Universität des Saarlandes
MI Fakultät für Mathematik und Informatik
Department of Computer Science

Bachelorthesis

Capabilities as a Solution against Tracking

submitted by

Tim Christmann

on January 01, 1970

Reviewers

Dr. Sven Bugiel

Noah Mauthe

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Saarbrücken, January 01, 1970,

(Tim Christmann)

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, January 01, 1970,

(Tim Christmann)

What is the dedication page?

Yes, you can

Abstract

Trusted Web Activities and Custom Tabs enable Android developers to seamlessly integrate web content into native applications, offering a powerful tool for features such as Single Sign-On and in-app monetization. However, as shown by HyTrack, this integration also introduces severe privacy risks by blurring the boundary between web and app contexts, allowing persistent cross-app tracking through the browser's shared cookie storage.

In this work, we propose a novel mitigation framework that applies capability-based access control to browser cookie handling. Cookie access is encapsulated in fine-grained, identity-bound capabilities, ensuring that only trusted first-party or explicitly authorized third-party web servers-defined by a developer-controlled policy-can access the shared browser state. All other untrusted third-party servers are confined to isolated, in-app cookie jars. This empowers developers to continue leveraging third-party libraries while preventing them from performing unauthorized cross-app tracking. At the same time, essential features such as Single Sign-On and personalized content delivery remain fully functional. Our approach strikes a deliberate balance between privacy and usability, enabling secure web-app integration regarding tracking without compromising user experience or developer effort.

Acknowledgements

Thanks Sven and Noah for . . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, Android applications have increasingly leveraged web content within their interfaces to enhance user experience and streamline features such as authentication and monetization. To enable this, developers often use Custom Tabs (CTs) and Trusted Web Activities (TWAs), technologies that provide seamless, browser-backed web integration while maintaining native-like performance and features. This approach allows web-based functionality like Single Sign-On (SSO), for example login via Facebook or embedded advertising, without forcing users to switch between app and browser.

However, these benefits come at a cost. CTs and TWAs share the browser's cookie storage across all apps, enabling continuity of web sessions—but also opening serious privacy vulnerabilities. Recent research by Wessels et al. introduced HyTrack [1], a novel tracking technique that exploits this shared browser state to persistently track users across different applications and the web, even surviving device changes, cookie clearing, or browser switching. HyTrack works by embedding a third-party library into multiple unrelated apps. Each app, unaware of the library's true purpose, opens a CT or TWA to the same tracking domain. This domain sets a unique identifier in a cookie, stored in the browser's shared cookie jar. When another app using the same library loads content from the same domain, the cookie is sent, enabling the tracker to correlate activity across apps—and even into regular browser use. Due to Android's backup mechanisms, the tracking ID can be restored even after a factory reset, rendering it more persistent than the evercookie [2].

This thesis explores whether capabilities, a fine-grained access control model, can be used to limit or prevent these privacy issues without breaking legitimate use cases of CTs and TWAs. Specifically, we aim to design and evaluate a framework that allows developers to retain the benefits of third-party libraries – such as SSO or monetization – without exposing users to invisible, cross-app tracking. The framework should be simple to integrate, practical in real-world deployments, and minimize interference with established app workflows.

Chapter 2

Related Work

Tracking mechanisms are typically divided into two broad categories: stateful and stateless tracking. Stateless tracking, also known as fingerprinting, infers a user's identity based on a combination of device-specific attributes. Consequently, this method is hard to detect and block, but is also inherently less reliable, as small system changes may alter the fingerprint and disrupt identification.

Instead, stateful tracking relies on storing unique identifiers on the client device, most commonly through cookies or local storage. When a user revisits a site or interacts with embedded third-party content across domains, these identifiers are sent along with requests, allowing persistent recognition.

While straightforward and highly effective, stateful tracking has become increasingly restricted through browser policies (e.g., third-party cookie blocking) and mobile platform changes such as the ability to disable the Google Advertising ID (GAID) on Android. Despite these mitigation efforts, stateful tracking techniques are now re-emerging in new contexts. A notable example is HyTrack [1], which demonstrates a novel cross-app and cross-web tracking technique in the Android ecosystem. HyTrack exploits the shared cookie storage between Custom Tabs and Trusted Web Activities (TWAs) to persistently track users across multiple applications and the browser, even surviving user efforts to reset or sanitize their environments. While HyTrack highlights a serious privacy vulnerability, no concrete mitigation has been proposed that balances privacy with the legitimate need for seamless web integration – such as Single Sign-On or ad delivery – within mobile apps.

