A First Look at Privacy Compliance of Zoom Apps

Andrew York[†], Mohammed Aldeen[†], Jingwen Yan[†], Pranav Silimkhan[§], Song Liao[‡], Mert D. Pesé[†], Long Cheng[†]
[†]*School of Computing, Clemson University, Clemson, USA*
[‡]*Department of Computer Science, Texas Tech University, Lubbock, USA*
[§]*Viasat Inc., USA*
{adyork, mshujaa, jingwey, mpese, lcheng2}@clemson.edu;
song.liao@ttu.edu; pranavpradosh.silimkhan@viasat.com

***Abstract*—Video conferencing applications are extensively utilized in various day-to-day activities. Since the onset of the COVID-19 pandemic, the usage of such applications has experienced a significant surge. Among these, Zoom stands out as one of the most popular video conferencing platforms. In 2021, Zoom unveiled its apps marketplace, a groundbreaking initiative aimed at empowering developers to augment the platform's functionality by introducing their own applications tailored to enhance its capabilities. These apps extend third-party access to data about Zoom's users and their usage of the Zoom platform, potentially sacrificing Zoom users' privacy. In this paper, we conduct comprehensive analysis of privacy compliance in Zoom marketplace apps by systematically examining the alignment between apps' granted permissions and their privacy policy disclosures. To identify these apps' data practices and privacy concerns, we collected the metadata and privacy policies of 2,733 apps on the Zoom marketplace. We found that 87% of these had reachable privacy policies. Of those policies, only 14% had logical contradictions that reduced their clarity. However, we found that a staggering 99.7% of these policies failed to disclose all data practices. In addition, we demonstrated that Zoom apps can run inside the client application without valid authentication with Zoom, enabling attackers to modify app code after being approved and exploit granted OAuth scopes to access contact information, meeting data, and user account details beyond disclosed privacy practices we identified.**

***Index Terms*—Zoom Apps; Privacy Policy; Privacy Compliance; Natural Language Processing**

I. Introduction

In the wake of the unprecedented global upheaval caused by the COVID-19 pandemic, the ubiquitous integration of video conferencing applications (apps) into the fabric of our daily lives has been nothing short of remarkable. Among the myriad of video conferencing solutions that have emerged as indispensable tools in this digital era, Zoom stands out as one of the front runners [1] [2], capturing widespread attention for its user-friendly interface, robust feature set, and seamless cross-platform compatibility. As millions around the globe turned to Zoom to bridge the gap created by physical isolation, the platform witnessed an exponential surge in its user base [3], underscoring its pivotal role in facilitating remote collaboration and connectivity.

Despite its success, Zoom's security and privacy protections quickly came under scrutiny. In April 2020, a Zoom shareholder filed a class-action lawsuit, alleging that Zoom had misled users by claiming its meetings were end-to-end encrypted when they were not [4]. In response, Zoom took actions to improve its security and privacy. Zoom appeared to have improved its security, even becoming the first video conferencing application to achieve Common Criteria certification [5]. In addition, Zoom embarked on a pioneering endeavor by unveiling its marketplace: a dynamic ecosystem designed to empower developers to harness the full potential of the platform through the creation and integration of third-party apps [6]. Through this marketplace, developers are afforded a unique opportunity to craft bespoke solutions tailored to the specific requirements and preferences of Zoom users.

However, amidst the excitement surrounding the proliferation of third-party apps within the Zoom marketplace [7], concerns [8] [4] pertaining to security, compliance, and regulatory adherence have emerged as paramount considerations warranting careful examination and scrutiny. Liu and Biczók [9] studied the occurrence of third-party apps that exist across multiple platforms such as google play, Zoom Marketplace, browser extensions and Google workspace. They show that these apps can disclose a user's private information across multiple platforms, including apps that can be added to the Zoom client. While they studied these risks associated with Zoom's third-party apps, they did not consider the broader issue of how third-party apps in Zoom can expose a user's information without dependence on other platforms. In other work, Dassel and Klien [10] interviewed educational users and identified three distinct response profiles—'Agnostic', 'Pragmatic', and 'Sceptic'—in how people weigh Zoom's well-publicized privacy and security risks, but their analysis focuses only on Zoom's core platform and does not address the additional privacy exposures introduced by third-party apps in the Zoom Marketplace, which share user data with external developers. Kagan et al. [11] explored the potential leakage of meeting participant identities via images from those meetings. They also did not consider the leakage of these identities through marketplace apps. Chen and Zou [12] described the impact of Zoom's unexpected growth on the company and its practices. They investigated the company's maturation of its own security and privacy practices during the COVID-19 pandemic, but also did not expand their research to marketplace apps.

In this work, we conduct a comprehensive analysis of Zoom Marketplace apps' privacy compliance by systemati-

2025 IEEE Secure Development Conference (SecDev) | 979-8-3315-9595-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/SECDEV66745.2025.00012

979-8-3315-9595-1/25/$31.00 ©2025 IEEE
DOI 10.1109/SecDev66745.2025.00012

cally examining the alignment between each app's granted permissions (Scopes) and their privacy policy disclosures. To do this, we first extract each app's public metadata to measure baseline compliance with Zoom's disclosure rules. Next, we parse every reachable privacy policy, assessing whether it is clear, concise, and free of internal contradictions. Finally, we collected privileges granted to these apps and their associated data practices in order to determine if these practices were disclosed to the user in the associated privacy policies.

