Javascript Injection in Android Applications

Basil Schöni

Software Composition Group
Institute of Computer Science
University of Berne

28.05.19
# Table of Contents

1. The Problem
2. The Hunt
3. The Loot
4. The End
Phones store a lot of sensitive data:
Privacy Implications of Mobile Devices

Phones store a lot of sensitive data:

- call history, messages, pictures, e-mails
Privacy Implications of Mobile Devices

Phones store a lot of sensitive data:

- call history, messages, pictures, e-mails
- location, speed, position, audio, video
Privacy Implications of Mobile Devices

Phones store a lot of sensitive data:
- call history, messages, pictures, e-mails
- location, speed, position, audio, video
- shopping, banking, medical
What Is a WebView?

WebViews allow developers to build hybrid apps.
What Is a WebView?

WebViews allow developers to build hybrid apps.
- Basically a browser engine that renders webpages
What Is a WebView?

WebViews allow developers to build hybrid apps.

- Basically a browser engine that renders webpages
- Comfortable way to build OS-agnostic applications
What Is a WebView?

WebViews allow developers to build hybrid apps.

- Basically a browser engine that renders webpages
- Comfortable way to build OS-agnostic applications
- Sandboxed. Possible to define bridges to native app code
What Is a WebView?

WebViews allow developers to build hybrid apps.

- Basically a browser engine that renders webpages
- Comfortable way to build OS-agnostic applications
- Sandboxed. Possible to define bridges to native app code
- Mixing of data and code
What Is Cross Site Scripting?

Cross Site Scripting is an important security risk for web applications.
What Is Cross Site Scripting?

Cross Site Scripting is an important security risk for web applications.

- There is some attacker-controllable data input
What Is Cross Site Scripting?

Cross Site Scripting is an important security risk for web applications.

- There is some attacker-controllable data input
- The data does not get sanitized
What Is Cross Site Scripting?

Cross Site Scripting is an important security risk for web applications.

- There is some attacker-controllable data input
- The data does not get sanitized
- The data is made part of an HTML document that is displayed to the user
Table of Contents

1. The Problem
2. The Hunt
3. The Loot
4. The End
Pipeline

Getting the data:
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories
2. Decompile APKs
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   - Selection based on metadata that was filtered by categories
2. Decompile APKs
   - Throw out apps that decompiled with errors
Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   - Selection based on metadata that was filtered by categories

2. Decompile APKs
   - Throw out apps that decompiled with errors
   - Throw out apps without 'dangerous' permissions

3. Run analyzer script against decompiled apps
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions

3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories
2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions
3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
   b. Check filenames against blacklist
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions

3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
   b. Check filenames against blacklist
   c. Build AST from javascript code
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions

3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
   b. Check filenames against blacklist
   c. Build AST from javascript code
   d. Find search terms in AST
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without ’dangerous’ permissions

3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
   b. Check filenames against blacklist
   c. Build AST from javascript code
   d. Find search terms in AST
   e. Put matching code slices in json file
Pipeline

Getting the data:

1. Download random APKs from AndroZoo
   a. Selection based on metadata that was filtered by categories

2. Decompile APKs
   a. Throw out apps that decompiled with errors
   b. Throw out apps without 'dangerous' permissions

3. Run analyzer script against decompiled apps
   a. Find HTML and JS files
   b. Check filenames against blacklist
   c. Build AST from javascript code
   d. Find search terms in AST
   e. Put matching code slices in json file

4. Delete apps without matches
Pipeline

Analyzing the data:
Pipeline

Analyzing the data:

1. Go through resulting json file
Analyzing the data:

1. Go through resulting json file
2. Find promising code slices while ignoring noise
Pipeline

Analyzing the data:

1. Go through resulting json file
2. Find promising code slices while ignoring noise
3. Follow data from sink to source
Analyzing the data:

1. Go through resulting json file
2. Find promising code slices while ignoring noise
3. Follow data from sink to source
4. Find out if source is attacker controllable
Analyzing the data:

1. Go through resulting json file
2. Find promising code slices while ignoring noise
3. Follow data from sink to source
4. Find out if source is attacker controllable
5. Confirm vulnerability by exploiting it
I concentrated on finding sinks, rather than sources.
I concentrated on finding sinks, rather than sources.

- DOM API sinks
  - elem.innerHTML = data
  - document.write(data)
  - ...
  - Jquery sinks
    - $(selector).html(data)
    - $(selector).append(data)

...
I concentrated on finding sinks, rather than sources.

