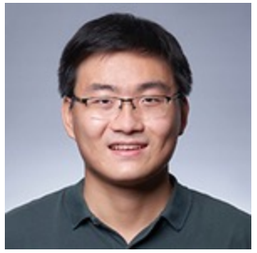
34TH USENIX
SECURITY SYMPOSIUM

Universal Cross-app Attacks: Exploiting and Securing OAuth 2.0 in Integration Platforms

Kaixuan Luo[1], Xianbo Wang[1], Adonis Fung[2], Wing Cheong Lau[1], Julien Lecomte[2]

1 The Chinese University of Hong Kong, 2 Samsung Research America

About us



Kaixuan Luo*
PhD Candidate
kaixuan@ie.cuhk.edu.hk

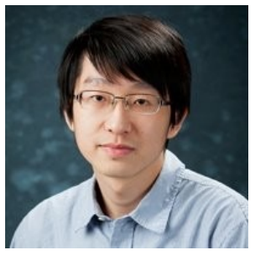
Xianbo Wang
PhD Candidate
xianbo@ie.cuhk.edu.hk

Wing Cheong Lau
Professor

MobiTeC
Mobile Technologies Centre

香港中文大學
The Chinese University of Hong Kong

* Part of the work done while interning at Samsung

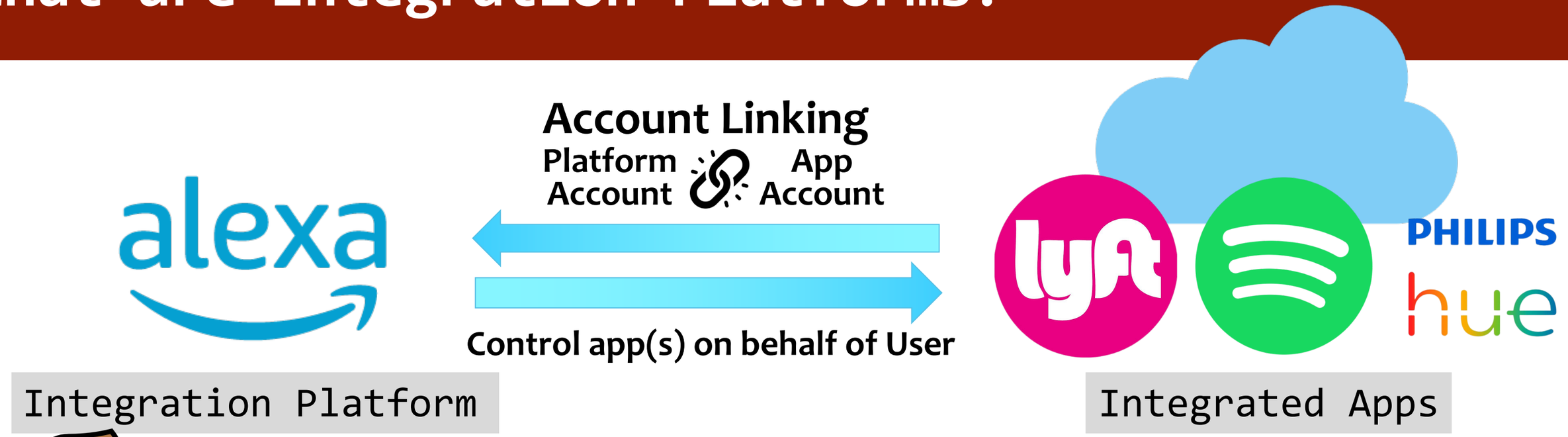


Adonis Fung
Director of Engineering, Security
Samsung Research America

Julien Lecomte
Senior Director of Engineering
Samsung Research America

Samsung Research America

What are Integration Platforms?



- **Integration Platform** Connects & Aggregates functionalities of diverse apps/ services/ devices
- **Account Linking** Links the end-user's App accounts to Integration Platform account
- **OAuth 2.0** is the de facto standard protocol to achieve Account Linking

What are Integration Platforms?