This study will address the following research questions. We begin by determining the availability of privacy policies among Zoom Marketplace apps **(RQ1)**, establishing the baseline for our study. Next, we examine the clarity and consistency of those policies, identifying any logical contradictions or narrowing clauses **(RQ2)**. With policy quality assessed, we then evaluate whether disclosed documents fully cover every data practice implied by each app's granted scopes **(RQ3)**. After finding in RQ3 that some privacy policies omit certain app-granted privileges, we next test whether Zoom enforces app authentication at runtime, and observe if apps can still have their code executed even with invalid credentials—meaning previously granted privileges could be exercised despite failed authentication **(RQ4)**.

To answer the above research questions, we conduct the following analysis and make the following contributions:

- We systematically collected metadata, OAuth scopes, and privacy-policy URLs for all 2,733 Zoom-Marketplace apps and verified whether each policy privacy page could be reached. We found that 87% of apps delivered a reachable policy page; the remaining 13% offered users no accessible notice[1].
- We examined every reachable privacy policy for (i) internal contradictions and (ii) omissions relative to the app's declared scopes. We observed that 14% of policies contain logical contradictions and were less likely to be understood, and 99.7% fail to cover all data practices by their associated app.
- We created a test app to evaluate Zoom app authentication, then served the JavaScript from the same launch URL after providing invalid authentication credentials, highlighting the fact that apps can run inside of the client without working authentication with Zoom.

The remainder of this paper is organized as follows: Section II provides necessary background on Zoom's app ecosystem and integration model. Section III describes our methodology and analysis pipeline. In Section IV, we present our results regarding privacy policy accessibility, policy inconsistencies, and undisclosed data practices. Section V discusses the privacy and security implications of our findings, and Section VI reviews related work. We conclude in Section VII with a summary of our key insights and suggestions for future research.

[1]Data and source code are available at: https://anonymous.4open.science/r/new-zoom-app-analysis-2D3B/README.md

II. Background

As Zoom's usage expanded beyond simple video conferencing, its third-party app ecosystem grew into a critical surface for both innovation and privacy risk. In July 2021, Zoom introduced "Zoom Apps," enabling developers to embed custom web applications directly within the Zoom client interface [13]. This feature saw rapid uptake: by the end of 2022, the marketplace hosted over 1,000 apps [14], and as of May 2025, it offers more than 3,000 apps covering diverse workflows such as customer relationship management, collaborative white boarding, and many others [15].

A. Architecture and Integration Model

Under the hood, Zoom Apps are JavaScript-based web applications built on the Zoom Developer Platform, which provides APIs and SDKs that allow third-party developers to create and publish custom apps alongside Zoom's own apps. Figure 1 illustrates the end-to-end life cycle of a Zoom App once users first browse and install an app via the Zoom App Marketplace (accessible in either the Zoom client or a web browser). In this architecture, developers register a launch URL that hosts their app on any Internet-accessible server; at runtime, the Zoom client loads this URL into a WebKit-based embedded web view powered by the Zoom Apps SDK [16]. Zoom stores only the app's metadata and launch URL in its backend; the app's JavaScript and business logic reside on the developer's own server and are fetched into the client's WebKit-based embedded web view at runtime, as shown in Figure 2. This separation enables lightweight marketplace

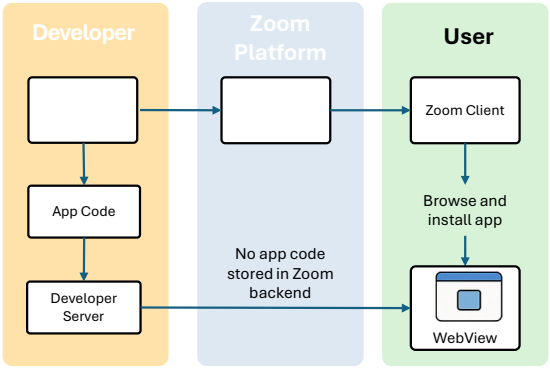

Fig. 1: Third-Party App Integration Model in Zoom Marketplace

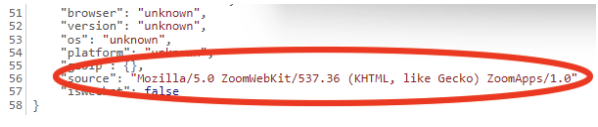

Fig. 2: Embedded Browser Configuration for Third-Party App Execution in Zoom Client. The user-agent string shows ZoomWebKit to support Zoom apps run within a customized WebKit-based browser environment.

management but also can place untrusted code within the context of the user's meeting environment.