- **DOM API sinks**
  - `elem.innerHTML = data`
  - `document.write(data)`
  - ...

- **Jquery sinks**
  - `$(selector).html(data)`
  - `$(selector).append(data)`
  - ...
Search Terms

I concentrated on finding sinks, rather than sources.

- **DOM API sinks**
  - `elem.innerHTML = data`
  - `document.write(data)`
  - ...

- **Jquery sinks**
  - `$(selector).html(data)`
  - `$(selector).append(data)`
  - ...

- **DOM API sources**
  - `localStorage.getItem(key)`
  - `document.referrer`
  - ...

Challenges

Along the way, there were some challenges:
Challenges

Along the way, there were some challenges:

- Using the proper tools
Challenges

Along the way, there were some challenges:

- Using the proper tools
- Filtering out the noise
Challenges

Along the way, there were some challenges:

- Using the proper tools
- Filtering out the noise
- Practically confirming vulnerabilities
Table of Contents

1. The Problem
2. The Hunt
3. The Loot
4. The End
Marine Pollution App

Marine Pollution Protection Pocket Checklist
Marine Pollution App

Marine Pollution Protection Pocket Checklist
- Application that helps you comply to maritime laws
Marine Pollution App

Marine Pollution Protection Pocket Checklist

- Application that helps you comply to maritime laws
- 10’000+ installs
Marine Pollution App

Marine Pollution Protection Pocket Checklist

- Application that helps you comply to maritime laws
- 10’000+ installs
- read/write storage, record audio/video, internet, network state, phone state
Marine Pollution App

Sink

```javascript
function updateQuestionNote(qnum, qnid, qnnote) {
    str = '<p class="notesbox">' + qnnote + '</p>;
    $('#mqn' + qnid).html(str)
    $('#tqn' + qnid).html(str)
}
```

Source

```javascript
$('#qnotesave').click(function(e) {
    saveQuestionNote($('#notesform #qnotes').val());
    return false;
})
```
Ing & McKee App
Car Insurance App

Ing & McKee App

- Application that helps you manage your insurance policy and make insurance claims
Car Insurance App

Ing & McKee App

- Application that helps you manage your insurance policy and make insurance claims
- 100+ installs
Car Insurance App

Ing & McKee App

- Application that helps you manage your insurance policy and make insurance claims
- 100+ installs
- write to storage, coarse/fine location, internet, network state
function updateDrivers(tx, results) {
    driverHTML += results(i).firstName;
    $("#allDrivers").prepend(driverHTML);
}

function updateToLatestDrivers() {
    notesDB.transaction(function (tx) {
        tx.executeSql('SELECT * FROM otherDrivers
            where NoteID = ?', [id], updateDrivers);
    });
}
Wikipedia App

Wikipedia
Wikipedia App

Wikipedia

- Application that allows you to access and edit Wikipedia
<table>
<thead>
<tr>
<th>The Problem</th>
<th>The Hunt</th>
<th>The Loot</th>
<th>The End</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Wikipedia App

Wikipedia

- Application that allows you to access and edit Wikipedia
- 10’000’000+ installs
Wikipedia App

Wikipedia

- Application that allows you to access and edit Wikipedia
- 10’000’000+ installs
- write to storage, fine location, get/authenticate/manage accounts, internet, network state, ...
Sink

```javascript
bridge.registerListener(
    "displayLeadSection",
    function( payload ) {
        var content = document.createElement( "div" );
        content.innerHTML = payload.section.text;
        document.getElementById( "content" ).appendChild( content );
    }
);
```
What To Do Next

What I want to do next:

- Find more useful vectors for injecting the code
- Confirm / reject vulnerability in Wikipedia app
- Report findings
- Automate the detection of injection vulnerabilities
What To Do Next

What I want to do next:

- Find more useful vectors for injecting the code
What To Do Next

What I want to do next:

- Find more useful vectors for injecting the code
- Confirm / reject vulnerability in Wikipedia app
What To Do Next

What I want to do next:
- Find more useful vectors for injecting the code
- Confirm / reject vulnerability in Wikipedia app
- Report findings
What To Do Next

What I want to do next:

- Find more useful vectors for injecting the code
- Confirm / reject vulnerability in Wikipedia app
- Report findings
- Automate the detection of injection vulnerabilities
Thank You for Your Attention.

Questions?