This can be seen by taking a closer look at two mitigation strategies discussed by the authors, namely Browser State Partitioning and Forced User Interaction. However, both approaches introduce significant drawbacks. Browser state partitioning would allow each app to use its own cookie storage and hence prevent cross-app tracking. The seamless integration of web content remains intact, as no changes to the UI are necessary, but by completely removing the browsers shared state, benign uses like Single Sign-On (SSO) or ad personalization would be broken.

In contrast, Forced User Interaction does not have this problem, as it allows the browser to use its shared cookie storage. But this highly introduces a significant usability issue, as the user is forced to interact witht the browser every time a web content is loaded, which is not only annoying but also breaks the seamless integration of web content into the app. Furthermmore, this approach hands control and responsibility to the user, which is not ideal from a security perspective, as user might not be aware of the implication of their actions and may inadvertently allow tracking by not interacting with the browser a s required.

Other strategies, such as Limiting CT and TWAs to First-Party Domainsm or introducing browser options to the user to disable CT and TWAs for certain domains boil down to the strategies mentioned above and hence are not workable as remedies against HyTrack.

This work addresses this gap by proposing a capability-based access control framework for Android applications using CTs and TWAs. By wrapping Cookies into fine-grained capability tokens created by the browser according to the app developers policy, the browser decides which cookies are allowed to be stored in the (by default) shared cookie storage and which are stored in the app-specific storage, and thus not accessible to third-party libraries. This ensures that there is no cross-app tracking possible for untrusted thrird-party libraries, as each app stores its own tracking cookie. As the shared cookie storage still exists for domains declared as first-party or trusted by the app developer, legitimate uses of the shared browser state (e.g. SSO) are preserved. Seamless integration of web content remains also untouched, as theres is no need for user interaction or changes to the UI.

Thus, in contrast to prior discussed mitigation strategies, this approach provides developers with a practical and enforceable way to render cross-app tracking infeasible while adhering to the goals stated by the HyTrack authors.

reference again here to goals?

Chapter 3

Methodology: Preventing Cross-App Tracking in CTs and TWAs via Capabilities

Custom Tabs (CTs) and Trusted Web Activities (TWAs) enable a seamless integration between apps and the web by sharing browser state — notably session cookies. While this improves user experience, it also introduces a significant privacy concern: third-party libraries can exploit this shared browser state to track users across apps that embed them. HyTrack, for example, requires only a single shared third-party library used across multiple apps to persistently identify and track users, circumventing typical browser or OS sandboxing.

This solution seeks to preserve the seamless user experience of CTs/TWAs while enforcing cookie isolation across apps to prevent unauthorized cross-app tracking – particularly from embedded third-party libraries.

3.1 Proposed Solution: Capability-Based Cookie Isolation

To address the issue, we propose a developer-defined policy mechanism paired with cryptographically enforced capabilities that dictate how cookies are managed and isolated per app.

Upon app installation, the installer sends a policy crafted by the devoloper to the browser that defines:

- A list of trusted web servers (e.g., the developer's domains or select third parties).

- A set of predefined cookie names expected to be used by those servers.

According to this policy, the browser creates a creates creates the capabilities for each app wrapping important cookie meta data (e.g., app ID, domain, name, and rights) into a secure structure-encoded and signed by the browser similar to JWT-tokes [3].

Consequently, these capabilities serve as authorization tokens for cookie access and are either:

- **Wildcard Capabilities**, that allows the app to set any cookie name and value to be "filled" later or
- **Predefined Capabilities**, which have a fixed cookie name according to the policy and can only be stored in app-specific cookie jars.
- **Ambient Capabilities**, which signals the browser to behave like it would "normally" by storing any received cookie in the browsers cookie jar and including all cookies stored for certain webserver upon next requests. Hence, they do not provide any security and solely exists to keep the program flow from functioning in case the developer does not provide a policy!

check this again!

When the app opens a URL via CT or TWA, it includes a list of capabilities along with the regular intent with the target URL.

The browser parses and verifies each capability by checking the following fields:

- **Signature** to ensure the capability was issued by the browser and has not been tampered with.
- **App ID** to validate the origin, i.e. to ensure the app is authorized to use the capability.
- **Domain** to ensure the correct destination.
- **App Version Number**: to ensure the policy has not changed, i.e. to identify potential policy changes.
- **Rights** to restrict the access scope of the app, such as whether it can request the browser to read cookies. This is necessary to prevent libraries from using the browser to read capability values, which could otherwise be exploited for tracking.
- **Global Jar Flag** to determine whether to use the shared or app-specific cookie jar.

Cookies are included in the request only if the capability passes all checks.