B. Permission and Data-Access Model

When developers add their apps to Zoom, they must additionally identify which Zoom APIs they plan to use. These "APIs" correspond to methods that are defined in the various Zoom SDKs, which are selectable by the developer. These SDKs define methods for Meeting Actions, Meeting Views, Sharing, Reactions, the App Window, Layers, Recordings, Client Settings, Waiting Rooms, and many more functions. These methods allow the app to get and set numerous private details about the Zoom user and their session [17]. These methods can also control the Zoom client's behavior.

Zoom uses OAuth scopes to control access to information and capabilities. While the API methods specify what actions an app may take, the scopes constrain what these actions can access and manipulate. These scopes can allow access to meeting information, user information, and many other aspects of the Zoom client and its session [18]. They grant permissions to get and change information about the Zoom user's account, their meeting, and other Zoom functions. As shown in Figure 3, the zoomapp:inmeeting scope is required, at a minimum, to access the app within the Zoom client.

C. Security Compliance with Zoom's Policies

Before any app becomes publicly available, Zoom enforces a multi-stage review process covering submission completeness, branding, functionality, compliance, and security auditing [19]. As part of the submission checklist, developers must provide a publicly accessible privacy policy, terms of service, support URL, and documentation URL. Zoom requires developers to submit a Technical Design Document (TDD) describing how their app is built and secured, but does not require the actual source code for review [20]. After an app passes, the developer can update or completely replace the JavaScript served from that URL at any time—without triggering a new review or notifying users. We can safely assume that their testing approaches are limited to those that can be performed external to the application, such as dynamic analysis and penetration testing. This mirrors a vulnerability already observed on other major collaboration platforms. In Slack and Microsoft Teams, for example, researchers found third-party apps operating without thorough code vetting, exposing sensitive organizational data [21].

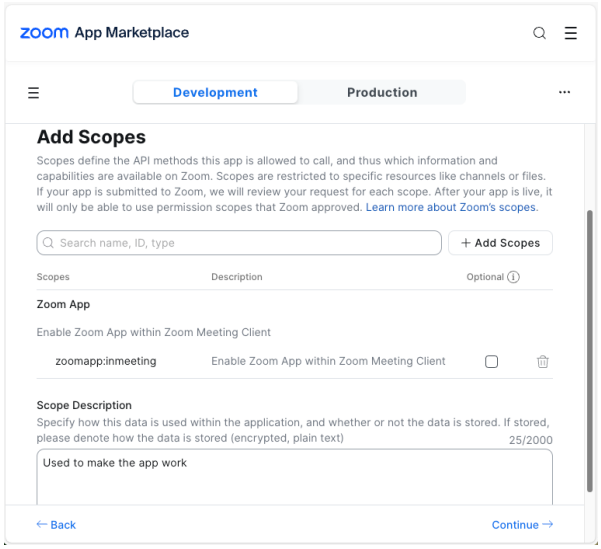

Fig. 3: Selecting Scopes for an App

Under Zoom's app-marketplace submission checklist, all URLs, including the privacy-policy URL, must be hosted on fully qualified domain names and undergo Zoom's one-time domain-validation process before an app can go live [22]. While Zoom supplements structural checks with a technical review, requiring a detailed TDD and OWASP-based security testing of OAuth scopes, the review focuses on functional security rather than the integrity or completeness of privacy disclosures [19]. In fact, many third-party apps still over-collect sensitive audio-visual metadata and obscure their data-sharing practices in opaque policy language [23].

III. Methodology

A. System Overview

Figure 4 depicts the processing pipeline that we employed in this study. We first collect app metadata directly from Zoom's public Marketplace, capturing the full set of OAuth scopes, developer descriptions, and policy links (Section III-B). Next, we validate the accessibility of each privacy policy links to distinguish between accessible and broken links (Section III-C). Once a policy is successfully retrieved, we feed the text into a semantic analyzer to detect contradictions and narrowing clauses that may hide true data practices (Section III-D). Then, we extract and compare data-practice statements implied by the app's scopes against those declared in its privacy policy to highlight undeclared privileges (Section III-E). Finally, to test whether Zoom enforces valid OAuth credentials before loading and executing app code, we deploy a simple app with incorrect (dummy) OAuth credentials and observe that the client loads and runs the app despite invalid credentials, such that pre-authorized scopes may remain reachable during execution (Section III-F).

Ethical Consideration: Our research collected publicly available metadata and privacy policies. We used neither a Zoom user account nor any other private credentials to collect data about the apps. Our Marketplace API crawler was single-threaded, collecting lists of App meta data in sets of 30 at a time. The individual privacy policies were retrieved sequentially using a single thread of the Microsoft Edge browser, executed over a period of a few days to avoid overloading Zoom's servers. This activity did not generate a large amount of traffic with Zoom, as these policies are hosted separately by each app's publisher.

