An Experimental Approach to Evaluate the Security of Mobile Autofill Frameworks on iOS and Android

Sean Oesch

Anuj Gautam

Scott Ruoti

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Overview

- How was the research question born?
- First try
- New direction
- Interactive discussion
- Paper brainstorming

The beginning of the journey

How was the research question born?

Generating Research Ideas

- Zhiyun Qian (https://bit.ly/3x9mzHB)
  - Fill in the blank
  - Expansion
  - Hammer and nails
  - Start small and generalize
  - Reproduction of prior work
  - Needs in industry

Where we began - new context, similar approach

- Replicate our work on desktop managers on mobile
  - USENIX 2020 - That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers
- iOS and Android separate papers
- Replicate & expand work of Aonzo et al. on Android - explore similar vulnerabilities on iOS

USENIX Paper Methods

- Generation
  - Corpus 147 million generated passwords
  - Shannon entropy, χ^2 test, zxcvbn, and a recurrent neural net
- Storage
  - Encryption, metadata, master password requirements
- Autofill
  - iframes, form verification, website verification

USENIX Paper Findings & Recommendations

- Generation
  - Filter weak passwords during generation
- Storage
  - Require strong master passwords
- Autofill
  - Require user interaction before filling credential
    - Prevents automatic credential scraping
    - Increases the probability the user can detect attacks
  - Only autofill passwords into secure field
  - Thoroughly vet the fill page

Desktop would benefit from having first-class support for password management in the browsers and/or OS

Aonzo et al. - Credential Mapping on Android

Table 2: Summary of findings for Keeper (K), Dashlane (D), LastPass (LP), 1Password (1P), and Google Smart Lock (GSL).

	K	D	LP	1P	GSL
Secure mapping					✓
One-to-one mapping	✓	✓	✓		✓
Many-to-one mapping		✓			
Crowdsourced mapping			✓		
Heuristic-based mapping	✓	✓	✓		
No mapping				✓	
Q1) Vulnerable?	✓	✓	✓	✓	
Q2) Can co-exist on device?	✓	✓	✓	✓	
Q3) Can co-exist on Play Store?	✓	✓	✓	✓	
Q4) Targeted suggestion?	✓	✓	✓		

First Try

Similar approach, new context

Similar Methodology, New Context

- Looked at ~20 managers on iOS and Android platform based on usage in app store / google play store
- Evaluated generation, storage, and autofill
  - For generation, chose not to repeat check for randomness
  - At this point, autofill was limited to apps and browser (no WebView)
- 2 Papers - 1 for iOS, 1 for Android

Caveat for Android

- Only evaluated generation and autofill on Android
- Expanded Aonzo's work from 5 managers to ~20
  - These results showed that none of the identified mapping vulnerabilities had been addressed in the last several years
  - At the time, I felt this was a very valuable contribution

