Experimental Security Analysis of the App Model in Business Collaboration Platforms

Yunang Chen*, Yue Gao*, Nick Ceccio, Rahul Chatterjee, Kassem Fawaz, Earlence Fernandes+

University of Wisconsin–Madison, University of California San Diego

* Equal contribution
+ Work done while at UW-Madison
Business Collaboration Platforms (BCPs)

- Productivity & Team Collaboration
- Third-Party Integrations (Apps)

Figures from slack.com and support.microsoft.com
BCPs Have Become A Hub for Sensitive Resources

- Zoom Calls
- DropBox File Sharing
- Email Forwarding
- Code Repository Management
- …

What if the apps are malicious?

Can BCPs enforce security correctly?
Background: App Workflow

“Zoom, start a meeting.”

1. Interact
2. API Calls
3. Notification
4. API Calls
5. Response

BCP User -> BCP Client -> BCP Server -> Cloud BCP Apps

Message API

User posted a message…

Render video call…

Video Call API: zoom link

"Zoom, start a meeting."
Background: App Installation

1. App Requests Permissions

2. Read Permission Scopes
   - Read user identity
   - Read public messages

3. Write Permission Scopes
   - Post messages
   - Post messages on behalf of users

4. User Approves Permissions
Threat Model: Malicious Apps in BCP

- Attacker tricks the user to install a malicious app
- The user is curious and installs a malicious app
- The benign app becomes malicious
Challenges & Our Methodology

- Incomplete permission model description.
  
  We extract a unified abstraction.

- Closed-source apps in the cloud.

  We examine all possible interactions.

- Unscalable in-depth analysis.

  We estimate potential attackers & victims.
A Two-Level Unified Permission Model

• Level 1: coarse-grained OAuth permissions scopes

Resource Group: Private Channels

BCP App

“read private channel messages”
A Two-Level Unified Permission Model

- Level 1: coarse-grained OAuth permissions scopes
- Level 2: fine-grained runtime policy checks

BCP App

“read private channel messages”

“refers to private channel #1”

Resource Group: Private Channels

Channel #1

Channel #2

Channel #3
Violation of Security Principles

• Least Privilege
  Runtime policies are ad-hoc and incomplete.
  “post messages to channels”
  “only if the app joined this channel”

• Complete Mediation
  Provenance of resources are not properly tracked.
  “post messages to users”
  null

delegate message → cloud → “command” → “command” → user “command”
All Types of Interactions Are Vulnerable

- App-to-App Interaction ➔ Delegation Attacks
  - Post scheduled message as me ➔ Talk to apps as user

- User-to-App Interaction ➔ Command Hijacking
  - Type “/zoom” to start a meeting ➔ Hijack (also register “/zoom”)

- App-to-User Interaction ➔ Privilege Escalation
  - “read direct message” ➔ “read private channels including other users”
Delegation Attack: Merge Malicious PRs

- **Step 1:** User installed Bitbucket app.

  “Merge #1”

  “Confirm merge #1?”

  “Yes”

  “Merged!”
Delegation Attack: Merge Malicious PRs

• **Step 2:** User installed “Send Later” app *(our malicious demo)*.
Delegation Attack: Merge Malicious PRs

• **Step 3:** Attacker creates a malicious Pull Request #1.
Delegation Attack: Merge Malicious PRs

- **Step 4:** Malicious “Send Later” app talks to the Bitbucket app.

  - First request “merge #1”
  - Confirm “merge #1”
  - “Pull request was successfully merged.”
Delegation Attack: Merge Malicious PRs

• **Step 5:** Bitbucket merges the malicious pull request.
Potential Prevalence Analysis

- Collect each app’s requested permissions.
- **Capable Apps** — Have write permissions needed for attacks.
- **Susceptible Apps** — Have read permissions affected by attacks.

| Attacks               | # Capable Apps (MS Teams) | # Capable Apps (Slack) | # Susceptible Apps (Slack) |
|-----------------------|---------------------------|------------------------|---------------------------|
| Delegation Attacks    | 427 (33%)                 | 563 (23%)              | 1,493 (61%)               |
| Command Hijacking     | 77 (6%)                   | 270 (11%)              | 1,266 (52%)               |
| Privilege Escalation  | n/a                       | 11                     | n/a                       |
Countermeasures: Improve Permission Models

Better Design

– Finer-grained Scopes
  – “post messages as user”
  – “… to apps”
  – “… to non-apps”

– Stricter Runtime Policies

Better Execution

– Track Provenance of Actions
  – user: message
  – Content: message
  – From: User (via App)

– Explicit User Confirmation

Message from B

User A

“check if apps can read A”

“check if apps can read A & B”

Start meeting “/zoom”

“Zoom #1 or Zoom #2 ?”
Disclosure & Responses

• Confirmed attacks
• Workspace ➔ a trusted environment
• Administrator ➔ will correctly manage apps

• Our tips for administrators
  • Consider limiting users from installing apps
  • Actively monitor the behavior of installed apps
  • Only approve delegation permissions from trusted apps
Summary

- BCPs have become a hub for **sensitive third-party resources**.
- We provide **security analysis** under malicious apps.
- All types of interactions are **vulnerable & potentially prevalent**.