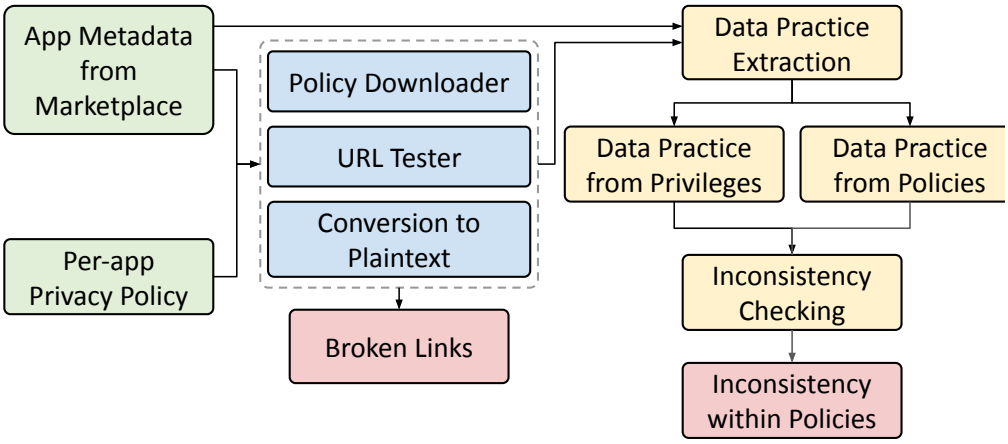

Fig. 4: Overview of our Processing Pipeline for Analyzing Zoom Apps Privacy Practices

B. Data Collection

In Zoom's architecture, third-party apps are delivered as JavaScript bundles that run inside the client's embedded web view, making their source code and their data-handling logic inaccessible to both end users and Zoom itself, as depicted in Figure 1. Also, many apps further require separate logins in the web view itself (i.e., Google login, Slack, etc.) before their interfaces become active, making crawling of these infeasible. To overcome this, we leverage Zoom's public Marketplace API [24], which provides for each app, a structured JSON record containing (1) the complete set of OAuth scopes, such as Zoom's permission strings that govern exactly which user data types an app may read or modify—and (2) the developer's privacy-policy URL. By analyzing these two metadata fields, we obtain a clear summary of what an app *could* do from the scopes versus what it *claims* it will do from the privacy policy, all without ever inspecting the app's underlying source code.

To automate the process across the entire Zoom Marketplace, we implemented a two-phase crawler in Python using the *requests* [25] library. In the first phase, the crawler invokes the paginated *"List Apps"* endpoint, illustrated in Figure 5 — which returns an array of app summaries (including each app's *'id', 'name'*, and *'description'*) but does not include references


```
[
    {
        "total": 2916,
        "pageNum": 1,
        "pageSize": 30,
        "uniqueRequest": null,
        "essentialAppIntroduction": null,
        "apps": [
            {
                "id": "VG_p3Bb_TwWe_bgZmPUaXw",
                "name": "Google Workspace",
                "displayName": "Google Workspace",
                "icon": "/XgIKAv2wQausbEiQUdcxXg/
                nhYXYiTzSYWf4mM3ZO4_dw/app/vYW9O0TIRP6HEXBue_jBYw/
                aeYqocK9SO-wztyqcSC4XQ.png",
                "darkModeIcon": "",
                "description": "Use Zoom to easily schedule, join,
```


Fig. 5: Paginated Marketplace Metadata


```
1-click. Generate meaningful meeting details by automatically
adding the topic, attendees, and attachments based on subject,
recipients, and sent documents. And customize meeting options like
join with video on, join with audio muted, join before host, and
other options to help you host your meetings your way.</p><p><br></
p><p>You can also place Zoom Phone calls directly within your
Google email or Google Calendar workspace. Simply click on the Zoom
icon on the right side of the screen and search for the contact you
wish to call or use the dialpad. You can even call the list of
attendees for a specific meeting or event by selecting the meeting
or event from your calendar and the contact information for the
attendee list will populate under the Call Contacts list.</p>",
"privacyUrl": "https://zoom.us/privacy",
"supportUrl": "https://support.zoom.us/hc/en-us/articles/201362003",
"supportEmail": null,
"configUrl": null,
```


Fig. 6: Detailed App Metadata showing URL of the privacy policy

to either each app's data practices or its privacy policies. In the second phase, we iterate over every 'id' obtained in phase one and query the *"App Details"* endpoint. This call yields the full set of OAuth scopes and the privacy policy URL for that app. By chaining these two retrieval operations for each app, our crawler collected metadata for 2,733 applications on Zoom marketplace.

For the privacy policies which we found to be reachable, we implemented a two-step process to retrieve these for further analysis: We first used a Python script which iterated over the list of privacy-policy URLs. This script automated the use of the Microsoft Edge browser to download each reachable policy. We then used Andow et al.'s HTMLToPlaintext [26] code to make each policy suitable for NLP analysis.

