Mobile implementation

- WebView integration
  - Requirements
    - JavaScript support
    - Network access
    - Environment consistency
  - Platform-specific implementation
    - Android WebView
    - iOS WKWebView (Swift)
    - React Native WebView
    - Flutter WebView
- Common implementation issues
  - User Agent consistency
  - Content Security Policy (CSP)
  - Domain configuration
  - Cookie and storage issues

Turnstile requires a browser environment because it runs JavaScript challenges in the visitor's browser. On mobile devices, Turnstile works in mobile browsers without additional configuration.

For native mobile applications, Turnstile does not run natively. Instead, you use a WebView — a browser component embedded inside your native app — to load a webpage that contains the Turnstile widget.

WebView integration

A WebView embeds a browser engine within your native application, enabling you to show web pages, forms, and JavaScript-powered content like Turnstile widgets.

Requirements

For Turnstile to function properly in WebView, the following requirements must be met.

JavaScript support

- JavaScript execution must be enabled.
- DOM storage API must be available.
- Standard web APIs must be accessible.

Network access

- Access to `challenges.cloudflare.com`
- Support for both HTTP and HTTPS connections.
- Allow connections to `about:blank` and `about:srcdoc`

Environment consistency

- Consistent User Agent throughout the session
- Stable device and browser characteristics
- No modification to core browser behavior

Platform-specific implementation

Android WebView

```
WebView webView = findViewById(R.id.webview);
WebSettings webSettings = webView.getSettings();

// Required: Enable JavaScript
webSettings.setJavaScriptEnabled(true);

// Required: Enable DOM storage
webSettings.setDomStorageEnabled(true);

// Recommended: Enable other web features
webSettings.setLoadWithOverviewMode(true);
webSettings.setUseWideViewPort(true);
webSettings.setAllowFileAccess(true);
webSettings.setAllowContentAccess(true);
```

```
// Load your web content with Turnstile
webView.loadUrl("https://yoursite.com/protected-form");
```

iOS WKWebView (Swift)

```
import WebKit

class ViewController: UIViewController {
    @IBOutlet weak var webView: WKWebView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Configure WebView
        let configuration = WKWebViewConfiguration()
        configuration.preferences.javaScriptEnabled = true

        // Load your web content with Turnstile
        if let url = URL(string: "https://yoursite.com/protected-form") {
            webView.load(URLRequest(url: url))
        }
    }
}
```

React Native WebView

```
import { WebView } from "react-native-webview";

export default function App() {
  return (
    <WebView
      source={{ uri: "https://yoursite.com/protected-form" }}
      javaScriptEnabled={true}
      domStorageEnabled={true}
      allowsInlineMediaPlayback={true}
      mediaPlaybackRequiresUserAction={false}
    />
  );
}
```

Flutter WebView

```
import 'package:flutter_inappwebview/flutter_inappwebview.dart';

class WebViewScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return InAppWebView(
      initialUrlRequest: URLRequest(
        url: Uri.parse('https://yoursite.com/protected-form')
      ),
      initialOptions: InAppWebViewGroupOptions(
        crossPlatform: InAppWebViewOptions(
          javaScriptEnabled: true,
          useShouldOverrideUrlLoading: false,
        ),
        android: AndroidInAppWebViewOptions(
          domStorageEnabled: true,
        ),
        ios: IOSInAppWebViewOptions(
          allowsInlineMediaPlayback: true,
        ),
      ),
    );
  }
}
```

Common implementation issues

User Agent consistency

Changing the User Agent during a session causes Turnstile challenges to fail because the system relies on consistent browser characteristics to validate the visitor's authenticity. When the User Agent changes mid-session, Turnstile treats this as a potential security risk and rejects the challenge.

```
// Android - Set consistent User Agent
webSettings.setUserAgentString(webSettings.getUserAgentString());
```

```
// iOS - Maintain default User Agent
webView.customUserAgent = webView.value(forKey: "userAgent") as? String
```

Content Security Policy (CSP)

Strict Content Security Policy settings can prevent Turnstile from loading the necessary scripts and making required network connections. This happens when CSP headers or meta tags block access to the domains and resources that Turnstile needs to function properly.

```
<meta
  http-equiv="Content-Security-Policy"
  content="
    default-src 'self';
    script-src 'self' challenges.cloudflare.com 'unsafe-inline';
    connect-src 'self' challenges.cloudflare.com;
    frame-src 'self' challenges.cloudflare.com;
  "
/>
```

Domain configuration

WebView security restrictions can prevent access to the domains that Turnstile requires for proper operation. Some WebViews are configured to only allow specific domains or block certain types of connections, which can interfere with Turnstile's ability to load challenges and communicate with Cloudflare's servers.

To resolve this, configure your WebView's allowed origins to include all domains that Turnstile needs:

- `challenges.cloudflare.com`
- `about:blank`
- `about:srcdoc`
- Your own domain(s)

The exact configuration method varies by platform, but the principle is to explicitly allow network access for these domains.

Cookie and storage issues

Cookies and local storage not persisting between sessions can cause Turnstile to fail because it relies on these mechanisms to maintain state and track visitor behavior. This commonly occurs when WebView storage settings are too restrictive or when the app clears storage between sessions. Ensure that your WebView is configured to properly handle cookies and local storage.

```
// Android - Enable cookies
CookieManager.getInstance().setAcceptCookie(true);
CookieManager.getInstance().setAcceptThirdPartyCookies(webView, true);
```

```
// iOS - Configure cookie storage
webView.configuration.websiteDataStore = WKWebsiteDataStore.default()
```

□□□ #2
□ □□□ □□□ 2026-11-01 04:59:29 UTC
□ □□□ □□□ 2026-11-01 04:59:31 UTC