iOS Paper - PWM Overview

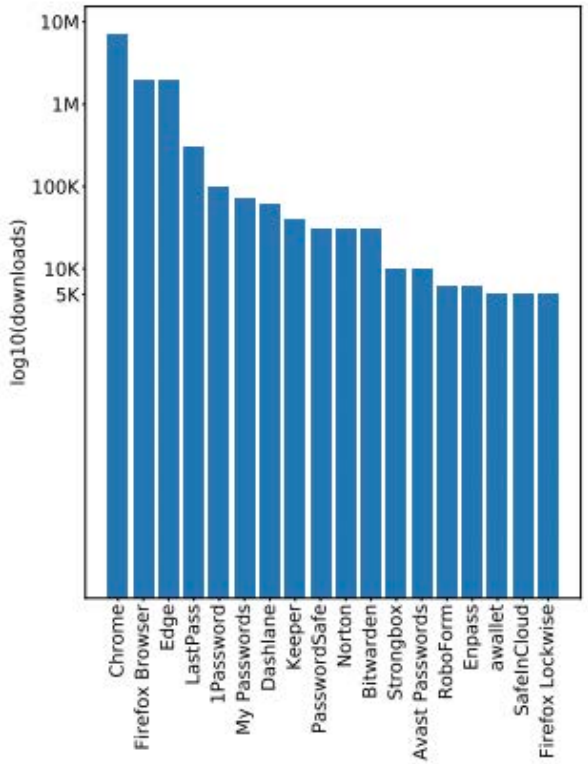


Figure 2: April 2020 Download Estimates

	System	Version tested for §4, §5, §6	Version tested for §7	Supports generation	Supports Password AutoFill	Supports app extensions	Supports in-app browser	Supports copy & paste	Supports Touch ID	Supports PIN	Locks on exit	Has assessment tool	Erase data on auth failure	Open source
Password AutoFill	1Password	7.4.7	7.5.2	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	Avast Passwords	1.15.4	1.15.4	✓	✓	✓		✓	✓	✎	✎			
	Bitwarden	2.3.1	2.4.3	✓	✓	✓		✓	✎		✎			✓
	Dashlane	6.2013.0	6.2023.0	✓	✓			✓	✎	✎	✎	✓		
	Enpass	6.4.2	6.4.5	✓	✓	✓		✓	✓	✎	✓			
	iCloud Keychain	13.3.1	13.3.1	✓	✓			✓	✓	✓				
	Keeper	14.9.1	14.10.1	✓	✓	✓	✓	✓	✓		✓	✓		
	LastPass	4.8.0	4.8.3	✓	✓	✓		✓	✓	✓	✎	✓	✓	
	Lockwise	1.7.3	1.7.3		✓			✓	✓	✓				✓
	Norton	6.8.78	7.0.70	✓	✓	✓	✓	✓	✓	✓	✎			
	RoboForm	8.9.2	8.9.5	✓	✓	✓	✓	✓	✓	✓	✎	✓		
	SafeInCloud	20.0.1	20.1.0	✓	✓			✓	✓	✓	✓		✎	
	StrongBox	1.47.4	1.48.12	✓	✓			✓	✓	✓	✎	✓	✓	✓
Clipboard	aWallet	8.0.2	8.0.2	$				✓	$				✎	
	My Passwords	3.11	3.11	✓				✓	✎				$	
	PasswordSafe	4.2	4.4.1	✓				✓	$		✓			✓
Browser	Chrome	81.0.4044	83.0.4103	✓				✓						
	Firefox	24.1	26.0					✓	✎	✎	✎			✓
	Edge	45.2.16	45.5.0					✓						

✓ Supports or enabled by default ✎ Optionally enabled $ = Available in paid version

Table 1: Overview of password managers

iOS Paper - Storage

	System	File type	Algorithm	MP KDF	KDF Rounds	Requires strong MP	Username	URL	Creation time	Modification time	User's email	User's settings	Icons	Attachments
		Password encryption				Metadata encryption								
AutoFill	1Password	SQLite w/ encrypted cells	AES-256	PBKDF2	100,000	●	●	●	○	○	●	●	●	-
AutoFill	Avast	SQLite w/ encrypted cells	AES-256	PBKDF2	10,240	○	●	●	○	○	●	●	●	-
AutoFill	Bitwarden	Encrypted SQLite file	AES-256	PBKDF2	100,001	●	●	●	●	●	●	●	●	-
AutoFill	Dashlane	Encrypted SQLite file	AES-256	Argon2D	3	●	●	●	●	●	○	●	○	-
AutoFill	Enpass	Custom encrypted file	AES-256	PBKDF2	100,000	○	●	●	●	●	●	◐	●	○
AutoFill	iCloud Keychain	SQLite w/ encrypted values	AES-256	Uses secure enclave			●	●	-	-	-	-	-	-
AutoFill	Keeper	SQLite w/ encrypted cells	AES-256	PBKDF2	100,000	●	●	●	○	○	○	●	●	●
AutoFill	Lastpass	Custom encrypted file	AES-256	PBKDF2	100,100	●	●	●	●	●	●	●	-	-
AutoFill	Lockwise	SQLCipher	AES-256	Uses secure enclave			●	●	●	●	●	●	-	-
AutoFill	Norton	Custom encrypted file	AES-256	PBKDF2	1,000	●	●	●	●	●	●	●	●	-
AutoFill	RoboForm	Custom encrypted file	AES-256	PBKDF2	4,000	○	●	●	●	●	○	●	●	-
AutoFill	SafeInCloud	Encrypted SQLite file	AES-256	PBKDF2	10,000	○	●	●	●	●	●	●	-	-
AutoFill	StrongBox	KeePass or PasswordSafe file	AES-256	Argon2D	2	○	●	●	●	●	●	●	-	◐
Clipboard	aWallet	Custom encrypted file	AES-256	SHA256	1,000	○	●	●	●	●	●	●	-	●
Clipboard	My Passwords	SQLite w/ encrypted cells	AES-256	PBKDF2	1,000	○	●	●	●	●	●	●	-	-
Clipboard	PasswordSafe	Encrypted SQLite file	TwoFish-256	PBKDF2	256	○	●	●	●	●	●	●	-	-
Browser	Chrome	iCloud Keychain		Uses secure enclave			○	○	○	○	○	-	-	-
Browser	Edge	iCloud Keychain		Uses secure enclave			○	○	○	○	○	-	-	-
Browser	Firefox	SQLCipher		Uses secure enclave			●	●	●	●	●	●	-	-