By Usage Scenario

Workflow Automation Platforms

Microsoft Power Automate

IFTTT

Virtual Voice Assistants

Google Assistant

alexa

Smart Homes

Google Home

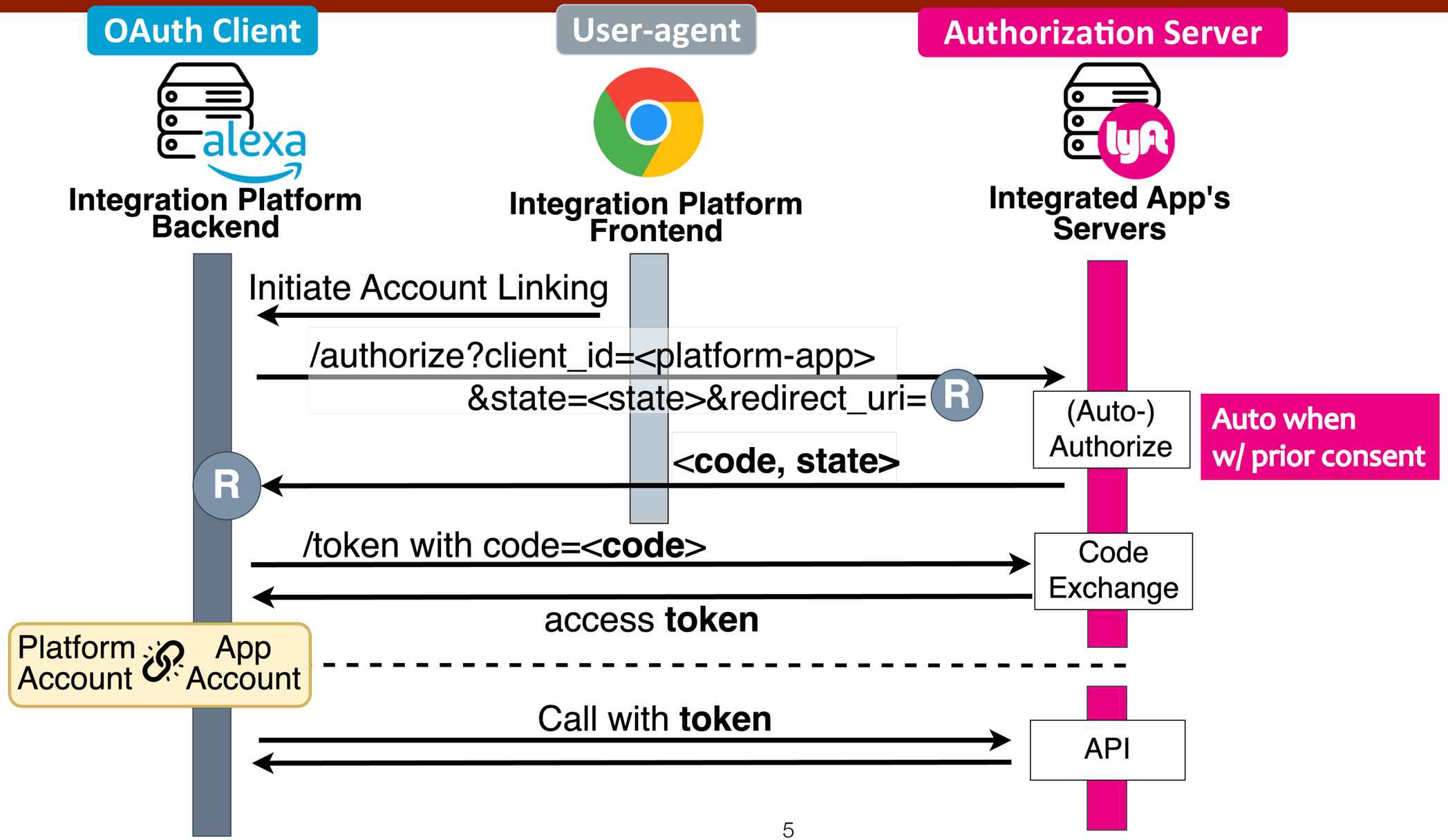
LLM Platforms with Plugins

By Development Approach

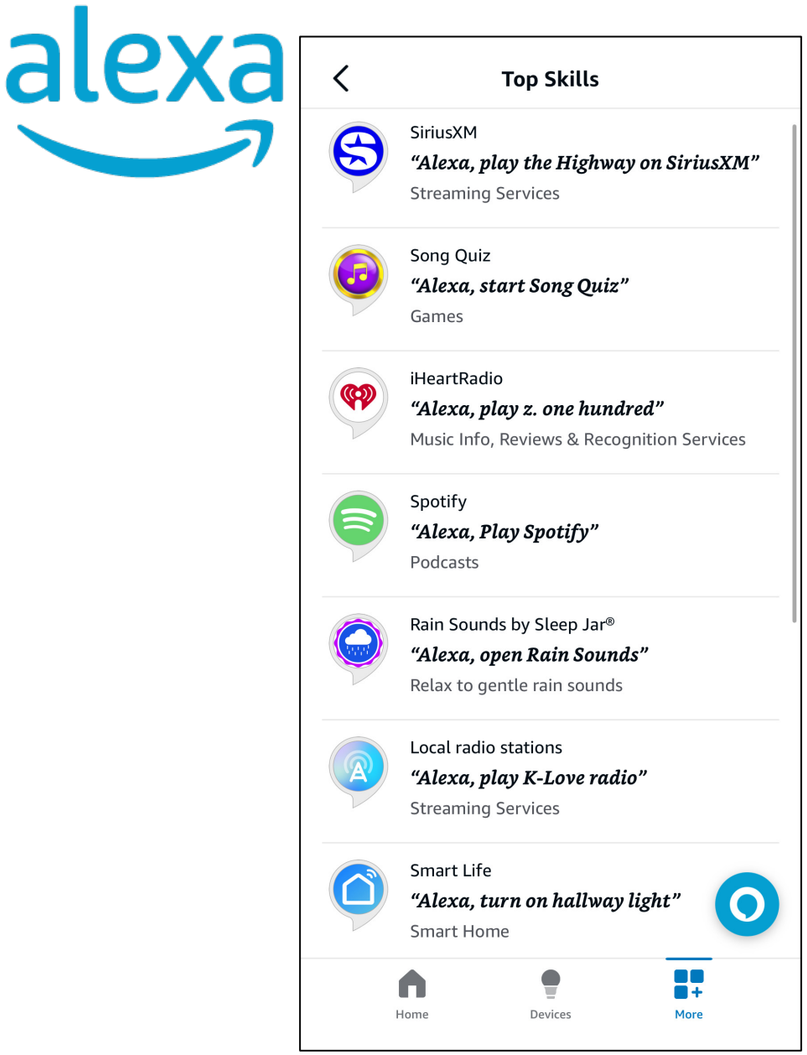
Trigger-action Platforms

Low-Code/No-Code Platforms

Traditional OAuth
OAuth 2.0 Authorization Code Grant

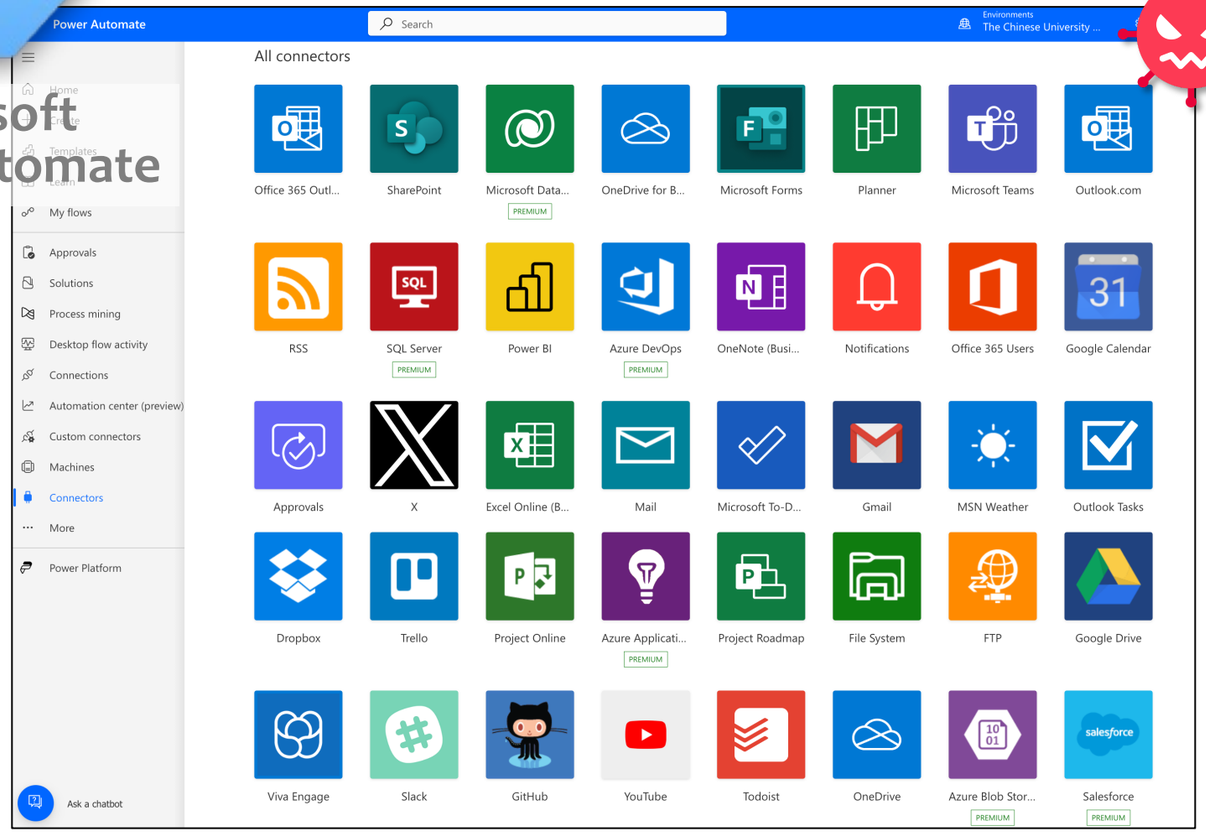


Open Ecosystem: Marketplace Design

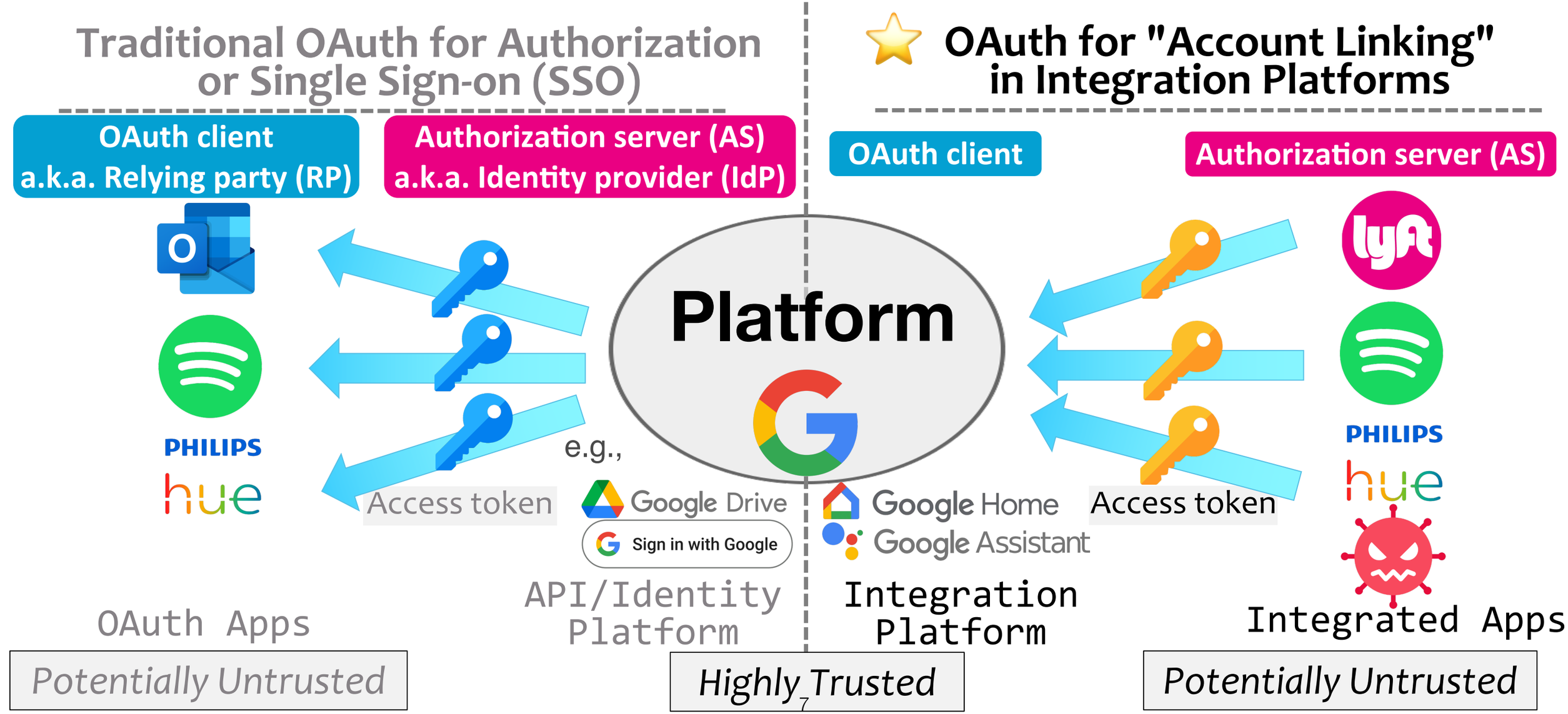


Microsoft
Power Automate

Anyone can publish an app



Paradigm Shift: OAuth Role Reversal



When OAuth-based Account Linking Goes Wrong

			Consequence
User		Same User	LGTM !
Attacker		Victim	Account Takeovers → Cross-app OAuth Account Takeover (COAT)
Victim		Attacker	Privacy Leakage → Cross-app OAuth Request Forgery (CORF)

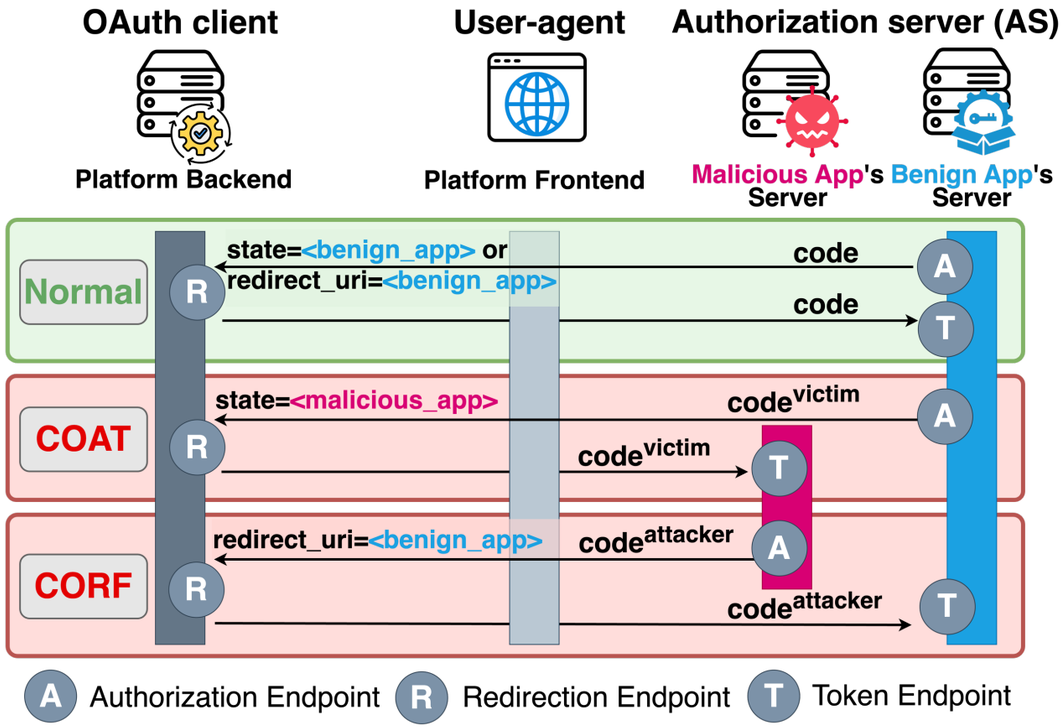
Overview: Cross-app OAuth Attacks

Attack Scenario:

