block will be called with an error.
401
402       - parameter credentials: A `SyncCredentials` object representing the user to log in.
403       - parameter authServerURL: The URL of the authentication server (e.g. "http://realm.example.org:9080").
404       - parameter timeout: How long the network client should wait, in seconds, before timing out.
405       - parameter callbackQueue: The dispatch queue upon which the callback should run. Defaults to the main queue.
406       - parameter completion: A callback block to be invoked once the log in completes.
```

Found in the file **Pods/Realm/include/RLMSyncUser.h**

```
148   The completion block always runs on the main queue.
149
```

```
150   @param credentials      A credentials value identifying the user to be logged in.
151   @param authServerURL    The URL of the authentication server (e.g. "http://realm.example.org:9080").
152   @param completion       A callback block that returns a user object or an error,
153                           indicating the completion of the login operation.
154   */
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **Pods/Flurry-iOS-SDK/Flurry/Flurry.h**

```
65  *
66  *  @note This class serves as a delegate for Flurry. \n
67  *  For additional information on how to use Flurry's Ads SDK to
68  *  attract high-quality users and monetize your user base see <a href="http://wiki.flurry.com/index.php?title=Publisher">Support Center - Publisher</a>.
69  *  @author 2010 - 2014 Flurry, Inc. All Rights Reserved.
70  *  @version 6.3.0
71  *
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **DVIA-v2/Home/HomeViewController.swift**

```
62                      destinationVC.urlToLoad = "https://twitter.com/prateekg147"
63                      destinationVC.navigationItem.title = "Twitter"
64                  }else{
65                      destinationVC.urlToLoad = "http://damnvulnerableiosapp.com"
66                      destinationVC.navigationItem.title = "DVIA"
67                  }
```

```
68              }
```

Found in the file **DVIA-v2/Constants.swift**

```
35          case .insecureDataStorageArticle:
36              return "http://highaltitudehacks.com/2013/10/26/ios-application-security-part-20-local-data-storage-nsuserdefaults"
37          case .homePage:
38              return "http://damnvulnerableiosapp.com"
39          case .jailbreakDetetionArticle:
40              return "http://highaltitudehacks.com/2013/12/17/ios-application-security-part-24-jailbreak-detection-and-evasion"
41          case .runtimeArticle1:
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **Pods/couchbase-lite-ios/CouchbaseLite.framework/Headers/CBLReplication.h**

```
101 @property (nonatomic, strong, nullable) id<CBLAuthenticator> authenticator;
102
103 /** The credential (generally username+password) to use to authenticate to the remote database.
104     This can either come from the URL itself (if it's of the form "http://user:pass@example.com")
105     or be stored in the NSURLCredentialStorage, which is a wrapper around the Keychain. */
106 @property (nonatomic, strong, nullable) NSURLCredential* credential;
107
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **Pods/Flurry-iOS-SDK/Flurry/Flurry.h**

```
732  *  are extremely valuable as they allow you to store characteristics of an action. For example,
733  *  if a user purchased an item it may be helpful to know what level that user was on.
734  *  By setting this parameter you will be able to view a distribution of levels for the purcahsed
```

```
735    *  event on the <a href="http://dev.flurry.com">Flurrly Dev Portal</a>.
736    *
737    *  @note You should not pass private or confidential information about your users in a
738    *  custom event. \n
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **Pods/Parse/README.md**

```
12 <p align="center">
13     <a href="https://twitter.com/intent/follow?screen_name=parseplatform"><img alt="Follow on Twitter"
src="https://img.shields.io/twitter/follow/parseplatform?style=social&label=Follow"></a>
14     <a href="https://github.com/parse-community/Parse-SDK-iOS-OSX/issues/1356"><img alt="Maintenance help wanted"
src="https://img.shields.io/badge/maintenance-help%20wanted-red.svg"></a>
15     <img alt="Platforms" src="http://img.shields.io/cocoapods/p/Parse.svg?style=flat">
16     <a href=" https://github.com/parse-community/Parse-SDK-iOS-OSX/blob/master/LICENSE"><img alt="License"
src="https://img.shields.io/badge/license-BSD-lightgrey.svg"></a>
17     <a href="https://cocoapods.org/pods/Parse"><img alt="Podspec" src="https://img.shields.io/cocoapods/v/Parse.svg"></a>
18     <a href="#backers"><img alt="Backers on Open Collective" src="https://opencollective.com/parse-server/backers/badge.svg" />
</a>
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **DVIA-v2/Vulnerabilities/Attacking Third Party Libraries/Controller/FlurryLeakDetailsViewController.swift**

```
42          Flurry.logEvent("PhoneEntered", withParameters:params as Any as! [AnyHashable : Any])
43
44          let queryItem = URLQueryItem(name: "q", value: "\(textField.text!)")
```

```
45          let url = URL(string: "http://google.com?\(queryItem)")
46          let request = URLRequest(url: url!)
47          var connection = NSURLConnection(request: request, delegate: nil, startImmediately: true)
48                  return true
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **Pods/Parse/Parse/Parse/PFInstallation.m**

```
315   Save localeIdentifier in the following format: [language code]-[COUNTRY CODE].
316
317   The language codes are two-letter lowercase ISO language codes (such as "en") as defined by
318   <a href="http://en.wikipedia.org/wiki/ISO_639-1">ISO 639-1</a>.
319   The country codes are two-letter uppercase ISO country codes (such as "US") as defined by
320   <a href="http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3">ISO 3166-1</a>.
321
322   Many iOS locale identifiers don't contain the country code -> inconsistencies with Android/Windows Phone.
323   */
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **DVIA-v2/Vulnerabilities/Donate/Controller/DonateViewController.swift**