The communication between Browser and Webserver remains unchanged, i.e. the browser sends Request and Set-Cookie headers as usual and the webserver responds with an (customized) Response and possibly new Cookies.

The browser them matches received cookies to capabilities:

- If the cookie name is predefined, only the value is updated in the capability.
- If it's new and a wildcard capability exists, the browser creates a new capability by copying the wildcard capability and setting the name and value accordingly.

Additionally, if the global jar flag is set, the cookie is stored in the browser's cookie jar and the capability is sent back to the app. Otherwise, the capability is directly returned to the app and stored in its private app jar.

If validation fails at any point, the browser ensures a failsafe default by creating a fresh state before sending the request to the webserver, making sure that no cookies are sent along with it.

3.2 Benefits

Besides eliminating the possibility of cross-app tracking and the by hytrack postulated goals ,I see the following additional benefits this approach offers:

reference to chap:eval where statet?

B1) **Fine-Grained Control**: Developers can specify which cookies are shared and which are isolated, allowing for a more tailored approach to privacy.

B2) **No Browser State for Apps**: The browser does not need to remember app capabilities — the app holds and re-sends them with each intent.

B3) **No Ambient Authority**: By avoiding the need for the browser to store app states, we minimize the attack surface and potential vulnerabilities.

B4) **No Third-Party Code Changes**: The webserver does not need to be aware of the capabilities or make any changes to its code, as the browser handles the capability management.

B5) **Backwards Compatibility**: When the developer does not implement its own policy, the policy will default to allowing any webserver using the shared browser state, consequently behaving like the state of the art cookie management. This is done by allowing wildcards for the domain name in addition to the cookie name and value. Ofcourse, for the cookie to be stored in the shared cookie jar, the global jar flag must also be set by default.

rethink this concept again! Do i even need this? in what other sense could backwards comapatibility exist?

Maybe there exist a trusted authority that can declare a domain as trusted e.g. Facebook for SSO etc. for default policy? But also not optimal I guess

3.3 Hypotheses

My evaluation is guided by the following hypotheses:

H1) Sending capabilities with each intent is sufficient to ensure that the browser can validate and manage cookies effectively.

H2) The installer can send the developer-defined policy to the browser and can return it to the app.

H3) The approach adheres to each goal of HytTrack (stated in 4).

own section label for the goals?

H4) The approach fully eliminates cross-app tracking.

is this a Hypothesis?

H5) Backwards compatibility can be implemented via "Ambient Capabilities".

H6) Empty Requests can be used to provide a failsafe default in cases of invalid capabilites.

H7) There are no code changes of webservers required.

3.4 Components

In summary, the proposed solution requires the following components to be changed or added:

- **Installer** to send the developer-defined policy to the browser upon app installation.
- **Browser** to function as Policy Enforcement Point. For this, new functionality for creating capabilities according to the policy and validating them is necessary. In addition, it needs to be capable of storing a secret key for signing capabilities and verifying the signature. Ofcourse, decision making according to the capabilities context is unavoidable, i.e. where to store the capability, or the pair of name and value, either in the app-specific or the global cookie jar.
- **App** to send capabilities with each intent and to store the capabilities returned by the browser. It must also support functionality to send specific capabilities to the browser.

Chapter 4

Evaluation

To assess the effectiveness of our proposed mitigation strategy, I adopt the three primary goals identified by the authors of HyTrack as essential for any viable defense:

1) **Support all features of the web platform:** The solution must allow applications to display fully functional web content, including support for cookies, JavaScript, and modern APIs.

2) **Preserve seamless integration:** The user experience must remain uninterrupted. This includes avoiding obtrusive permission dialogs and maintaining smooth transitions between native and web content.

3) **Enable controlled access to shared browser state:** While isolation is required to prevent cross-app tracking, legitimate scenarios such as Single Sign-On (SSO) should continue to work within the context of a single application.

These criteria reflect the fact that HyTrack exploits standard Android behavior – specifically, the shared browser state exposed through Custom Tabs and Trusted Web Activities – rather than relying on unauthorized access or system vulnerabilities. Therefore, naive approaches like disabling shared cookies entirely would break common use cases and are not acceptable.

To validate these hypotheses, I will build on the open-sourced measurement tooling and proof of concept applications provided by the authors of HyTrack. Specifically, I plan to:

- Replicate the original HyTrack experiments under controlled conditions using two unrelated Android apps that embed the tracking library (similar to the HyTrack demo).

- Instrument network traffic (e.g., using mitmproxy or Frida) to observe the in capabilities wrapped cookies and their interactions with the browser.
- Apply the mitigation framework and compare observed behavior against the baseline.

keep this?