- Attacker sets up malicious app
- Victim establishes OAuth with it

Attacks:

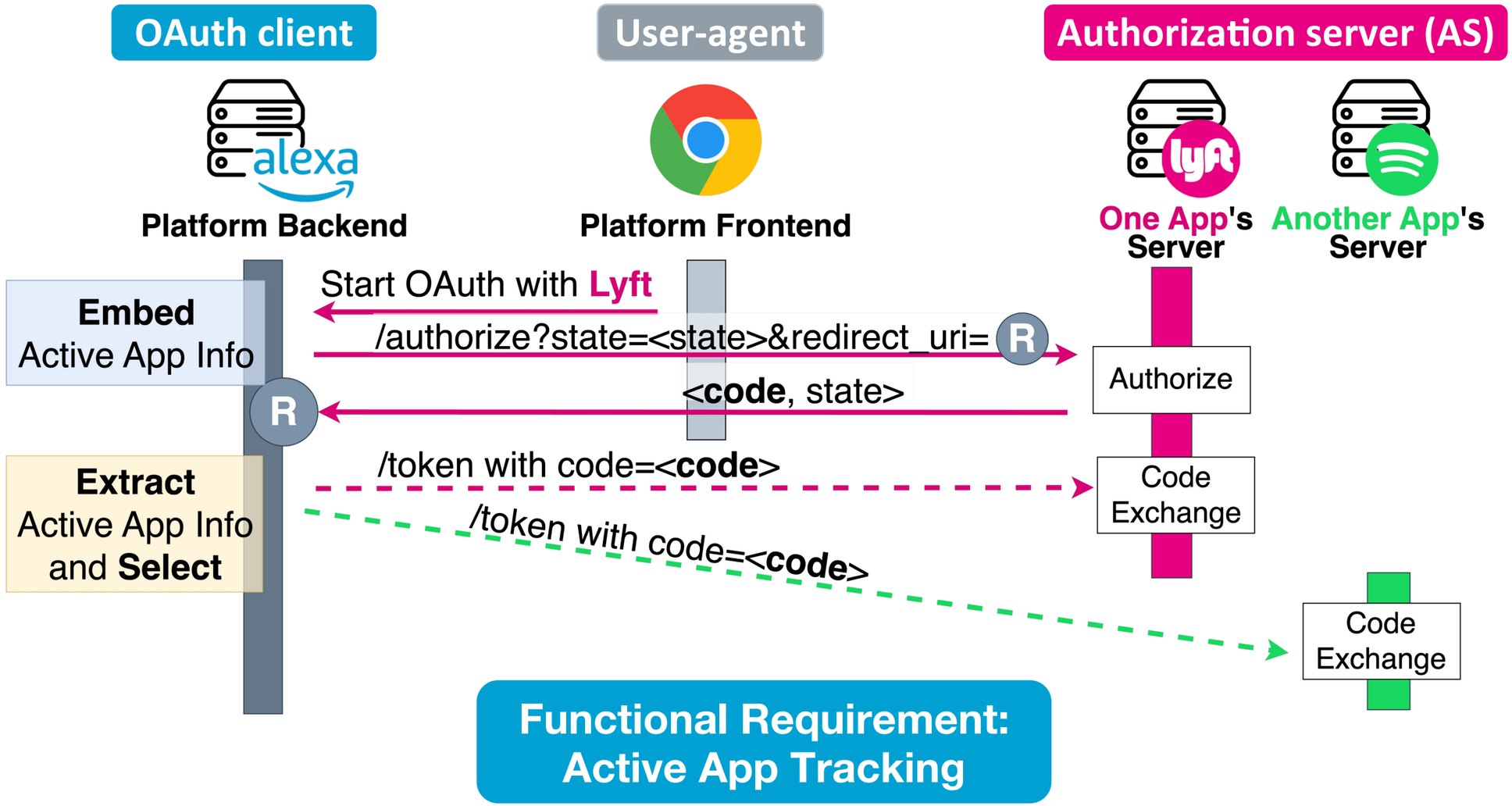
Cross-app OAuth Account Takeover (COAT)	Leak victim's *auth code*
Cross-app OAuth Request Forgery (CORF)	Inject attacker's *auth code*



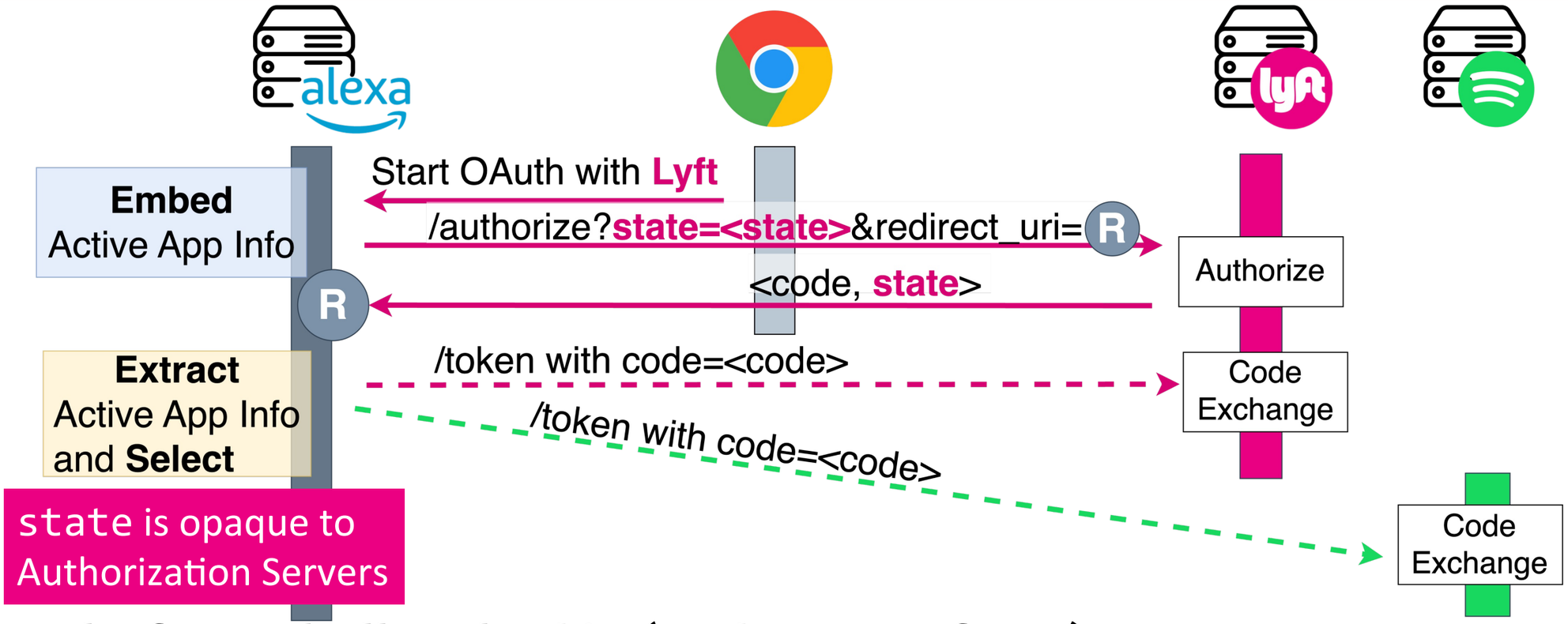
Defense:

- Enforce “App that starts” = “App that completes” with per-app redirect_uri

Challenge:
Supporting Multiple Integrated Apps



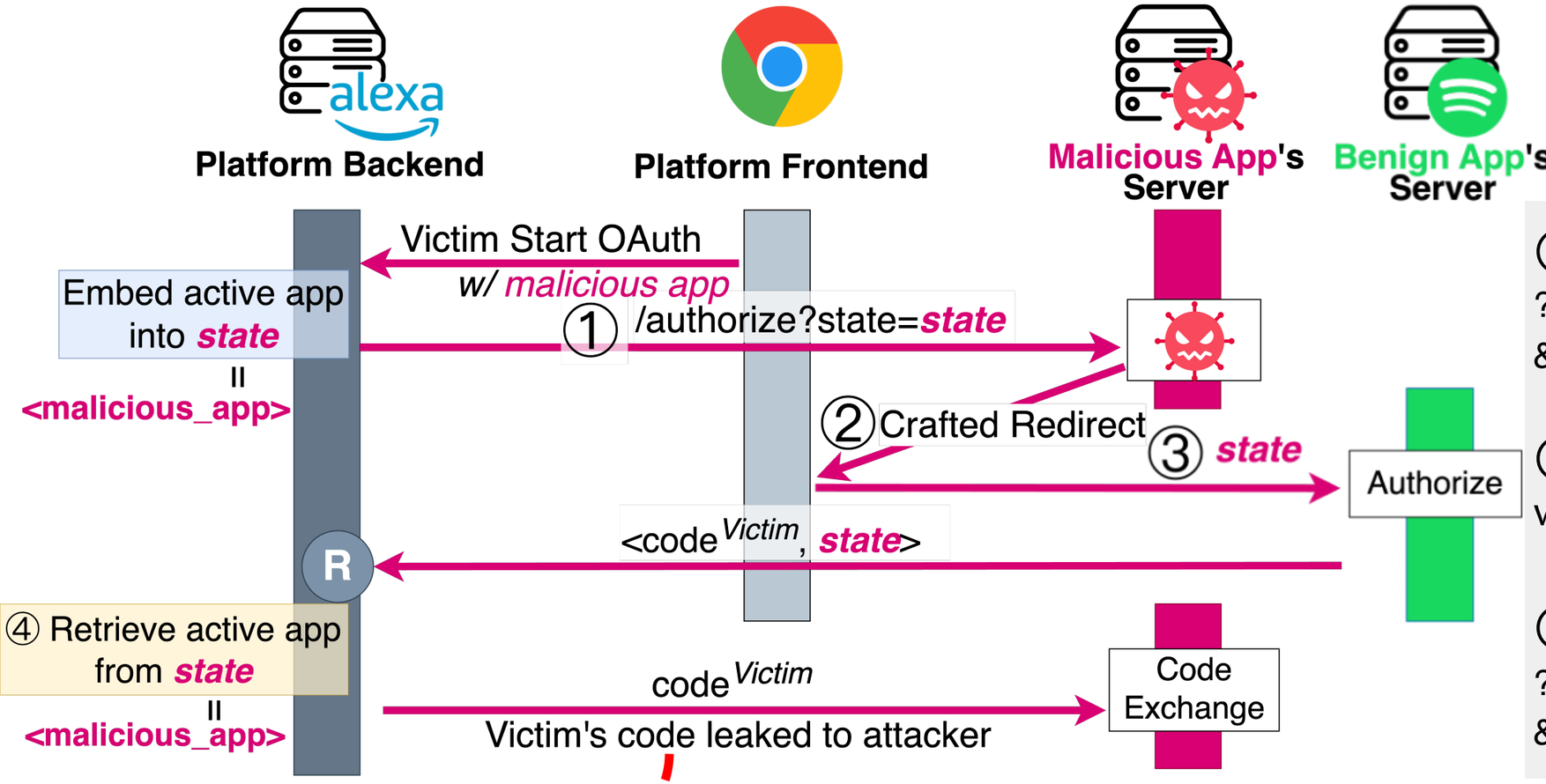
Common (but Flawed) Designs for Active App Tracking