```
58              self.navigationItem.title = " "
59              if let destinationVC:DonateDetailsViewController = segue.destination as? DonateDetailsViewController {
60                  if(!video){
61                  destinationVC.urlToLoad = "http://www.thejuniperfund.org/"
62                  }else{
63                  destinationVC.urlToLoad = "https://www.youtube.com/watch?v=HsV6jaA5J2I"
```

```
64                 }
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Use of insecure HTTP protocol

Found in the file **DVIA-v2/Vulnerabilities/Runtime Manipulation/Controller/RuntimeManipulationDetailsViewController.swift**

```
21     @IBOutlet var passwordTextField: UITextField!
22     @IBOutlet var codeTextField: UITextField!
23     let numToolbar: UIToolbar = UIToolbar()
24     let tutorialUrl:String = "http://highaltitudehacks.com/2013/11/08/ios-application-security-part-21-arm-and-gdb-basics"
25
26     override func viewDidLoad() {
27         super.viewDidLoad()
```

Found in the file **DVIA-v2/Vulnerabilities/Side Channel Data Leakage/Controller/Objective-C Methods/SetSharedCookies.m**

```
16    #import <Foundation/Foundation.h>
17    #import "SetSharedCookies.h"
18
19    static NSString *const SiteURLString = @"http://highaltitudehacks.com";
20    static NSString *const CookieUsername = @"admin123";
21    static NSString *const CookiePassword = @"dvpassword";
22
```

Found in the file **DVIA-v2/Vulnerabilities/Side Channel Data Leakage/Controller/CookiesViewController.swift**

```
17
18     fileprivate let CookieUsername = "admin123"
19     fileprivate let CookiePassword = "dvpassword"
20     fileprivate let SiteURLString = "http://highaltitudehacks.com"
21
22
23     class CookiesViewController: UIViewController {
```

Found in the file **DVIA-v2/Constants.swift**

```
33     var url: String {
34         switch self {
35         case .insecureDataStorageArticle:
36             return "http://highaltitudehacks.com/2013/10/26/ios-application-security-part-20-local-data-storage-nsuserdefaults"
37         case .homePage:
38             return "http://damnvulnerableiosapp.com"
39         case .jailbreakDetetionArticle:
40             return "http://highaltitudehacks.com/2013/12/17/ios-application-security-part-24-jailbreak-detection-and-evasion"
```

```
41          case .runtimeArticle1:
42              return "http://highaltitudehacks.com/2013/06/16/ios-application-security-part-3-understanding-the-objective-c-runtime"
43          case .runtimeArticle2:
44              return "http://highaltitudehacks.com/2013/07/02/ios-aios-appllication-security-part-4-runtime-analysis-using-cycript-yahoo-weather-app"
45          case .runtimeArticle3:
46              return "http://highaltitudehacks.com/2013/07/02/ios-application-security-part-5-advanced-runtime-analysis-and-manipulation-using-cycript-yahoo-weather-app"
47          case .runtimeArticle4:
48              return "http://highaltitudehacks.com/2013/07/25/ios-application-security-part-8-method-swizzling-using-cycript"
49          case .runtimeArticle5:
50              return "http://highaltitudehacks.com/2013/09/17/ios-application-security-part-16-runtime-analysis-of-ios-applications-using-inalyzer"
51          case .runtimeArticle6:
52              return "http://highaltitudehacks.com/2013/11/08/ios-application-security-part-21-arm-and-gdb-basics"
53          case .runtimeArticle7:
54              return "http://highaltitudehacks.com/2013/12/17/ios-application-security-part-22-runtime-analysis-and-manipulation-using-gdb"
55          case .URLSchemeArticle:
56              return "http://highaltitudehacks.com/2014/03/07/ios-application-security-part-30-attacking-url-schemes"
57          case .transportLayerArticle:
58              return "http://highaltitudehacks.com/2013/08/20/ios-application-security-part-11-analyzing-network-traffic-over-http-slash-https"
59          case .patchingApplicationArticle1:
60              return "http://highaltitudehacks.com/2013/12/17/ios-application-security-part-26-patching-ios-applications-using-ida-pro-and-hex-fiend"
61          case .patchingApplicationArticle2:
62              return "http://highaltitudehacks.com/2014/01/17/ios-application-security-part-28-patching-ios-application-with-hopper"
63          }
64      }
65  }
```

Vulnerability description

The mobile application uses the insecure HTTP protocol to communicate with the server. HTTP lacks encryption, so sensitive data like username, password, etc. can be easily intercepted and replaced by an attacker who is connected to the same network as the user's device—for instance, if the user is using a public WiFi network.

Remediation

Replace all http links in the application with their https equivalents.

Links

https://cwe.mitre.org/data/definitions/319.html https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Local network address

Found in the file **Pods/couchbase-lite-ios/CouchbaseLite.framework/Headers/CBLReplication.h**

```
101  @property (nonatomic, strong, nullable) id<CBLAuthenticator> authenticator;
102
103  /** The credential (generally username+password) to use to authenticate to the remote database.
104      This can either come from the URL itself (if it's of the form "http://user:pass@example.com")
105      or be stored in the NSURLCredentialStorage, which is a wrapper around the Keychain. */
106  @property (nonatomic, strong, nullable) NSURLCredential* credential;
107
```

Vulnerability description

The application includes links to local addresses of applications that probably relate to the development infrastructure and indicate the internal structure of of the developer's company network. These resources may be attacked by malicious parties.

Remediation

Delete links to local resources from the release version of your application.

Links

https://cwe.mitre.org/data/definitions/359.html https://cwe.mitre.org/data/definitions/215.html

---