● Secure behavior ◐ Partially secure behavior ○ Insecure behavior - Not stored

MP = Master Password KDF = Key Derivation Function

Table 4: Overview of Password Vault Encryption

iOS Paper - Generation

	System	Default composition	Default length	Supported lengths	Set character minimums	Avoid difficult characters	Make Pronouncable	Generate passphrase	Preserve safe settings	Generation history
Autofill	1Password	ld	24	4–64	✓	✓		✎		
	Avast Passwords	ls	9	4–30					✓	
	Bitwarden	ld	14	5–128	✓	✓				
	Dashlane	ld	8	4–40						
	Enpass	ls	50	4–100	✓	✓	✓	✓		✓
	iCloud Keychain	lsd	20	20				✓	✓	
	Keeper	lsd	20	8–51					✓	
	LastPass	lsd	16	8–64	✓					✓
	Norton	lsd	8	4–64						
	RoboForm	lsd	16	4–511		✓			✎	
	SafeInCloud	lsd	12	8–31			✓	✎		
	StrongBox	lsd	16	6–88	✓	✓	✓	✎		
Clip	aWallet	lsd	8	4–20		✓				
	MyPasswords	ld	10	10						
	PasswordSafe	ld	8	1–50		✎				
B	Chrome	ld	15	15						

✓ Default ✎ Optional

Table 3: Password Generation Features

New Direction

Systematic Analysis of Autofill Frameworks

Secure Autofill Properties

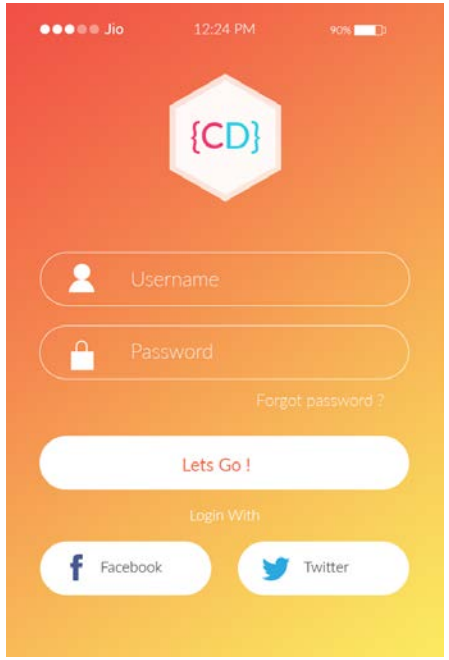
- Managers should only fill credentials when:
  - P1: Users explicitly authorize operation
  - P2: Credential is securely mapped to web domain or app
  - P3: Credential is only accessible to mapped domain
- Protects against credential scraping and phishing

Autofill dialogue tells user it is safe to fill credentials

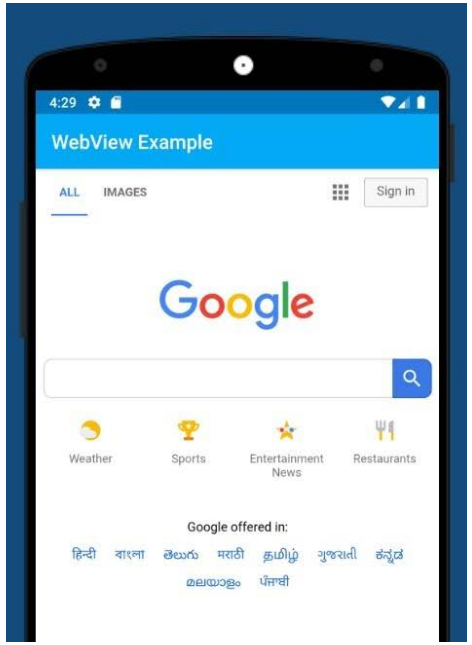
Autofill on Mobile

- Multiple contexts for autofill
  - Browser
  - Apps
- Multiple approaches to autofill