C. Testing Privacy-Policy URLs

With the complete set of privacy-policy URLs (example shown in Figure 6) from our last phase, our next task is to verify that each link actually resolves to a functional page. We issue an HTTPS GET request [27] to every URL with a 10-second timeout and record the HTTP status code. We then classify each response as follows: (i) Responses in the 200-299 range are marked as *"reachable,"* (ii) 4xx and 5xx codes denote client or server-side failures both marked as *"broken"*, and any connection or DNS error likewise denotes a broken

TABLE I: Examples of Contradictions

Data Practice	Type	Contradicting Practice (in same policy)
We will ***never sell or share*** your contact data with any third party except **for ... *proper functionality of the app.***	C1	Specifically, we ***may share*** the Contact information with the company our Users work with or for in order ***to improve the services*** we offer.
... we ***may share Personal Information*** with third parties (including agents and sub-contractors) assisting us to provide information, products, services or direct marketing to you.	C2	We do ***not sell information*** about you to advertisers or other third parties.
Our ***customers and your teachers may use personal Information about you*** to improve their marketing and promotional efforts...	C3	As a matter of policy, we do ***not sell or rent any personally identifiable information about you*** to any third party.
... has ***disclosed the following categories of personal information to third parties*** for a business or commercial purpose in the preceding twelve months	C4	... has ***not sold any personal information to third parties*** for a business or commercial purpose in the preceding twelve months.
... has ***disclosed the following categories of personal information*** to third parties for a business or commercial purpose in the preceding twelve months ...	C5	... will ***not sell personal information in the future*** belonging to website visitors, users and other consumers... has ***not sold any personal information to third parties*** for a business or commercial purpose in the preceding twelve months.

TABLE II: Examples of Narrowing Definitions

Data Practice	Type	Narrowing Statement (in same policy)
We ***may disclose personal information*** to service providers.	N1	Your ***emails, travel history, and travel data are NOT shared*** with any third parties, including OpenAI
We can (and you authorize us to) ***disclose any information about you to law enforcement and other government officials*** ...	N2	As a matter of policy, we do ***not sell or rent any personally identifiable information*** about you to any third party.
These ***third parties may use your personal information only as directed or authorized*** by us and in a manner consistent with this Privacy Policy, and are prohibited from using or disclosing your information for any other purpose.	N3	To the extent we share personal information with our business customers, this Privacy Policy does ***not apply to our business customers'*** subsequent use of that information.
... provides this Privacy Statement (the "Statement") to help you understand how ... ***collects and uses the Personal Information and Non-Personal Information*** (each as defined below) ...	N4	Also, if you provide Personal Information to ... through other means, such as when you call customer service or send Personal Information related to promotions or product registrations via postal mail, this Statement does ***not apply*** to that information.

link. Each URL's final status and code are logged for further analysis. Using these criteria, we were able to collect 2,455 privacy policies out of 2,733.

D. Internal Consistency Checking

Legal notices from privacy policies can unintentionally—or deliberately—hide critical information through contradictory statements or overly narrow qualifiers. To evaluate clarity and logical coherence of these polices, we employed PolicyLint tool [28] [26], a static analysis tool designed to flag two classes of semantic anomalies. First, it detects contradictions, where a privacy policy makes directly opposing claims. For example, one sentence might say *"we may share your data,"* while another says *"(we will not share your data.)"*. Second, it finds narrowing definitions, in which a broad statement *("we collect usage data")* is immediately undercut by a restrictive qualifier *("but only for service improvement")*, making the overall meaning unclear to the users. PolicyLint further divides these anomalies into detailed subcategories (C1–C5 for contradictions and N1–N4 for narrowing definitions), whose definitions and examples are provided in Appendix VIII-A.

We processed all 2,455 retrievable plaintext privacy policies using PolicyLint's built-in rules for contradictions (C1–C5) and narrowing definitions (N1–N4). For each document, we recorded the sentences flagged under each subtype and their raw occurrence counts. Example sentences illustrating each subtype from the retrieved privacy policies are shown in Table I and Table II.

E. Data Practice Analysis

To quantify gaps between the permissions granted to each Zoom Marketplace app and the practices it discloses in its privacy policy, we adapt the data-practice extraction pipeline of Liao *et al.* [29]. In their original design, the authors compared voice applications' descriptions against their policies in order to find data practices mentioned in the descriptions that are omitted from the corresponding privacy notices. In our case, instead of using app descriptions, we used each app's *OAuth scope name* and *developer-provided payload description* as the input text and compared the extracted data practices with those disclosed in the app's privacy policy.

Using this adapted pipeline, we first derive the set of implied practices for each app by processing its metadata: every OAuth scope *name* string tokenized, lowercased, and lemmatized and payload description is similarly normalized yielding a precise, machine-readable summary of the privileges Zoom has technically granted to the app. In parallel, we apply the same normalization and mapping steps to the full plain-text of each app's privacy policy: we tokenize and lemmatize the text, then use named-entity recognition and dependency-based pattern matching to identify all references to those same data practices in the privacy policy. This enable us to map which privileges implied by the OAuth scopes are actually disclosed in the privacy policy. Therefore, by computing the set difference between the implied-practice set (from the privileges) and the declared-practice set (from the policy) for each app, we isolate those data practices that Zoom has authorized but the developer fails to disclose.