Platform shall embed in (and extract from):

- **state=eyJxxx.yyy.zzz**
  {"app_id": <lyft>,
  …}
  or platform's internal state
  (e.g., session, frontend-managed state)

Attack #1:
Cross-app OAuth Account Takeover (COAT)



① **GET** malicious.com/authorize
?client_id+redirect_uri=<malicious_app>
&state=<malicious_app>

② Redirect to benign app, while keeping malicious app's state

③ **GET** benign.com/authorize
?client_id+redirect_uri=<benign_app>
&state=<malicious_app>

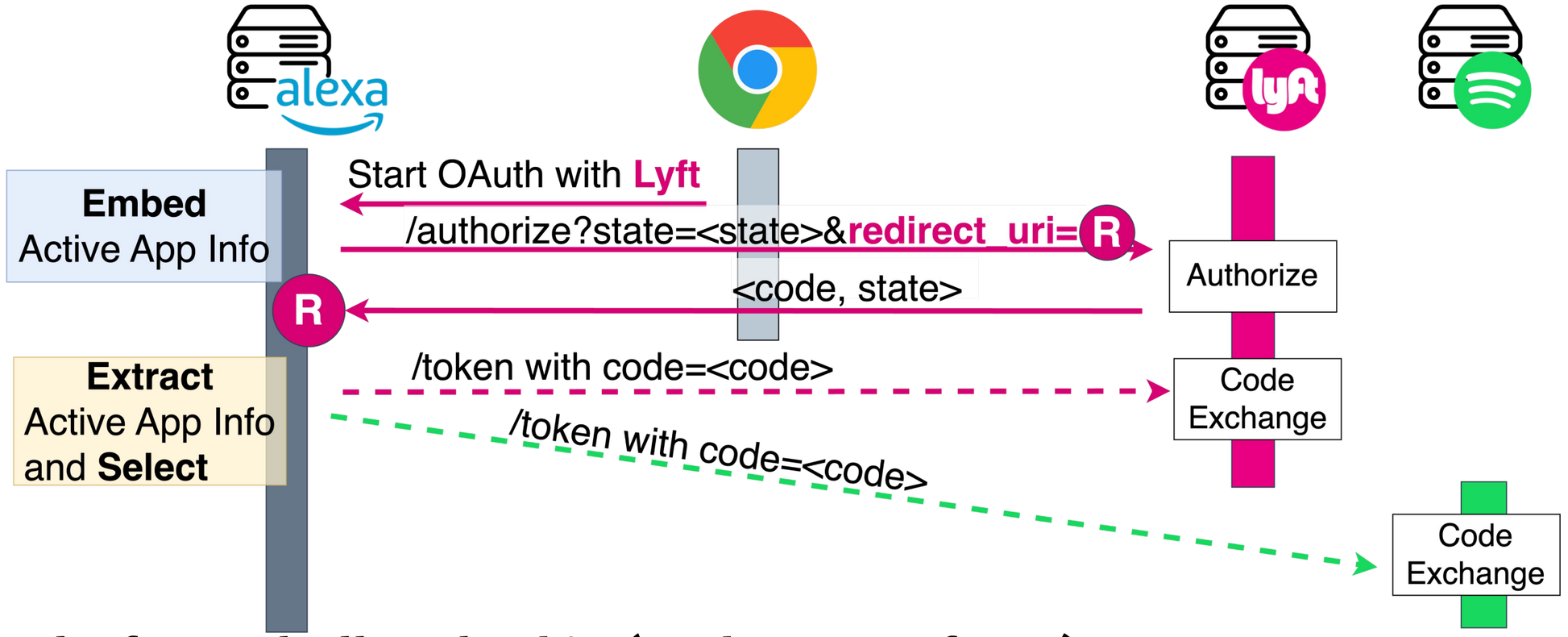
④ **Flaw:** Track active app solely by `state`

Unauthorized Access

Attacker's Platform Account 🔗 **Victim's App Account**

Authorization Code Injection
per Sec 4.5 of OAuth 2.0 Security BCP

Common (but Flawed) Designs for Active App Tracking



Platform shall embed in (and extract from):

- **state=eyJxxx.yyy.zzz**
  {"app_id": <lyft>,
  …}
  or platform's internal state
  (e.g., session, frontend-managed state)

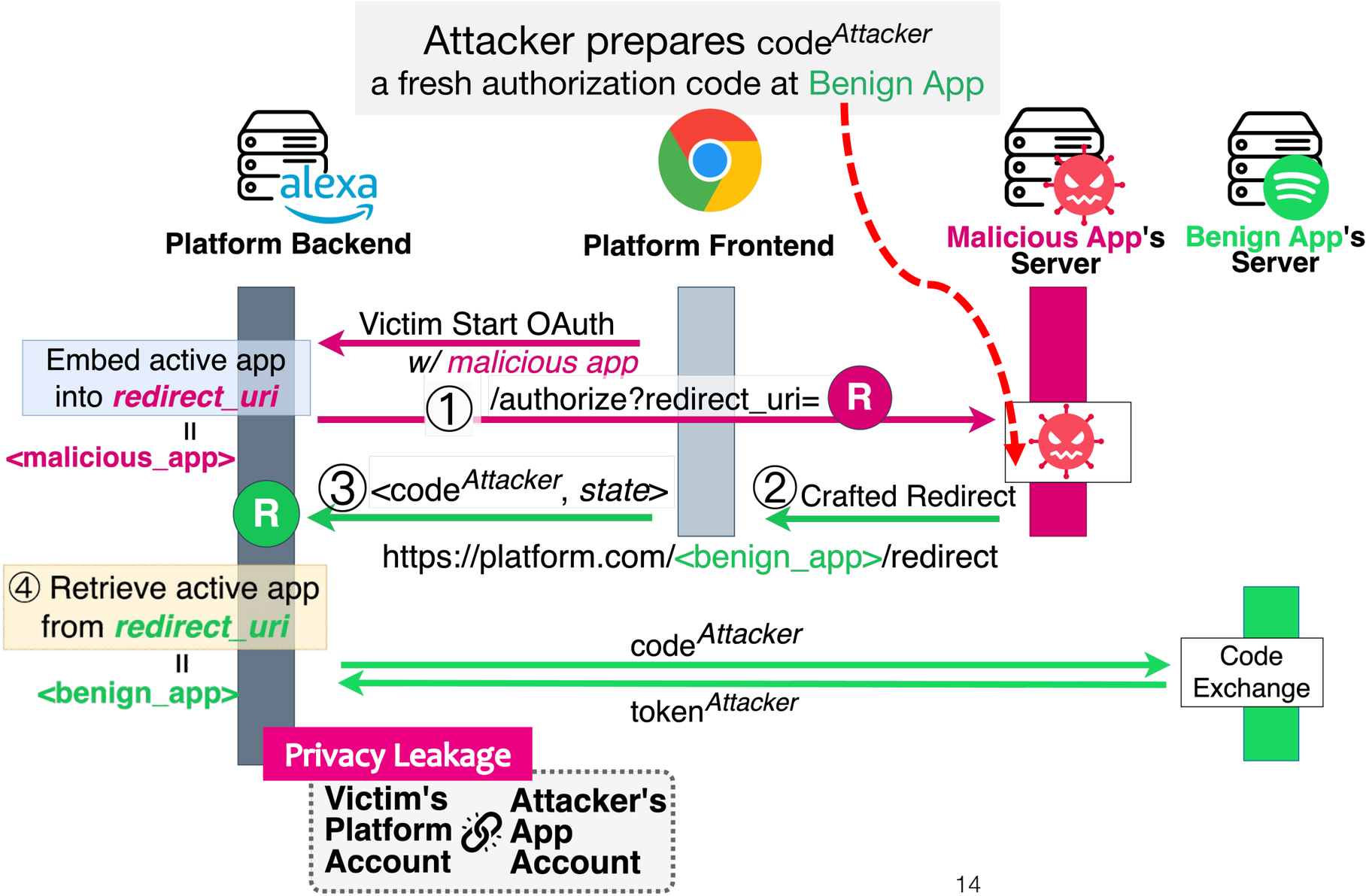
OR

- **redirect_uri:**
  platform.com/<lyft>/redirect

redirect_uri has weak integrity

app_id

Attack #2:
Cross-app OAuth Request Forgery (CORF)