Contexts for Autofill in Apps



Native UI Elements

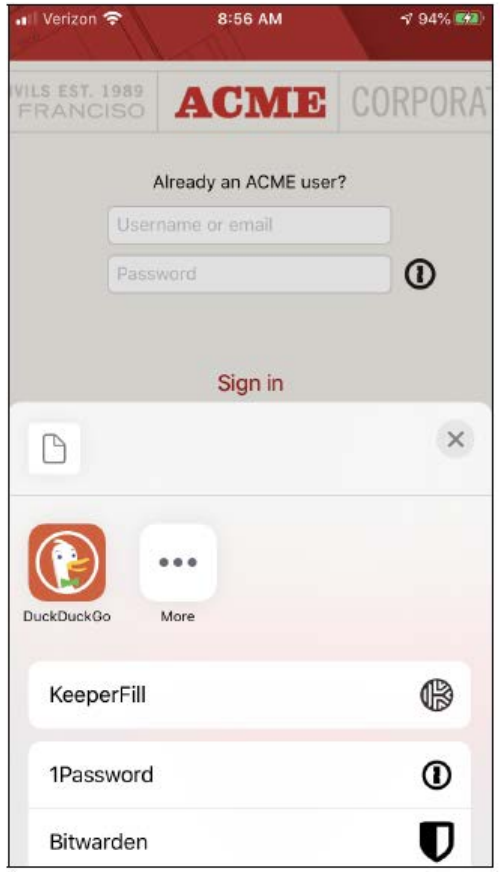


WebView

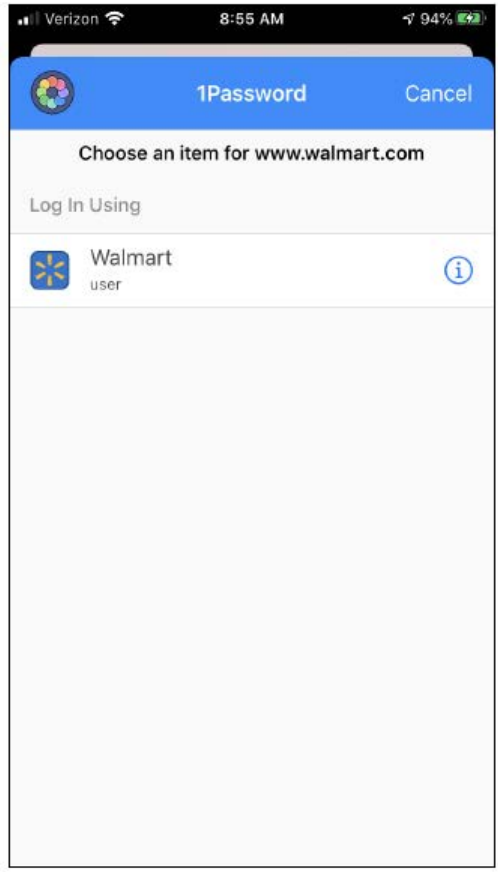
Custom UI Elements

iOS App Extensions

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



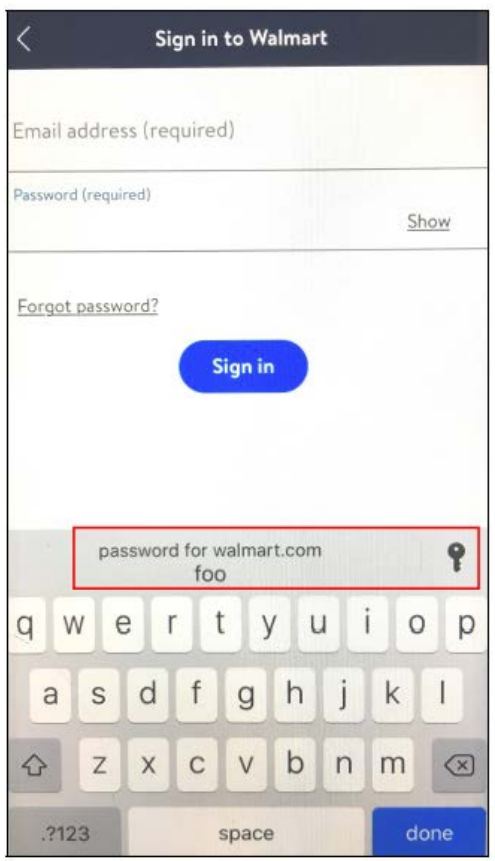
(a) Selecting app extension



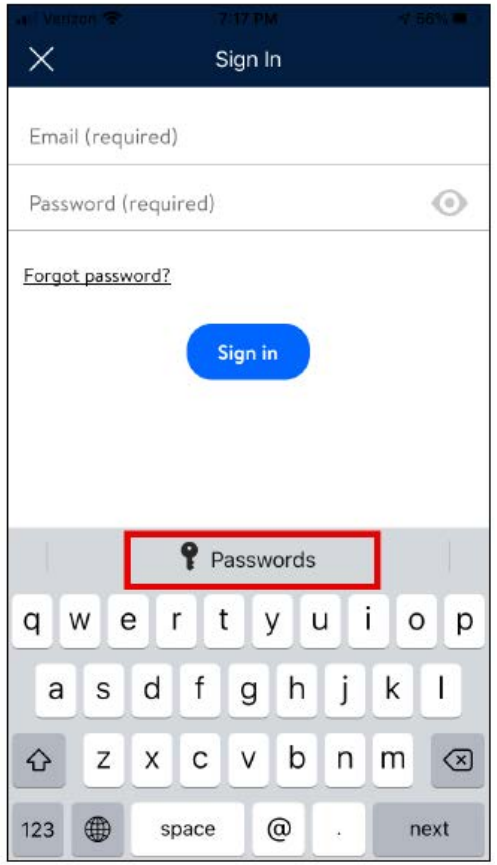
(b) Selecting password

iOS AutoFill

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



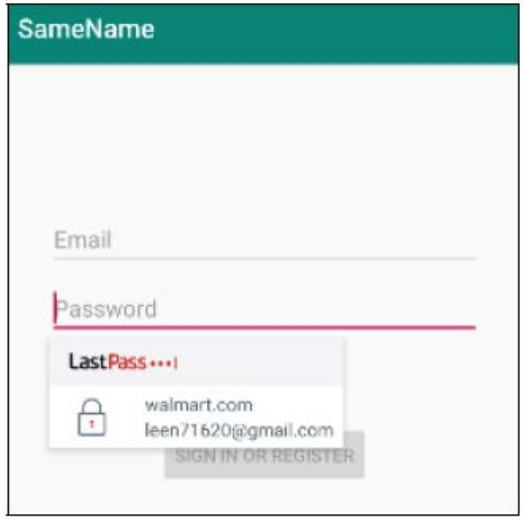
(a) Credential found



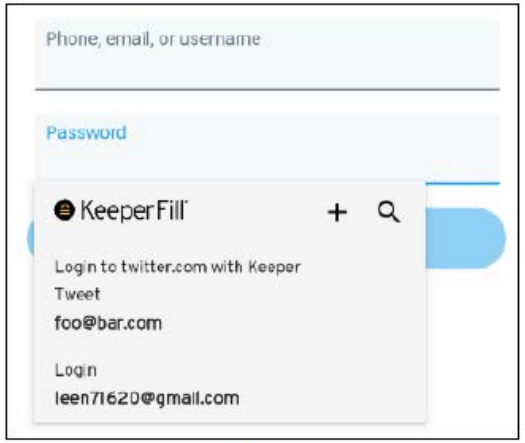
(b) No credential found

Android Autofill Service

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



(a) LastPass



(b) Keeper

Methodology Deep Dive

Testing Autofiill in the Browser, Apps, & WebView

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

Selecting Managers

- Wanted to determine which managers most utilized
- On Android, used download data from Google Play Store
  - Accessible via API
- iOS does not provide detailed information on downloads from App Store
  - used April 2020 SensorTower estimates as a stand-in

Preparing the Devices

- Android
  - Genymotion emulated devices already rooted
  - “Open GApps” to enable the Google Play Store
  - Appmon / Frida to watch network comms
- iOS
  - Because not open source, no ideal emulation platforms
    - Could not install 3rd party apps - only your own
  - Jailbroke the device