F. OAuth Enforcement Validation

To determine whether Zoom enforces server-side OAuth verification before loading and executing a third-party app, we developed a focused proof-of-concept test. Zoom's OAuth framework requires each app to present a valid `ZM_CLIENT_ID` and `ZM_CLIENT_SECRET`—a credential pair generated when the developer registers the app in the Zoom Developer Console. These values uniquely identify and authenticate the app during the OAuth handshake. To test whether the Zoom client enforces this credential check at runtime, we registered a new app and then replaced both

the generated client ID and secret with arbitrary placeholder strings, ensuring that any OAuth request would fail. We hosted the app's launch on a local Node.js/Express server (Webview) [30], and—because Zoom's client can only load apps over HTTPS—we used Ngrok [16] (a secure tunneling service that exposes local servers at public HTTPS URLs) to map our localhost to an externally reachable address.

We then registered our public Ngrok URL as the app's OAuth redirect URI in the Zoom Developer Console. In our Express server, we implemented a handler at `'/oauth/callback'` that logs every incoming request—capturing query parameters like `'error=invalid_client'` and records the HTTP error response sent back by Zoom's authorization endpoint. This setup let us observe, in a controlled and auditable way, whether the client still fetches and executes the app's JavaScript bundle despite the invalid credentials.

IV. Results & Findings

A. Privacy Policy URL Analysis

We began our evaluation by attempting to retrieve the privacy-policy URL for each of the 2,733 apps in our Zoom Marketplace corpus by issuing an HTTPS GET request (10s timeout) and recorded the HTTP status code. To understand which response codes reliably indicated a working policy, we manually validated samples from each code range. Of 128 URLs returning a 2xx status, 93% actually served a full privacy policy. Those with response codes 400 and 401 did not return a privacy policy (based on all responses) and those with response code 404 (not found) only returned a privacy policy 4% of the time. URLs with a response code of 403 (forbidden) were found to return a privacy policy 91% of the time when queried with the Microsoft Edge browser (based on a sample of 110). Response codes in the 500-599 range only returned privacy policies 8.7% of the time. We found that response codes 400, 401, 404 and any in the range of 500-599 were the best indicators of an unreachable privacy policy. Based on these measurements, we found that 87% of Zoom apps had working privacy policy links. A breakdown of these results is plotted in Figure 7. It is important to note that, some privacy policy URLs returned error status codes (e.g., 401 or 403) but were still accessible in a browser due to server configurations or misconfigurations.

B. Detecting Inconsistencies in Privacy Policies

Building on our methodology from Section III-D, we ran PolicyLint over the 2,455 retrievable, English-language privacy policies. Among these, PolicyLint found a total of 1,189 contradictions and narrowing definitions in 345 of these. This indicates that 14% of available privacy policies contained these issues that could impair the user from understanding the privacy practices.

Our review results showed that type C1 contradictions were reported at a significantly higher rate than other types. This because C1 by definition, lags every pair of directly opposing statements within a policy, even if those statements are the

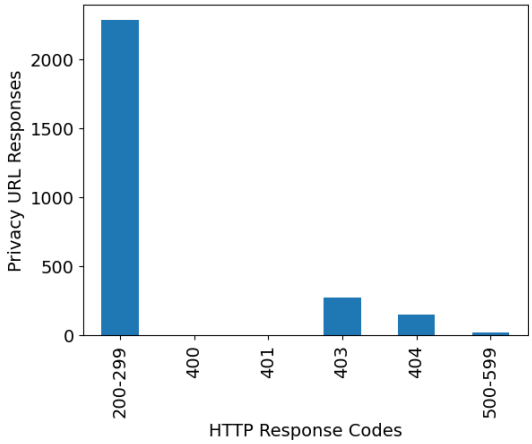

Fig. 7: Responses to Privacy Policy URLs

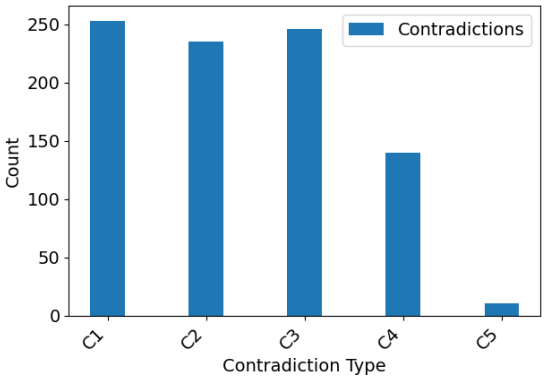

Fig. 8: Counts of Contradictions, by Type

result of ambiguous, boilerplate, or poorly structured language rather than true logical opposition. For example, some policies broadly state *"We will never share your contact data with any third party,"* yet elsewhere say, *"We may share contact information with service providers to improve our services."*. Although these may refer to different contexts or include implicit exceptions, the tool flags them as direct contradictions (C1). As a result, PolicyLint reported roughly twice as many instances of type C1 contradictions as were confirmed by manual review. For C1 contradictions specifically, we observed 253 flagged cases. However, upon closer manual inspection, many of these could be resolved by understanding the context or intent behind each statement. Most of these logical contradictions and cases of narrowing could be clarified through a close reading of the privacy policy. Similarly we observed 246 instances of type C3, 235 instances of type C2, 140 instances of type C4, and 11 instances of type C5, the count of these contradictions are shown in Figure 8. To understand the significance of these direct contradictions, examples are given in Table I.

