datatheorem VIEW IN PORTAL

Enable Sandboxed JavaScript Injection

CultureNEXT - PreProd 1.0.0   ID $959773   Severity: Medium   No Priority

The following code locations within the App leverage the `WKWebView` APIs to inject JavaScript into a web page:

OPEN since 4/10/2021 `_unsymbolicated_function` calls `-[(id) evaluateJavaScript:completionHandler:]`

Stack Trace

The following related source code symbols were identified:

```
-[PWPushManagerJSBridge postEvent:]
-[PWPushManagerJSBridge webScriptNameForSelector:]
-[PWPushManagerJSBridge isSelectorExcludedFromWebScript:]
@"%@('%@')"
@"%@()"
-[_OBJC_CLASS_$_NSString stringWithFormat:]
-[_OBJC_CLASS_$_NSString stringWithFormat:]
-[(id) localizedDescription]
```

This code was identified within the `cultureNEXT.app/cultureNEXT` binary.

OPEN since 4/13/2021 `-[PWWebClient webView:didFinishNavigation:]` calls `-[(id) evaluateJavaScript:completionHandler:]`

Stack Trace

The following related source code symbols were identified:

```
-[PWWebClient loadPushwooshUrl:]
-[PWWebClient webView:decidePolicyForNavigationAction:decisionHandler:]
-[PWWebClient webView:didFailNavigation:withError:]
@"-[PWWebClient webView:didFinishNavigation:]"
@"window.pushwoosh._customData = %@;"
@"webViewDidFinishLoad"
-[PushNotificationManager pushManager]
-[PushNotificationManager getCustomPushData:]
-[_OBJC_CLASS_$_NSString stringWithFormat:]
```

This code was identified within the `cultureNEXT.app/cultureNEXT` binary.

OPEN since 4/13/2021 `-[WKWebView (PWSynchronousEvaluateJavaScript) pw_stringByEvaluatingJavaScriptFromString:error:]` calls `-[(id) evaluateJavaScript:completionHandler:]`

Stack Trace

The following related source code symbols were identified:

```
-[PWUtils stopBackgroundTask:]
-[PWUtilsMobile startBackgroundTask]
-[PWUtils startBackgroundTask]
@"Timed out"
@""
@""
-[_OBJC_CLASS_$_NSRunLoop (unknown)]
-[_OBJC_CLASS_$_NSRunLoop (unknown)]
-[_OBJC_CLASS_$_NSDate dateWithTimeIntervalSinceNow:]
```

This code was identified within the `cultureNEXT.app/cultureNEXT` binary.

View 3 more components

DESCRIPTION

The App utilizes the `WKWebView` APIs that allow the App to inject JavaScript into web content without also leveraging platform APIs to sandbox the JavaScript from untrusted code.

Starting with iOS 14, `WKWebView` offers the following APIs that enable injecting JavaScript into a web page while also mitigating collision attacks from untrusted JavaScript:

`evaluateJavaScript:inFrame:inContentWorld:completionHandler:` `callAsyncJavaScript:arguments:inFrame:inContentWorld:completionHandler:`

By providing a `WKContentWorld` instance to the `inContentWorld:` parameter to these methods, iOS will ensure that the JavaScript injected into the web page will run in a sandboxed environment from the JavaScript originating from the web, which may be untrusted.

Without the use of `WKContentWorld`, injected App JavaScript and native web JavaScript run in a shared environment. This may lead to multiple attacks and bug scenarios, such as when a malicious web page intentionally declares new JavaScript functions or symbols with the same names as JavaScript injected by the App. Additionally, the WebKit team observed malicious analytics SDKs intentionally communicating with web JavaScript to pass along user tracking information only available through native APIs. As unsandboxed JavaScript injection may result in a variety of unexpected behaviors, as well as data leakage, all JavaScript injection should be performed within an isolated sandbox.

More documentation on `WKContentWorld` is available in this WWDC20 session.

RECOMMENDATION

When injecting JavaScript into a web page, utilize the iOS 14 APIs that allow sandboxed JavaScript execution. To do so, pass a `WKContentWorld` instance for the App JavaScript to execute within, which ensures code isolation from third-party JavaScript.

SECURE CODE

```
- (void)presentWebViewWithInjectedJS {
    // Set up a WKWebView, then inject JavaScript into the web page within an isolated execution context
    WKWebViewConfiguration* webViewConfiguration = [[WKWebViewConfiguration alloc] init];
    WKWebView *webView = [[WKWebView alloc] initWithFrame:self.view.frame configuration:webViewConfiguration];
    [webView loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:@"https://shop.example"]]];

    NSString* javaScriptToInject = @"alert('Symbols defined here are isolated from any untrusted web code');";
    // Fetch the sandbox within which our JavaScript should execute
    WKContentWorld* sandbox = [WKContentWorld defaultClientWorld];

    [webView evaluateJavaScript:javaScriptToInject
             inFrame:nil
             inContentWorld:sandbox
             completionHandler:^(id value, NSError *error) {
        NSLog(@"JS Result: %@, error: %@", value, error);
    }];
}
```

QUESTIONS & COMMENTS