Browser Testing Approach

- Improved browser testing website from USENIX paper
  - Based on vulnerabilities identified in Silver et al., Stock and Johns, Li et al.
- NodeJS website
  - Deployed to Heroku and UTK domain
  - UTK domain allowed broken HTTPS and HTTP

Browser Testing Workflow

- Save a password for a heroku domain and a UTK domain
  - UTK domain allowed me to break the cert (Let's Encrypt)
- Run test framework at heroku site
- Test HTTP and broken (invalid cert) HTTPS at UTK domain

Autofill in the Browser

Framework	P1	P2			P3				
	User interaction always required	Maps credentials to domains	Won't fill HTTPS→HTTP	Won't fill HTTPS→bad cert	Fills password only on transmission	Won't fill different `action` (static)	Won't fill different `action` (dynamic)	Won't fill different `method`	Won't fill cross-origin iframe
iOS Password AutoFill	●	●	○	○	○	○	○	○	○
iOS App Extensions	●	●	○	○	○	○	○	○	●
Android Autofill Service	●	●	○	○	○	○	○	○	✎
Most secure desktop manager [48]	●	●	◐	●	○	◐	◐	◐	●
Least secure desktop manager [48]	○	●	●	○	○	●	●	○	○

● Secure behavior ◐ Partially secure behavior
○ Insecure behavior ✎ Delegated to password manager

App Testing Approach

- Android
  - Appmon / Frida for network comms
  - dex2jar to reverse apk and inspect code
  - Blackbox testing via custom apps
- iOS
  - Blackbox testing via custom apps
  - Recall that mapping is always handled by OS