PolicyLint also identified instances of all four types of narrowing definitions: the second kind of inconsistency that PolicyLint detected. Narrowing definitions of type N1 were the most common (155), while 123 instances of type N2, 13

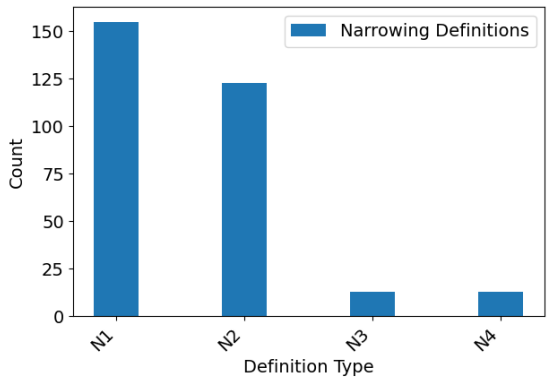

Fig. 9: Counts of Narrowing Definitions by Type

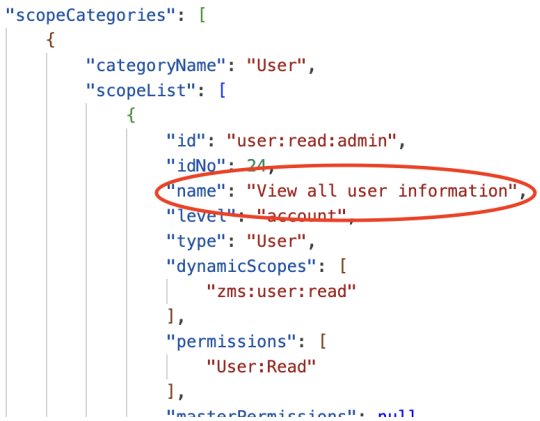

Fig. 10: Example scope name, in public metadata

instances of type N3, and 13 instances of type N4 were also observed. These counts are shown in Figure 9. Some telling examples of these are given in Table II.

C. Data Practice Extraction and Inconsistency Checking

As shown in Figure 10, we found that each app's permitted scopes were listed in its metadata, along with a short description of the data practice that was permitted by the scope. In this example, we see that this scope permits this app to "View all user information."

We further found detailed descriptions of what practices were specifically permitted by these scopes when allowed for the corresponding app. We found that these descriptions were presented to users prior to the user granting approval to install the app. We called these payload descriptions, as shown in the example in Figure 11.

We combined both of these fields: scope names and payload descriptions, to form lists of the data practices that were associated with each app. We used the NLP-based method presented in [29] to identify data practices that were disclosed in each app's public metadata, but were not addressed by the corresponding privacy policy.

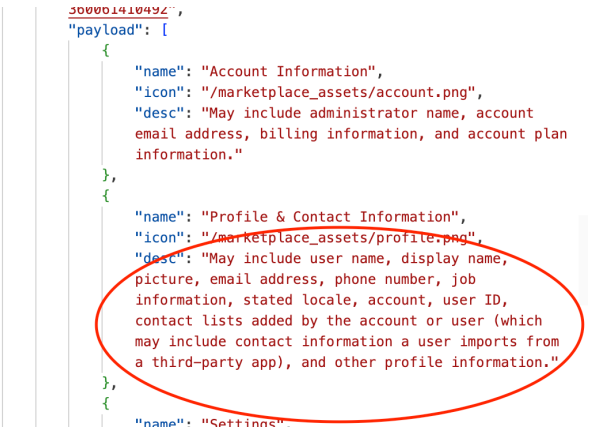

Fig. 11: Example payload description, in public metadata

We were able to analyze 2,447 of our collected privacy policies using our method described in Section III-E . All but 7 (99.7%) of these policies were missing at least one data practice that their OAuth scopes implied. In total, we detected 19,007 instances of missing practices across the corpus, yet these fell into only 26 of these were unique. Figure 12 visualizes the frequency distribution of these 26 unique omissions. When identifying data practices, our code phrased each of these beginning with the words "which include ...," as shown in Table III, which lists the 12 most common missing-practice phrases, illustrating how a small set of undeclared capabilities recurs across a large fraction of apps.

The most commonly overlooked data practice (over 2,142 times) was *"which include contact information."* This means that, in all of these cases, at least one scope permitting access to contact information was granted to an app, but this data practice was not covered by the app's privacy policy. These findings underscore a systemic gap between the privileges Zoom grants via OAuth scopes and the corresponding disclosures in developers' privacy policies, suggesting the need for more rigorous policy-scope alignment checks during the app–approval process.