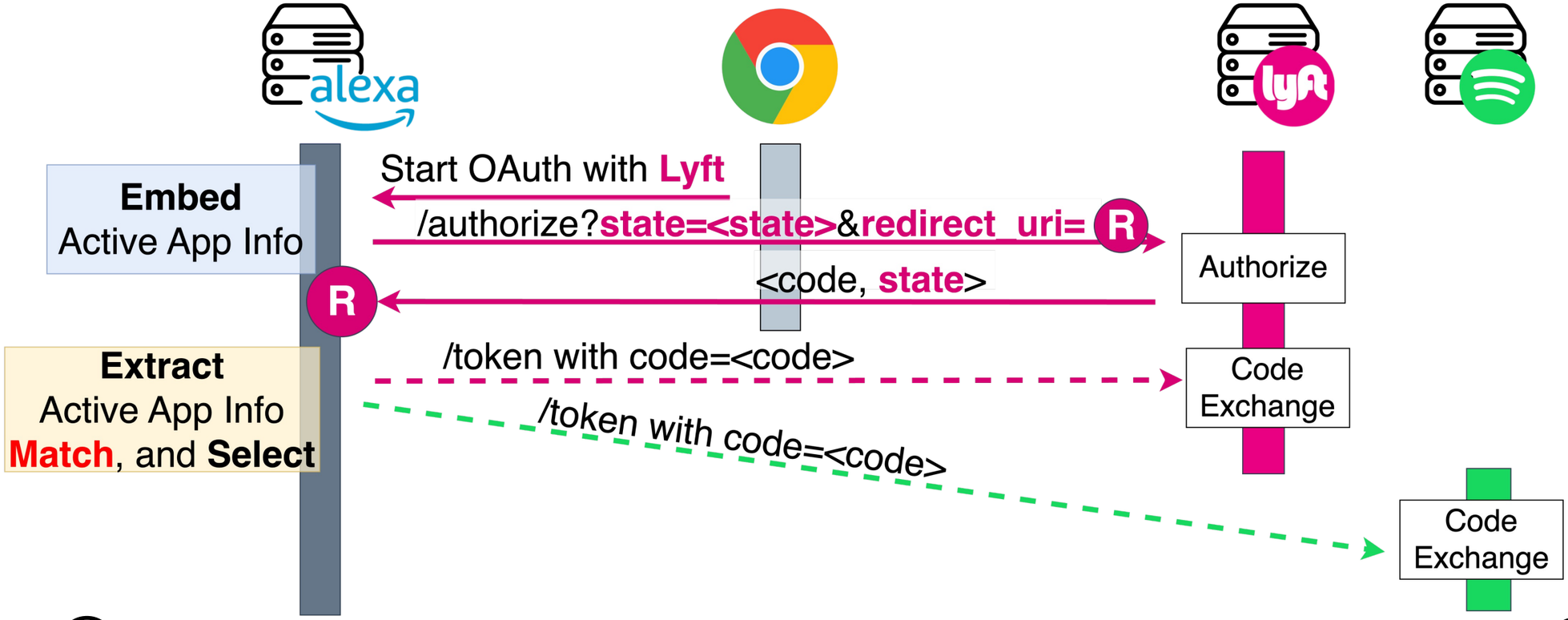
① **GET** malicious.com/authorize ?client_id=<malicious_app> &redirect_uri=platform.com/<malicious_app>/redirect&state=<state>

② Redirect to benign app's R, Injecting the prepared code

③ **GET** platform.com/<benign_app>/redirect ?code=<attacker>&state=<state>

④ **Flaw:** Track active app solely by distinct redirect_uri

Defense for both COAT and CORF:
Per-app redirect_uri + consistency check at redirection EP



①Shall embed a unique app ID in (and extract from) BOTH:

- **state=eyJxxx.yyy.zzz**
  {"app_id": <lyft>,
  …}
  or platform's internal state
  (e.g., session, frontend-managed state)

AND

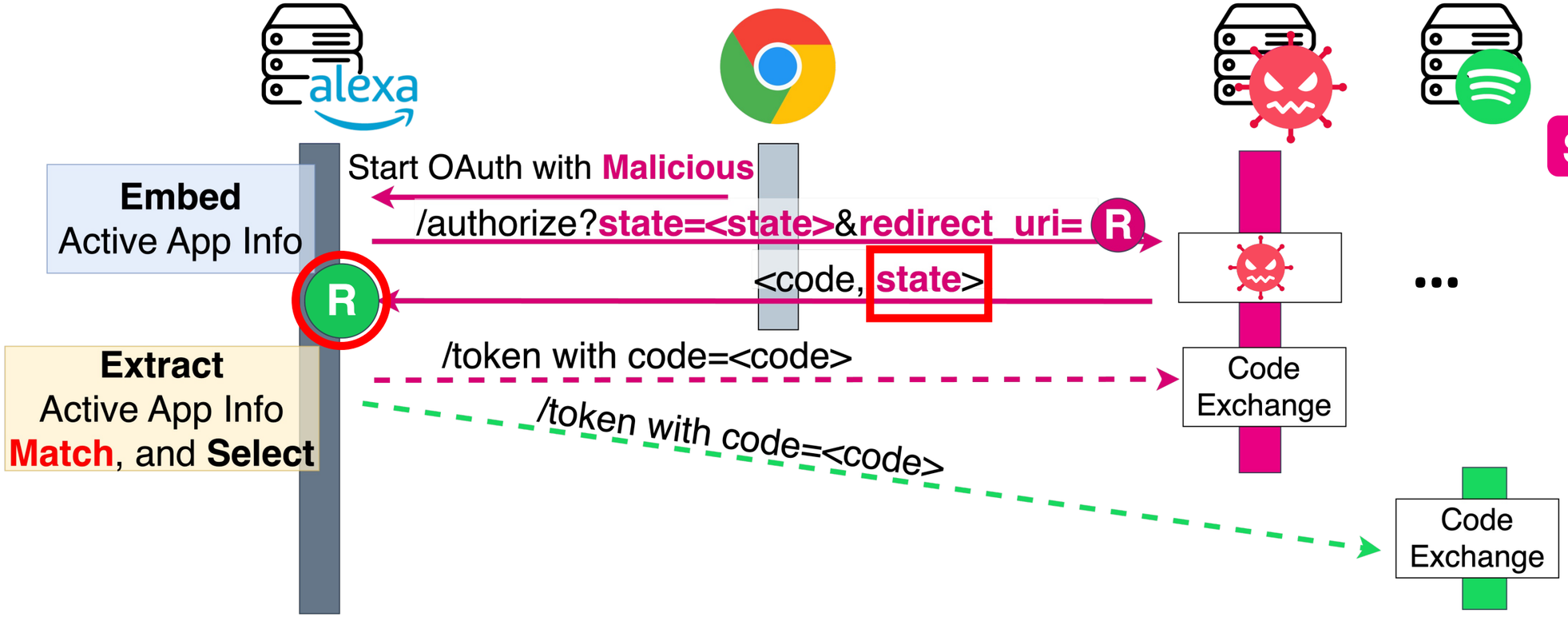
- **redirect_uri:**
  platform.com/<lyft>/redirect

app_id

②Enforce Matching at R

Security Requirement

Defense for both COAT and CORF:
Why does it work?



Security Requirement

"App that starts"
= "App that completes"

- **state=eyJxxx.yyy.zzz**
  {"app_id": <malicious>,
  …}
  or platform's internal state
  (e.g., session, frontend-managed state)

BUT

- **redirect_uri:**
  platform.com/<spotify>/redirect

app_id

Enforce Matching at R

Mismatch Detected!

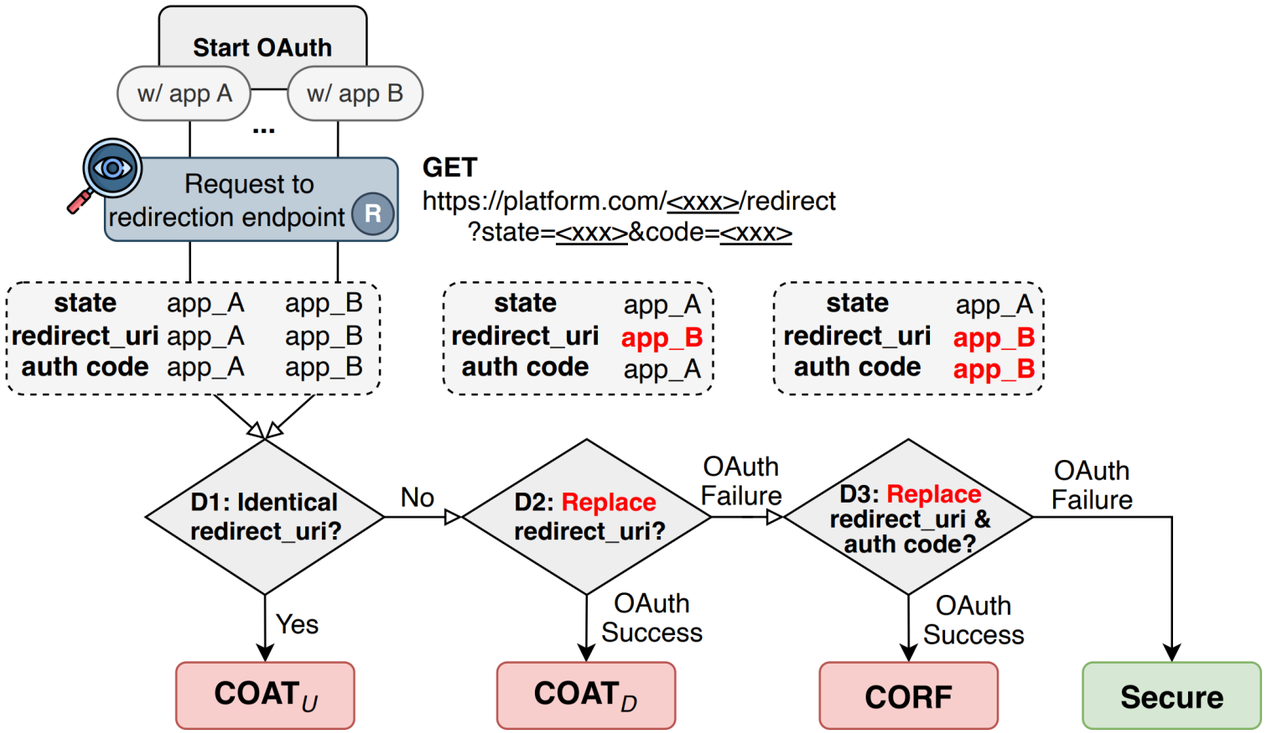
Vulnerability Detection: Methodology

Decision Tree-based approach

D1. Does the platform assign identical `redirect_uri` for both apps?
- **Yes:** Vulnerable to **COAT**$_U$;
- **No:** Proceed to D2.