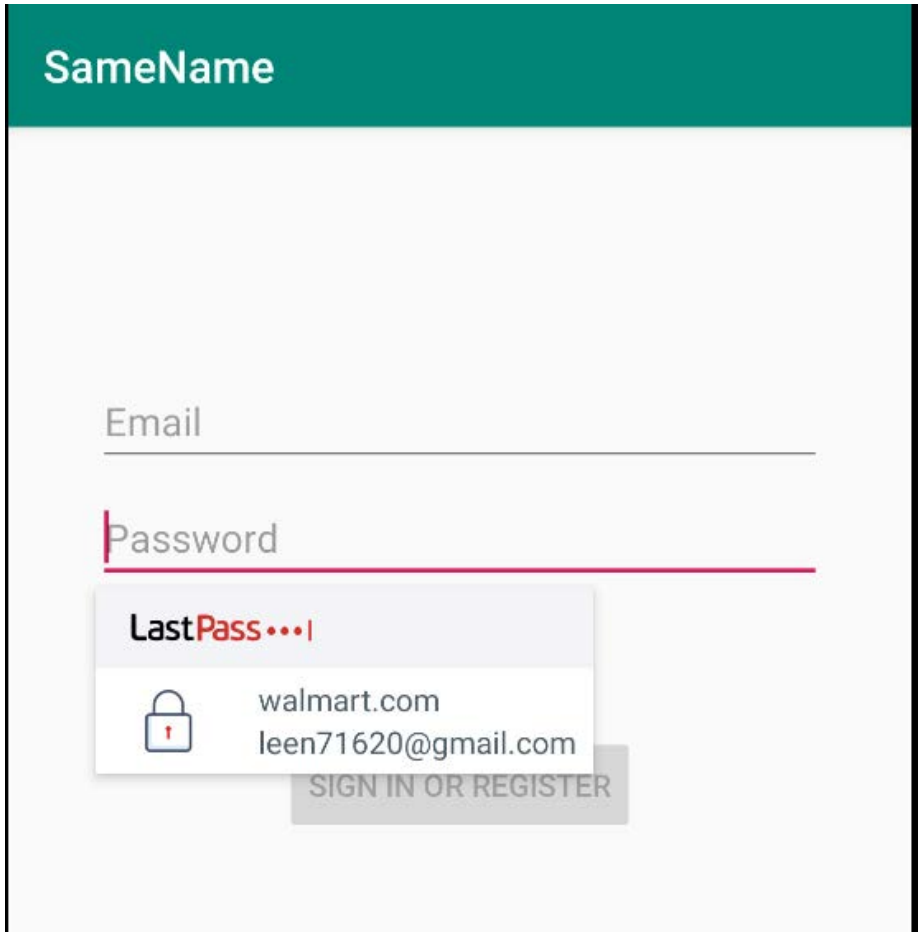
Example Appmon Data

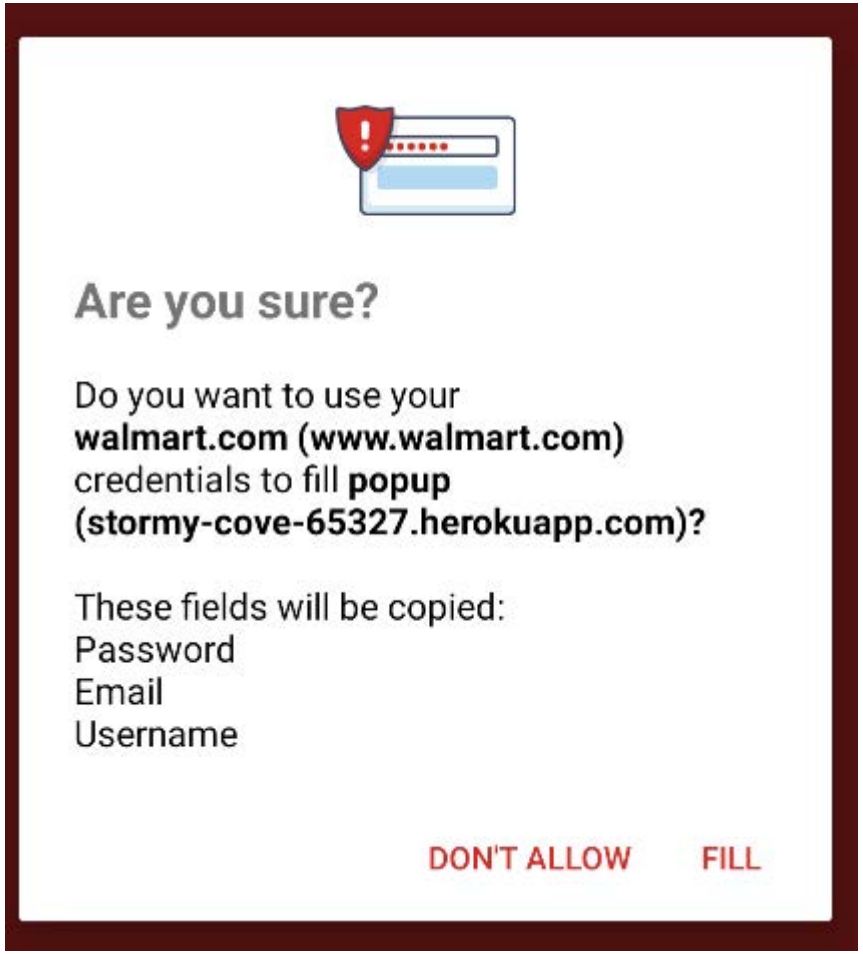
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	getPath	Name: DB Path/data/user/0/com.lastpass.lpandroid/databases/autofill.db	Feb 29 2020 09:07 PM	12626
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	rawQueryWithFactory	Name: SQL StatementSELECT * FROM `AppHash` WHERE `packageName` = 'com.walmart.gotchya' LIMIT 1Name: Selection ArgrumentsName: Edit Tablenull	Feb 29 2020 09:07 PM	12627
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	execSQL	Name: SQL StatementDELETE FROM `AppHash` WHERE `packageName` = 'com.walmart.gotchya' Name: SQL BindArgs	Feb 29 2020 09:07 PM	12652
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	insertWithOnConflict	Name: Table NamepropertiesName: NullColumnHacknullName: Valuesapp_uid=0 hits_count=27 adid=0 params=av=4.11.7.5061&an=LastPass&aid=com.lastpass.lpandroid&aiid=com.android.vending tid=UA-44112561-1 cid=0774f1ea-e155-44d5-b941-157d6e756f81Name: conflictAlgorithm5	Feb 29 2020 09:07 PM	12706
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	compileStatement	Name: SQL StatementSELECT CHANGES()	Feb 29 2020 09:07 PM	12741
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	execSQL	Name: SQL StatementDELETE FROM `WhitelistedVaultEntry` WHERE `packageName` = 'com.walmart.gotchya' Name: SQL BindArgs	Feb 29 2020 09:07 PM	12909
