АНАЛІЗ ВРАЗЛИВОСТЕЙ КОМПОНЕНТА WEBVIEW У МОБІЛЬНИХ ДОДАТКАХ ANDROID ТА МЕТОДИ ЇХ УСУНЕННЯ

Іванов Є.В., Сидоренко З.М.
Харківський національний університет радіоелектроніки, Харків, Україна

WebView – це масштабний програмний компонент, що дозволяє використовувати веб-контент у додатках [1]. У сучасній розробці на платформі Android компонент WebView набув надзвичайної популярності завдяки своїй здатності значно оптимізувати процес створення програмного забезпечення. Саме тому у деяких випадках уся функціональність додатка реалізована через веб-браузер.

Однак, оскільки WebView функціонує як своєрідний міст між зовнішнім веб-середовищем та внутрішніми нативними API операційної системи, він суттєво розширює поверхню атаки та стає привабливим для зловмисників. Аналіз вразливостей WebView та своєчасне застосування методів їх усунення здатні суттєво зменшити такі ризики, як: несанкціонований доступ до локальних файлів, витік сесійних даних через XSS, обхід механізмів пісочниці, а також віддалене виконання коду шляхом експлуатації небезпечних JavaScript-інтерфейсів [2].

Метою доповіді є дослідження вразливостей, що виникають при використанні компонента WebView в додатках на платформі Android, аналіз векторів атак на них та надання рекомендацій щодо їх усунення для підвищення загального рівня безпеки мобільного програмного забезпечення.

Головна причина виникнення вразливостей полягає у некоректній конфігурації класу WebSettings, який за замовчуванням може надавати надмірні привілеї в залежності від рівня API.

Найбільш критичним вектором атаки є зловживання мостом між JavaScript та нативним кодом Android. Використання методу addJavascriptInterface() дозволяє веб-сторінці викликати Java методи додатку. Якщо у конфігурації WebView дозволено виконання JavaScript коду завдяки методу setJavaScriptEnabled(enabled), то зловмисник може посилити вразливість від XSS до RCE.

Іншим поширеним вектором є експлуатація доступу до локальної файлової системи.

Якщо розробник залишає увімкненими параметри доступу до файлів через схему file://, або до контент провайдерів через схему content://, зловмисник може використати Universal XSS або символічні посилання для читання конфіденційних файлів додатку, до яких належать бази даних SQLite, файли SharedPreferences або токени сесій, що зберігаються в ізольованому середовищі.

Для підвищення стійкості мобільних додатків та усунення вразливостей компонента WebView необхідно імплементувати наступні елементи захисту на рівні конфігурації [3, 4].

1. Необхідно заборонити несанкціонований доступу до WebView з боку інших додатків на пристрої. Якщо Activity, що містить WebView, не передбачає виклику ззовні, її необхідно жорстко ізолювати, встановивши атрибут android:exported="false" у файлі маніфесту.

2. Вимкнути доступ до контент провайдерів, викликавши WebSettings.setGeolocationEnabled(false).

3. Вимкнути доступ до файлів, викликавши WebSettings.setAllowFileAccess(false), якщо minSdkVersion дорівнює 29 або менше.

4. Вимкнути доступ до файлів за URL-адресами, викликавши WebSettings.setAllowFileAccessFromFileURLs(false), якщо minSdkVersion дорівнює 15 або менше.

5. Заборонити універсальний доступ до файлів за URL-адресами, викликавши WebSettings.setAllowUniversalAccessFromFileURLs(false), якщо minSdkVersion дорівнює 15 або менше.

6. Якщо у WebView завантажуються зовнішні посилання, необхідно обов'язково перевірити, чи правильно завантажено джерело, – перевіривши як схему, так і ім'я хоста.

7. В усіх випадках, коли JavaScript викликається з даними, отриманими ззовні, необхідно переконатися, що вони пройшли санітизацію.

На основі цих принципів розробники можуть побудувати безпечне середовище для відображення веб-контенту, що дозволить мінімізувати поверхню атаки та захистити конфіденційні дані користувачів.

На основі цих досліджень можна стверджувати, що суворе дотримання принципу найменших привілеїв при конфігурації WebSettings та надійна ізоляція компонента від зовнішніх IPC-викликів (Intent) суттєво підвищують стійкість мобільних додатків на платформі Android.

У сукупності з ретельною валідацією вхідних даних це значно ускладнює експлуатацію вразливостей навіть для кваліфікованих зловмисників, що знижує ризик витоку конфіденційної інформації та компрометації сесій користувачів.

Список літератури

1. Contributors to Wikimedia projects. WebView - wikipedia. *Wikipedia, the free encyclopedia.* URL: https://en.wikipedia.org/wiki/WebView (дата звернення: 01.04.2026)/

2.Сєвєрінов, О. В., Шевцов, В. О., Сокол-Кутиловська, А. С. (2017). Аналіз сучасних методів атак на електронні ресурси органів управління. Системи озброєння і військова техніка, (1), 65-68.

3. Android security checklist: WebView. *News, Techniques & Guides.* URL: https://blog.oversecured.com/Android-security-checklist-webview#recommendations (дата звернення: 02.04.2026).

4. Mobile application security weakness enumeration (MASWE) - OWASP mobile application security. *OWASP Mobile Application Security - OWASP Mobile Application Security.* URL: https://mas.owasp.org/MASWE/ (дата звернення: 01.04.2026).