I will collect and analyze the following metrics:

- Number of Capabilities created and used by the browser.
- The latency of framework operations, such as creating and validating Capabilities, in comparison with the baseline.

In doing so, I aim to demonstrate that the proposed solution effectively blocks HyTrack's cross-app tracking channel while preserving compatibility with existing web features and maintaining seamless user experience. Additionally, I will show that this strategy can be used by developers to control cookie transmission in a fine-grained and policy-driven manner to enable safer and more transparent integration of third-party web content.

Chapter 5

Schedule

I plan to complete the project in the following phases:

- **Phase 1 (Week 1):** *Background Research and Literature Review*
  Study existing work on web and cross-app tracking, especially the HyTrack paper. Familiarize with Android's Custom Tabs (CTs), Trusted Web Activities (TWAs), and Chromium's cookie storage mechanisms. Reproduce the HyTrack proof of concept to establish a baseline for evaluation.
- **Phase 2 (Weeks 2-4):** *Framework Design and Initial Implementation*
  Design the capability-based access control framework. Define requirements, threat model, and integration points. Begin implementation of the framework, touching on the Installer, App and Chromium browser components.

  similar threat model to HyTrack (not consider malicious dev and thus false policies)
- **Phase 4 (Weeks 5):** *Testing*
  Thoroughly test the framework to ensure it works as intended and meets the requirements.

  write requirements here again?
- **Phase 3 (Week 6):** *Evaluation and Experimentation*
  Set up experimental infrastructure using HyTrack's open-source tooling. Measure the effectiveness of the proposed solution by replicating tracking scenarios and comparing cookie behaviors across test cases. Conduct experiments to evaluate the framework's performance, feasibility of integration and backwards compatibility.

5.1 Risks, Impact and Mitigation

At worst, the capability-based approach may not effectively prevent cross-app tracking in practice and indeed a complete redesign of the Custom Tab and Trusted Web Activity

API is needed, as suggested by the authors of HyTrack. If this occurs, the thesis will pivot to a critical analysis of why the capability model falls short in this context. The focus will shift towards identifying structural barriers in the Android platform and recommending future improvements to make such defenses feasible.

At much lower impact, the implementation may not be feasible within the given timeframe or due to unforseen technical challenges. In this case, development will focus on a minimal viable proof-of-concept that demonstrates core functionality and missing features will be discussed as future work.

what else can go wrong?

⋮

Chapter 6

Success Criteria

The following criteria define the scope of this thesis and provide measurable goals to assess its success.

6.1 Must-have criteria

- The solution must effectively prevent or significantly limit cross-app tracking via the shared browser state, while preserving the core functionality of Custom Tabs and Trusted Web Activities (e.g., Single Sign-On).
- It must adhere to the three primary goals outlined by HyTrack: (1) support all features of the Web platform, (2) do not break the seamless integration, and (3) make shared state available to the Custom Tab or Trusted Web Activity.
- A working proof-of-concept must be implemented to demonstrate the feasibility of the approach and to reproduce and compare against the HyTrack attack methodology.
- The thesis must provide a thorough discussion of how capability-based access control contributes to mitigating the identified tracking risks, including analysis of potential trade-offs (e.g., small form of ambient authority as based on identity).

6.2 May-have criteria

- Provide an analysis of the solution's impact on attributes like performance, usability, integration effort for developers, and potential security limitations.

- Evaluate the framework across multiple Chromium-based browsers to assess generalizability.
- Investigate whether enhancements to the Digital Asset Links (DAL) mechanism could further strengthen the solution or propose a good alternative.

6.3 Must-not-have criteria

- The thesis will not produce a production-ready implementationk, as the focus is on proof-of-concept and feasibility.
- The solution is not intended to prevent all forms of cross-app tracking, particularly not against malicious developers who intentionally bypass protections. The goal is to protect well-meaning developers from inadvertently integrating tracking libraries.
- It will not attempt to mitigate stateless tracking methods such as fingerprinting, which are outside the scope of shared state via cookies.

Bibliography

[1] M. Wessels, S. Koch, J. Drescher, L. Bettels, D. Klein, and M. Johns, “Hytrack: Resurrectable and persistent tracking across android apps and the web,” in *34th USENIX Security Symposium (USENIX Security 25)*. Seattle, WA: USENIX Association, Aug. 2025.

[2] S. Kamkar, “evercookie - virtually irrevocable persistent cookies,” https://samy.pl/evercookie/, 2010, accessed: 2025-05-25.

[3] M. B. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” RFC 7519, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7519

Additional Something

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Notes