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	insertOrThrow	Name: Table NamemessagesName: NullColumnHacknullName: Valuesentry=[B@9b8db0e type=0	Feb 29 2020 09:07 PM	14043
SQLiteDatabase	android.database.sqlite.SQLiteDatabase	insertWithOnConflict	Name: Table NamemessagesName: NullColumnHacknullName: Valuesentry=[B@9b8db0e type=0Name: conflictAlgorithmnull	Feb 29 2020 09:07 PM	14139
SharedPreferences	android.app.SharedPreferencesImpl	putString	Name: Keyautofill_selected_itemName: Value{"accountId":"278259551208598610","type":0}	Feb 29 2020 09:08 PM	16392
SharedPreferences	android.app.SharedPreferencesImpl$EditorImpl	putLong	Name: Keylast_pause_timeName: Value1583028221795	Feb 29 2020 09:08 PM	16615
SharedPreferences	android.app.SharedPreferencesImpl$EditorImpl	putLong	Name: Keytime_activeName: Value3450	Feb 29 2020 09:08 PM	16616
SharedPreferences	android.app.SharedPreferencesImpl$EditorImpl	remove	Name: KeyLogged in to Site	Feb 29 2020 09:08 PM	16654
SharedPreferences	android.app.SharedPreferencesImpl$EditorImpl	remove	Name: KeyAutofill Item Selected	Feb 29 2020 09:08 PM	16655
SharedPreferences	android.app.SharedPreferencesImpl$EditorImpl	putLong	Name: Keytime_activeName: Value3449	Feb 29 2020 09:08 PM	16722