D2. What is the OAuth outcome of replacing the distinctive element (*e.g.*, app ID) of `redirect_uri` in the request to the redirection endpoint?
- **Succeed:** Vulnerable to **COAT**$_D$;
- **Fail:** Proceed to D3.

D3. On top of the substitution made in D2, what is the OAuth outcome of replacing the auth code as well?
- **Succeed:** Vulnerable to **CORF**;
- **Fail:** Protection in place, secure.



- No knowledge of platform internals, which are blackbox (closed-source, server-side)
- No need to integrate malicious apps to launch actual attacks
- Only inspect & modify traffic suffice

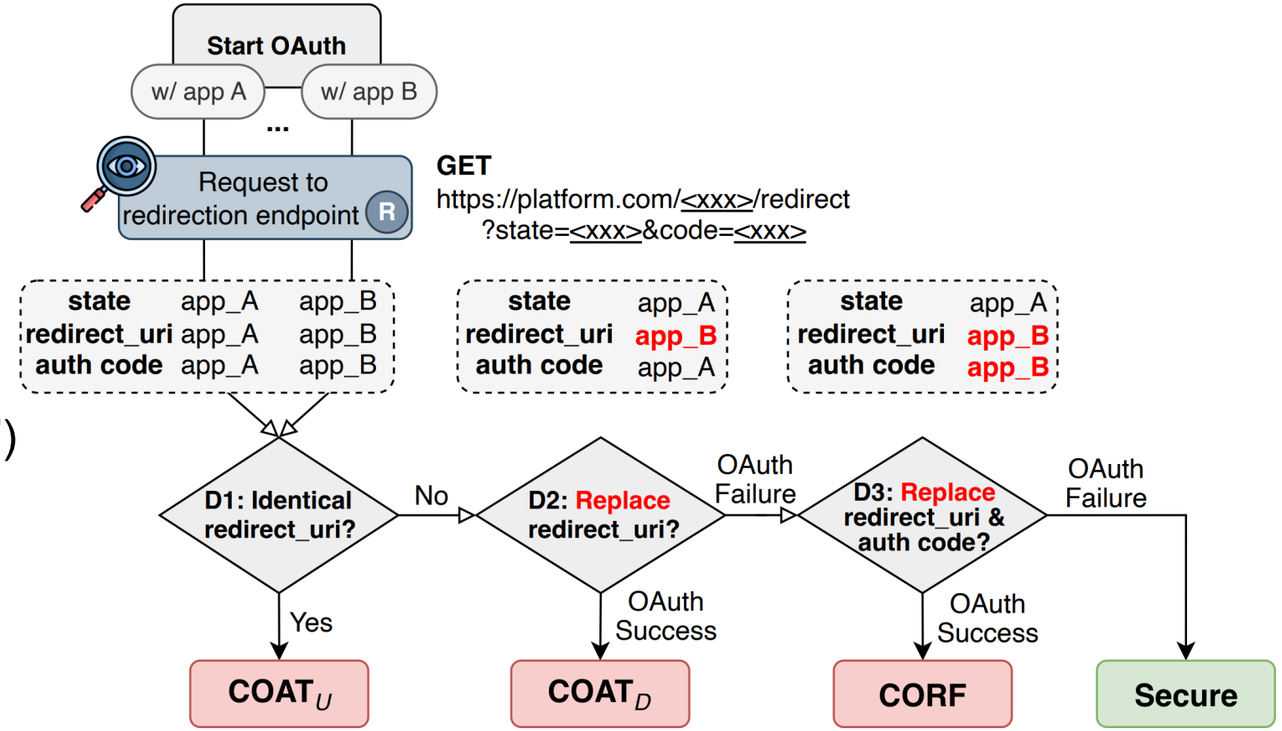
Vulnerability Detection: *COVScan*

COVScan: Cross-app OAuth Vulnerability Scanner

Semi-automated dynamic testing tool:

- Follow the ***decision tree***
- Manage ***state machine*** to track OAuth steps
- ***Inspect & modify traffic*** with selenium-wire/ mitmproxy
- Determine ***OAuth outcome*** with ***keyword filtering*** (e.g., “error”, “unsuccessful”, “invalid”)
- Manual navigation of the platform UI

- At most ***3 OAuth flows*** to derive concrete conclusion for a platform



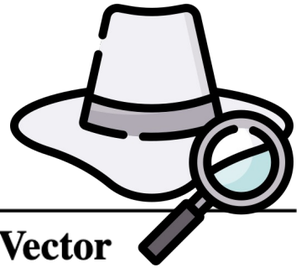
selenium-wire: https://github.com/wkeeling/selenium-wire
mitmproxy: https://github.com/mitmproxy/mitmproxy

Evaluation

Scope

24 Platforms — *open marketplace* → 18 Platforms

Bug Hunting



Type	Platform	# Users	COAT		CORF	Detectable	Attack Vector	
			COAT_U	COAT_D			App Distribution	Single-Click
6 Workflow Automation Platforms	Microsoft Power Automate	33M MAU	☠	☠		✓	*Share*, *Publish*	✓
	IFTTT	27M				✓	N/A	N/A
	Zapier	2.2M				✓	N/A	N/A
	A Business Collab. Platform	54M MAU	☠			✓	*Share*	✓
	Workato	21K Orgs	☠			✓	*Share*, *Publish*	
	A Top-tier iPaaS	70K Companies	☠			✓	*Publish* + *Share*	✓
6 Virtual Voice Assistants	Google Assistant	500M MAU		☠		N/A [1]	*Share*, *Publish*	
	Amazon Alexa	100M		☠		✓	*Share*, *Publish*	✓
	Samsung Bixby	200M		☠		N/A	*Publish*	
	Xiaomi XiaoAI	115M			☠	✓	*Publish*	✓
	Baidu Xiaodu	40M			☠	✓	*Publish*	✓
	Alibaba AliGenie	40M			☠	✓	*Publish*	

[1] Vulnerability fixed before COVScan came into being.

COAT_U: COAT with universal `redirect_uri` for multiple apps;

COAT_D: COAT with distinct (per-app) `redirect_uris`.

Evaluation

Bug Hunting (cont'd)

Type	Platform	# Users	COAT		CORF	Detectable	Attack Vector	
			COAT_U	COAT_D			App Distribution	Single-Click
4 Smart Homes	Google Home	500M Installs		💀		N/A	*Share*, *Publish*	
	Samsung SmartThings	285M	💀			✓	*Share*, *Publish*	✓
	Xiaomi Mi Home	83M			💀	✓	*Publish*	
	Yandex Smart Home	45M	💀			✓	*Share*, *Publish*	✓
2 LLMs w/ Plugins	A leading LLM platform	180M WAU			💀	✓	*Share*, *Publish*	
	ByteDance Coze	2M MAU	💀			✓	*Share*, *Publish*	✓
Total	18		7	5	5	15		9

COAT_U: COAT with universal `redirect_uri` for multiple apps;
COAT_D: COAT with distinct (per-app) `redirect_uris`.
Detectable: Platform's (in)security detectable by COVScan.
Single-Click: Feasibility of Single-click attack w/o malicious app distribution.

Share: Share a custom app individually w/o vetting;
Publish: Go through vetting process and publish in marketplace.

Summary

- **16/18** are vulnerable
- **11** to COAT, **5** to CORF 💀
- **9** can be done in 1-Click
- **COVScan** can precisely assess the platforms' (in)security

Responsible Disclosure

- Informed all 16 vulnerable platforms
- Confirmed by 11 platforms CVE-2023-36019 CVSS: 9.6
- Patched by 9: 6 Robust fix, 3 Extra consent screen

