The Emperor’s New Autofill Framework: A Security Analysis of Autofill on iOS and Android

Sean Oesch
Anuj Gautam
Scott Ruoti
Background

Motivation and information on mobile autofill frameworks
Motivation

• Prior evaluations focused on desktop managers
• Autofill frameworks are unique to mobile and present their own set of security challenges
  • Could be single point of failure
• We set out to understand three different approaches to an autofill framework on mobile
Autofill on Mobile

• Multiple contexts for autofill
  • Browser
  • Apps
• Multiple approaches to autofill
Contexts for Autofill in Apps

Native UI Elements

WebView

Custom UI Elements
iOS App Extensions

- iOS 8 – 2014
- Popular managers still support – 1Password, Keeper, LastPass
- Older devices – prior iOS 12
iOS AutoFill

- iOS 12 – 2018
- Controls entire autofill process
  - form identification
  - mapping app and domain
  - user interface
  - autofill
Android Autofill Service

- Android 8 (Oreo) 2017 – replaces accessibility service
- Leaves a lot of leeway to individual managers
Approach

Systematic evaluation of autofill properties and testing methodology
Secure Autofill Properties

- Managers should only fill credentials when:
  - P1: Users explicitly authorize operation
  - P2: Credential is securely mapped to web domain or app
  - P3: Credential is only accessible to mapped domain
- Protects against credential scraping and phishing

Autofill dialogue tells user it is safe to fill credentials
Testing

• Strategy
  • Evaluated 14 managers implemented with the autofill frameworks
  • Considered all three properties in all supported contexts
  • Looking for what the framework enforces, what it fails to enforce, and what it prevents managers from enforcing

• Environment
  • iPhone 7 running iOS 13, using Safari for browser tests
  • Genymotion Android emulator
    – Simulated a Google Pixel 2 running Android 9 (Pie)
    – Chrome for browser
Results

*Browser, Native UI Elements, and WebView*
# Autofill in the Browser

<table>
<thead>
<tr>
<th>Framework</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Password AutoFill</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>iOS App Extensions</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Android Autofill Service</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Most secure desktop manager [48]</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Least secure desktop manager [48]</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

- ● Secure behavior
- ○ Partially secure behavior
- ○ Insecure behavior
- Delegate to password manager

User interaction always required
Maps credentials to domains
Won’t fill HTTPS→HTTP
Fills password only on transmission
Won’t fill different action (static)
Won’t fill different action (dynamic)
Won’t fill cross-origin frame

The University of Tennessee, Knoxville
Cross-origin phishing attack
Autofill in Native UI Elements

<table>
<thead>
<tr>
<th>Framework</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Password AutoFill</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>iOS App Extensions</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Android Autofill Service</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

● Secure behavior  ○ Insecure behavior  ⊗ Delegated to password manager
WebView Overview
Violation P2

• Credential should be mapped to website hosted in WebView
• Some managers/frameworks fill the app credentials into any website hosted in WebView
• Users are conditioned to trust autofill dialogues
# WebView Overview

<table>
<thead>
<tr>
<th>Framework</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Password AutoFill</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>iOS App Extensions</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Android Autofill Service</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

- ● Secure behavior
- ○ Insecure behavior
- ✂ Delegated to password manager
Violation P3

• A host app should not be able to access credentials filled into a WebView
• Both iOS and Android allow JS callbacks
Summary & Recommendations

• P1: Users explicitly authorize operation
  • Obeyed by all mobile autofill frameworks in all contexts

• P2: Credential is securely mapped to web domain or app
  • Need a secure bi-directional app-to-domain mapping
  • Should disable autofill in cross-origin iframes

• P3: Credential is only accessible to mapped domain
  • Need secure autofill in WebView and Browser
Questions?

toesch1@vols.utk.edu

@oeschsec