Miscellaneous Things We Checked

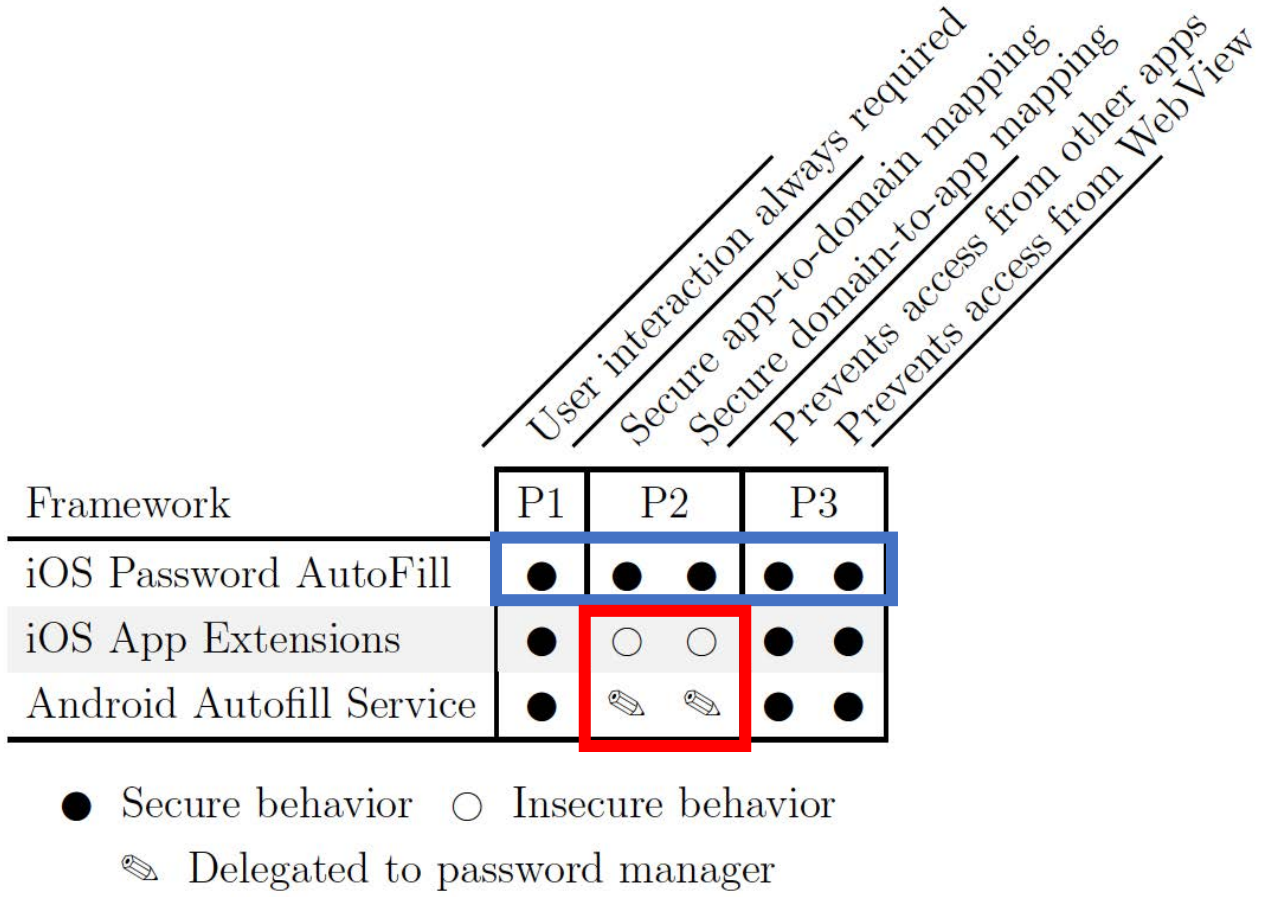
- For every PWM:
  - Permissions required on install
    - Autofill service, observe your actions, manage keyboard, observe text you type, etc.
  - If it clears the clipboard after copying a password
  - Form types it would fill
    - Hint type, invisible form, tiny form
  - Warning rooted device

Example Test App for Mapping





Autofill in Native UI Elements



WebView Overview

Framework	P1	P2		P3					
	User interaction always required	**Maps credentials to domains**	Won't fill HTTPS→HTTP	Won't fill HTTPS→bad cert	**Prevents access from hosting app**	Fills password only on transmission	Won't fill different `action` (static)	Won't fill different `action` (dynamic)	Won't fill different `method`	Won't fill cross-origin iframe
iOS Password AutoFill	●	●	○	○	○	○	○	○	○	○
iOS App Extensions	●	✎	○	○	○	○	○	○	○	●
Android Autofill Service	●	✎	○	○	○	○	○	○	○	✎

● Secure behavior ○ Insecure behavior ✎ Delegated to password manager

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

Framework	P1	P2		P3					
	User interaction always required	**Maps credentials to domains**	Won't fill HTTPS→HTTP	Won't fill HTTPS→bad cert	**Prevents access from hosting app**	Fills password only on transmission	Won't fill different `action` (static)	Won't fill different `action` (dynamic)	Won't fill different `method`	Won't fill cross-origin iframe
iOS Password AutoFill	●	●	○	○	○	○	○	○	○	○
iOS App Extensions	●	✎	○	○	○	○	○	○	○	●
Android Autofill Service	●	✎	○	○	○	○	○	○	○	✎

● 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

Javascript Callback iOS

```
let contentController = WKUserContentController()
contentController.add(self, name: "callbackHandler")

func userContentController(
_ userContentController: WKUserContentController, didReceive message: WKScriptMessage) {
if(message.name == "callbackHandler") {
print("User credentials are \(message.body)")
}
}
```

Listing 1: Callback to communicate with injected JavaScript

```
var username = document.getElementById("email").value;
var password = document.getElementById("password").value;
var credentials = `window.location.hostname:username:password`;
window.webkit.messageHandlers.callbackHandler.postMessage(credentials);
```

Listing 2: Injected JavaScript that steals credentials and sends to malicious app

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 + Paper Discussion

toesch1@vols.utk.edu

@oeschsec