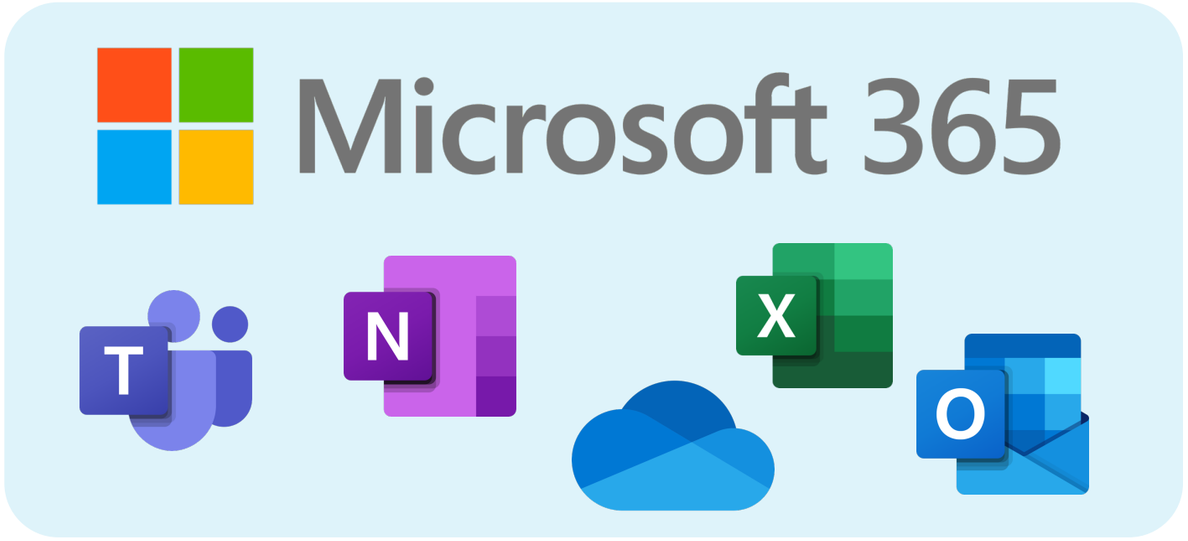
Worst Case: 1-Click Account Takeover

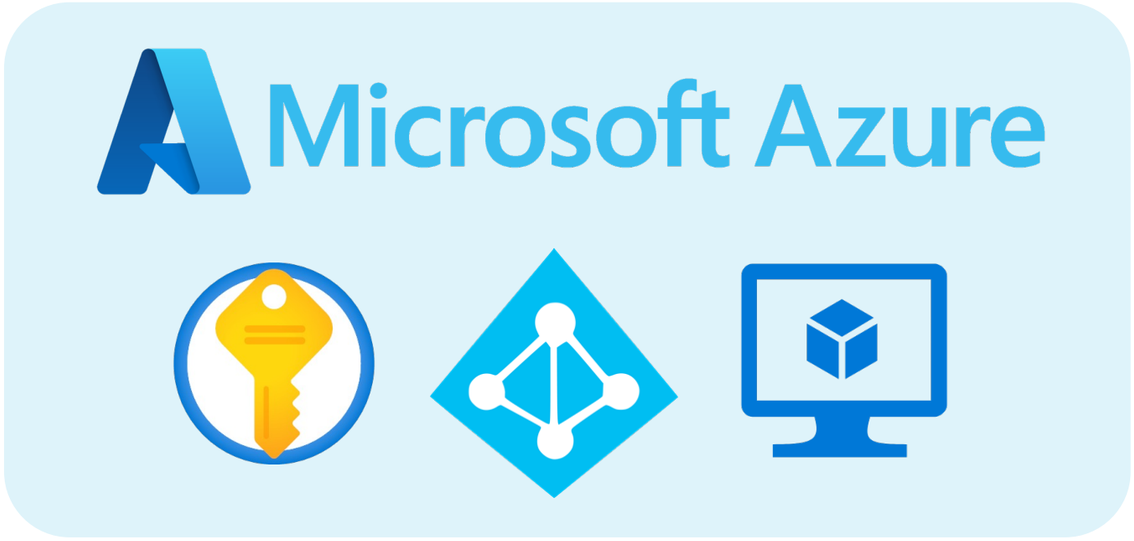
With just a click on an unassuming link

Microsoft Power Automate

CVE-2023-36019 CVSS: 9.6

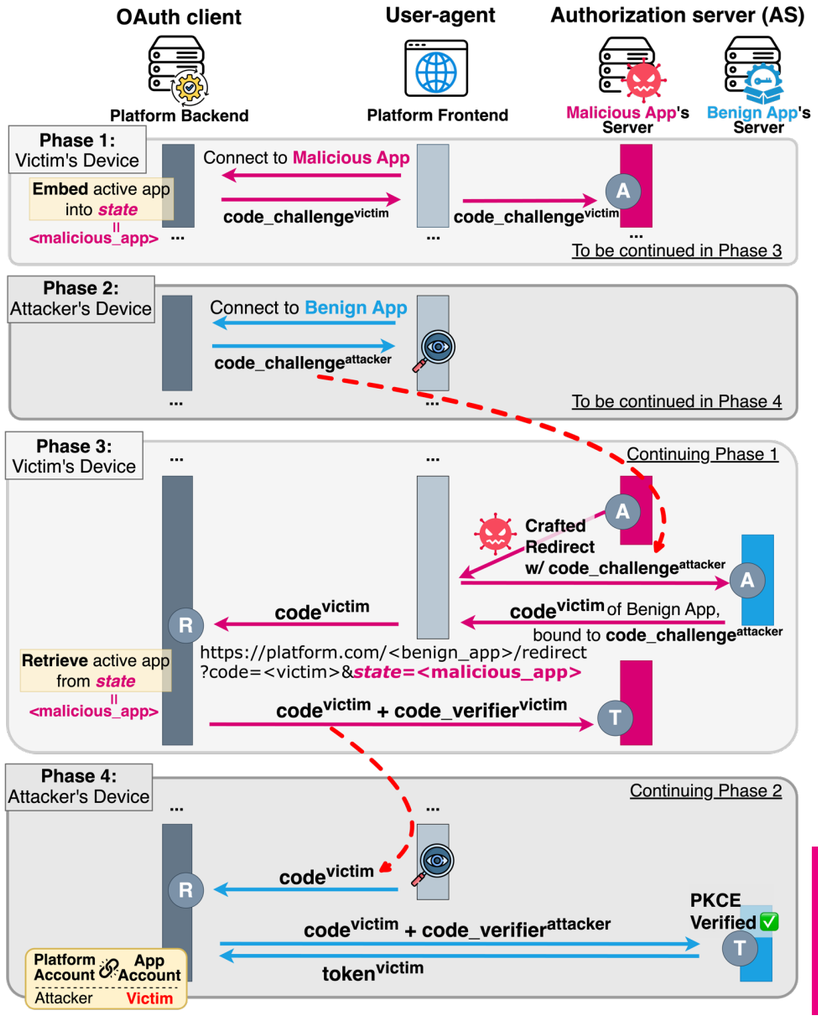
- Steal Office 365 Outlook emails
- Leak Azure Key Vault secrets (1 more click to steal another app's access)
- And more … (50+ Apps/Services in Microsoft 365 and Azure)

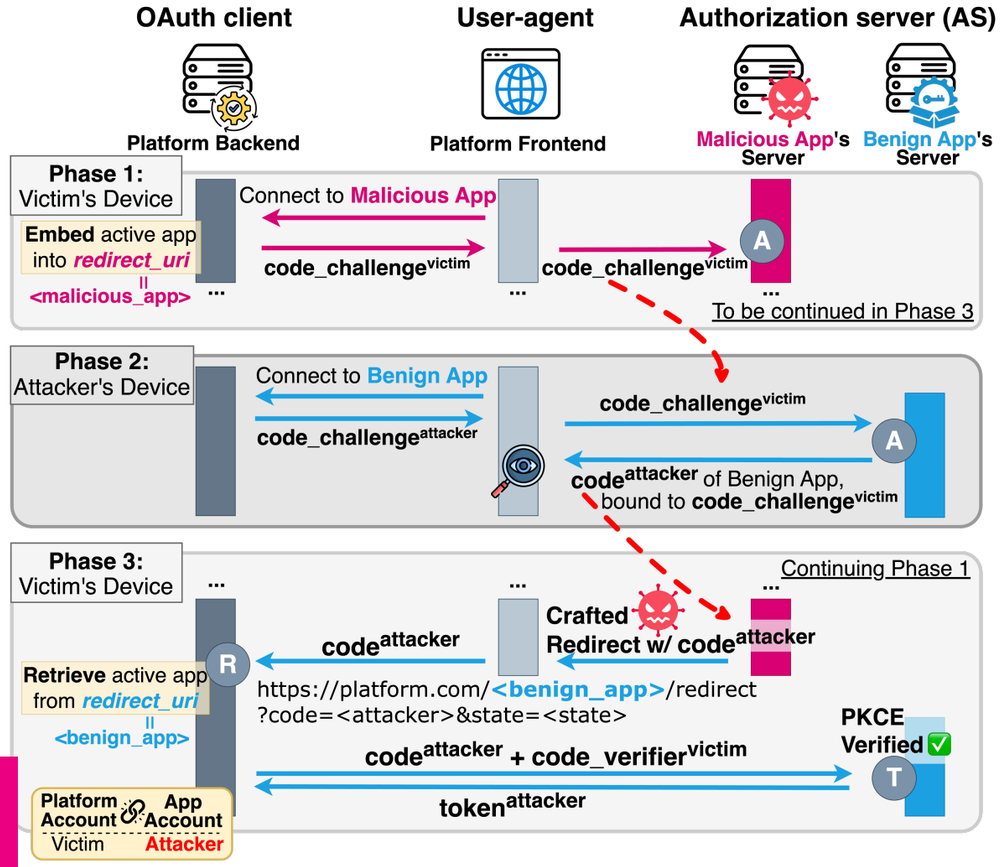




Note 1: PKCE cannot solve the problem

RFC 7636 - Proof Key for Code Exchange (PKCE)





Invalidate PKCE protection!

COAT w/ PKCE: (PKCE Chosen Challenge Attack)
Victim uses Attacker's `code_challenge`

CORF w/ PKCE:
Attacker uses Victim's `code_challenge`

Note 2: Comparison with Mix-up Attack

(IdP) Mix-up Attacks

Initial Discoveries

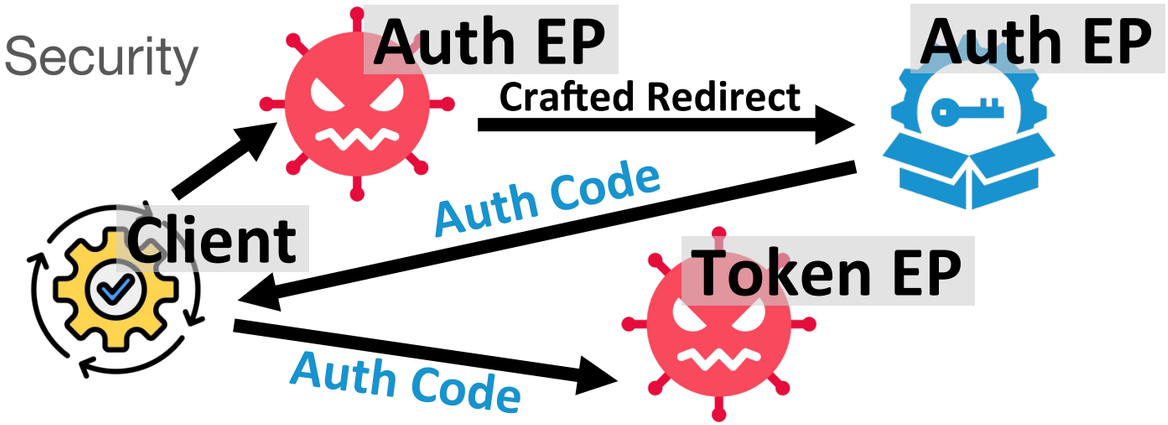
- [CCS '16] A Comprehensive Formal Security Analysis of OAuth 2.0
- [EuroS&P '17] SoK: Single Sign-On Security — An Evaluation of OpenID Connect

Standardization Efforts

- [RFC9207] OAuth 2.0 Authorization Server Issuer Identification
- [RFC9700] Best Current Practice for OAuth 2.0 Security

Major Claims for COAT/CORF

- Typical Attack Scenario (and attack practicality) is different
- Existing Mix-up Defenses are impractical for Integration Platforms



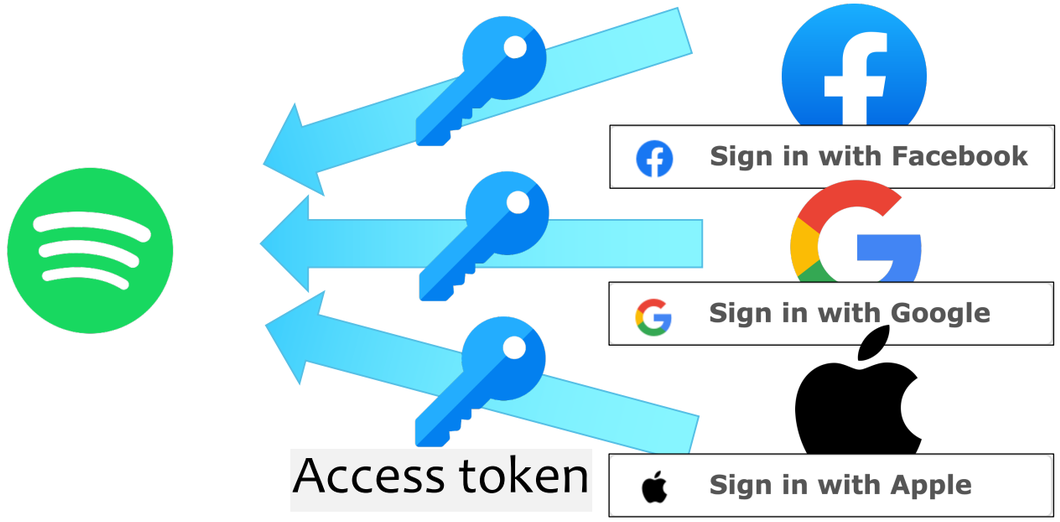
Note 2: Comparison with mix-up attack

Attack Scenario – Trust Implications of the Paradigm Shift

OAuth for Single Sign-on (SSO)

OAuth client
a.k.a. Relying party (RP)

Authorization server (AS)
a.k.a. Identity provider (IdP)

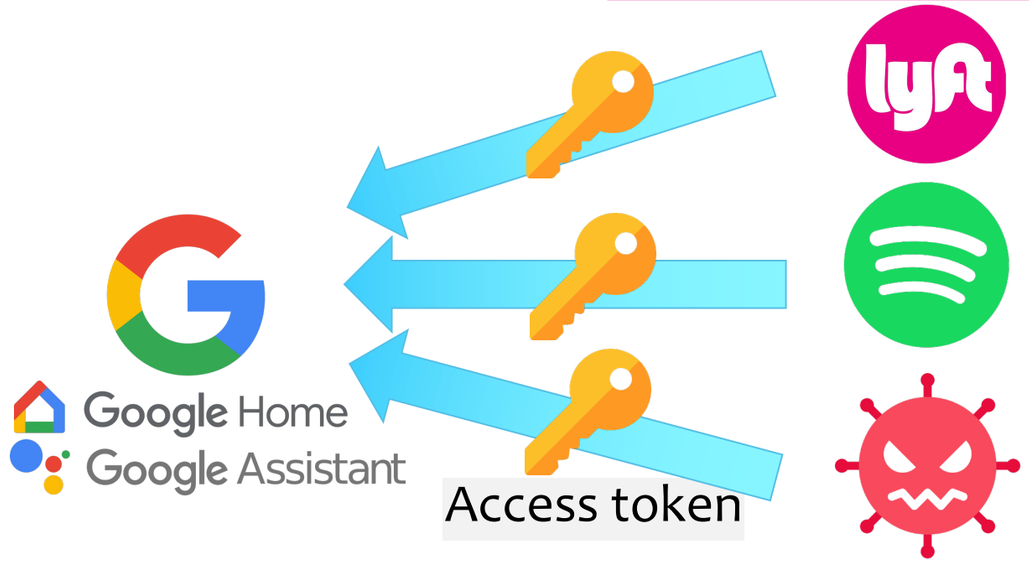


Theoretical attacks ← **Trusted** IdPs

OAuth for "Account Linking" in Integration Platforms

OAuth client

Authorization server (AS)



Practical attacks ← **Untrusted** Apps

- Multiple Auth Servers: Easy
- One of them is malicious: **Hard**

❖ (IdP) mix-up attack → Cross-app OAuth Account Takeover (COAT)
❖ Naïve RP session integrity attack → Cross-app OAuth Request Forgery (CORF)

Note 2: Comparison with mix-up attack
Existing Defenses – Impractical for Integration Platform

Security Requirement

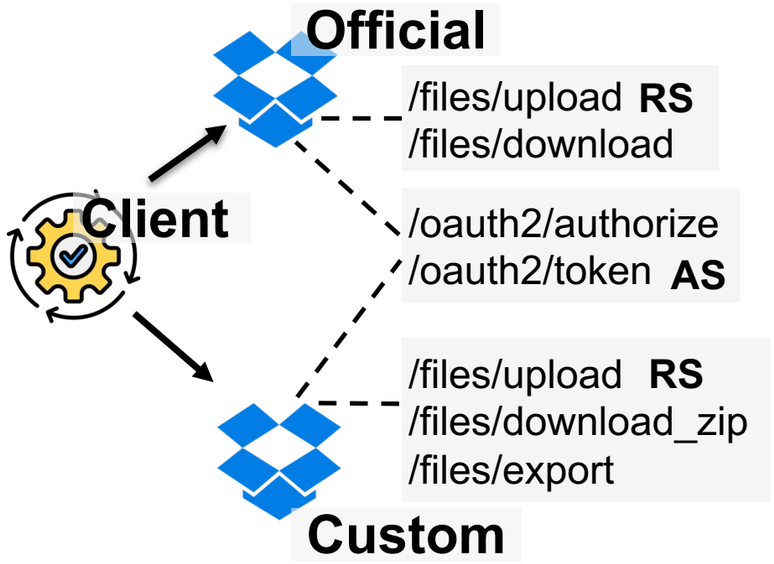
Compare *received issuer (from authorization response)* with *stored* issuer *(in session)*

Functional Requirement

Why are Issuer-based defenses not practical?

- **Two apps can share the same AS:**
  - *issuer* (per-AS ID) is not unique
  - per-App ID is unique.
- **Scalability concerns:**
  - Most apps lack RFC8414-compliance: No trusted ground truth for *issuers.*
  - Most platforms also do NOT support fetching AS Metadata per RFC8414, or identifying the <auth EP, token EP> tuple for each app

Issuer-sharing Apps



- ***(AS) Issuer:***
  `iss` claim in RFC8414 OAuth AS Metadata, or an abstract identifier for the AS' <authorization endpoint, token endpoint>

Note 2: Comparison with mix-up attack

New Countermeasure – App ID as the isolation boundary

Security Requirement

Compare *received app ID (from authorization response)* with *stored app ID (in session/state)*

Functional Requirement

- ***(AS) Issuer:***
  `iss` claim in RFC8414 OAuth AS Metadata, or an abstract identifier for the AS' <authorization endpoint, token endpoint>
- ***(Integrated) App / Integration:***
  - AS <authorization endpoint, token endpoint>
  - Client Registration
  - RS

Practical Defense:

- Use a ***per-app ID*** rather than a ***per-authorization server ID (issuer)***, to better reflect the multi-app nature of integration platforms.
- To maximize compatibility, impose ***no new dependencies*** on apps' authorization servers that are only compliant with original OAuth spec [RFC6749].

⇒ Essential for securing platforms integrated with hundreds of apps, potentially with shared *issuers.*

- Defense applies to ***both COAT and CORF***.

Contribute to the Community

- Present & Discussed at:
  - OAuth Security Workshop 2025 (OSW 2025) [link]
  - IETF Agenda Meeting (IETF 123) [link]

- Published IETF Draft:

  Tim Würtele, Pedram Hosseyni, Kaixuan Luo, and Adonis Fung. **Updates to OAuth 2.0 Security Best Current Practice.** Internet-Draft draft-wuertele-oauth-security-topics-update-01, Internet Engineering Task Force, June 2025. Work in Progress.

  https://datatracker.ietf.org/doc/html/draft-wuertele-oauth-security-topics-update-01

Proposed updates to
RFC 9700 - Best Current Practice for OAuth 2.0 Security

Key Takeaways

- **Attack:** As ***open ecosystems***, Integration Platforms enable practical Cross-app OAuth Attacks via ***malicious app integrations***, leading to account takeovers or forced linking.
- **Defense:** Existing RFCs provide ***authorization server-specific*** defenses but lack ***app-specific*** defense. A per-app ID is the appropriate isolation boundary for integration platforms.
- **Pervasiveness:** Identified 15+ vulnerable mainstream platforms, with hundreds to thousands of integrated apps per platform.
- **Vulnerability Detection:** Developed ***COVScan***, a decision tree-based testing tool.
- **Community Impact:** Helped secure platform vendors across the industry; Contacted IETF & Working on updates to OAuth Security specification.

34TH USENIX
SECURITY SYMPOSIUM

Universal Cross-app Attacks: Exploiting and Securing OAuth 2.0 in Integration Platforms

Learn more?

https://mobitec.ie.cuhk.edu.hk/cross-app-oauth-security

Kaixuan Luo[1], Xianbo Wang[1], Adonis Fung[2], Wing Cheong Lau[1], Julien Lecomte[2]

1 The Chinese University of Hong Kong, 2 Samsung Research America