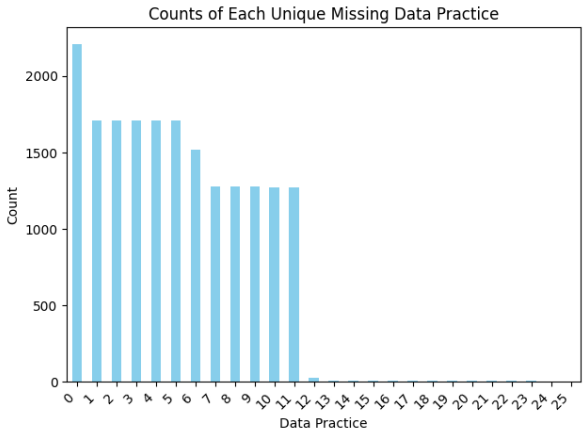

Fig. 12: Counts of Each Unique Missing Data Practice

TABLE III: Top 12 Most Overlooked Data Practices

Data Practice	Missing Instances
which include contact information.	2142
which include video messages meeting name agenda transcriptions.	1751
which include video messages meeting name agenda.	1751
which include video messages meeting name agenda transcriptions responses to polls.	1751
which include video messages meeting chat name.	1751
which include video messages meeting name agenda transcriptions responses.	1750
which include video messages transcriptions feedback responses to polls Q&A files context as details meeting chat name.	1540
which include name responses other registration information.	1307
which include name responses to registration questions.	1307
which include name responses.	1300
which include name contact information.	1281
which include name.	1256

```
# Client ID for your Zoom App
ZM_CLIENT_ID=blah

# Client Secret for your Zoom app
ZM_CLIENT_SECRET=blah

# Redirect URI set for your app in the Zoom Marketplace
ZM_REDIRECT_URL=https://706b1799a371.ngrok-free.app

# App Name used for isolating logs
APP_NAME=${_APP_NAME}

# Key used Sign Session Cookies
SESSION_SECRET=${_SESSION_SECRET}
~
```

Fig. 13: Invalid App Authentication Parameters

This led us to believe that developers thought that they did not have to address protections and disclosures of user contact information in their privacy policies. The remaining 10 most commonly overlooked data practices were either related to contact information or meeting information. Again, based on the very high frequencies at which these practices were found to be missing from policies, we asserted that developers did not assume that these kinds of practices were relevant to their privacy policies, as shown in Table III.

D. Potential App Authentication Vulnerability

To evaluate the integrity of Zoom's app execution environment, we conducted a targeted experiment simulating a compromised or malicious app deployment. Our study found that we could execute our custom app's code inside our Zoom client without knowledge of either the app's `ZM_CLIENT_ID` or its `ZM_CLIENT_SECRET`. In this case, we set the value for both of these to "blah," as shown in Figure 13. Once we set these values to "blah", our app's HTTP server generated an internal error due to its inability to authenticate with zoom. Our app was still loaded by our Zoom client, as shown in Figure 14.

To confirm this vulnerability, we modified our app's code to bypass cryptographic verification of the Client ID and Client

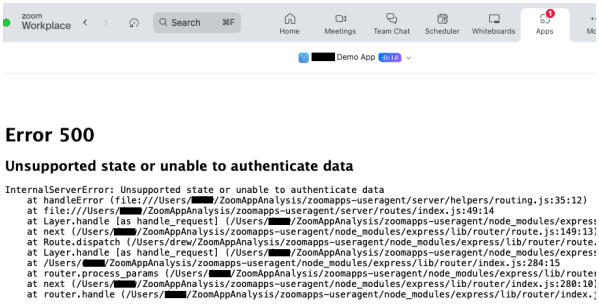

Fig. 14: Our App with Bad Credentials in Zoom

```
import express from 'express';
import { handleError, sanitize } from '../helpers/routing.js';
import { contextHeader, getAppContext } from '../helpers/cipher.js';
import { getInstallURL } from '../helpers/zoom-api.js';
import session from '../session.js';

const router = express.Router();

/*
 * Home Page - Zoom App Launch handler
 * this route is used when a user navigates to the deep link
 */
function isContextExpired(context) {
    const currentTime = Date.now();
    return context.exp && context.exp < currentTime;
}

router.get('/', async (req, res, next) => {
    try {
        sanitize(req);

        const header = req.header(contextHeader);

        // Disable Zoom Authentication Checking
        // const context = header && getAppContext(header);

        // Disable Zoom Authentication Checking
        // Force context = true
        const context = true

        if (!context) {
```

Fig. 15: Cipher Verification Bypassed in Code

Secret, as shown in the code snippet in Figure 15. Once this modification was made, we reloaded our app in the Zoom client. In this case, it correctly executed, returning the user agent value of the Zoom client, as shown in Figure 16. This raises important security concerns for the user, as the contents of a third-party app could easily be corrupted or replaced by a threat since these apps are not required to authenticate with Zoom in order for their code to execute within the context of the user's Zoom client.

This vulnerability has significant real-world security implications. We model this threat as an attacker that uses a malicious app (either one that they have developed, or a safe one that they have compromised) to exploit data practices allowed by Zoom-granted privileges. Since Zoom does not revalidate apps during runtime or monitor changes to the hosted code [20], [31], their assessment of app security is limited to what they can test externally and what can be assessed based on the developer-provided documentation, metadata, attestations, and other evidence.

Such a threat could potentially leverage the permissions

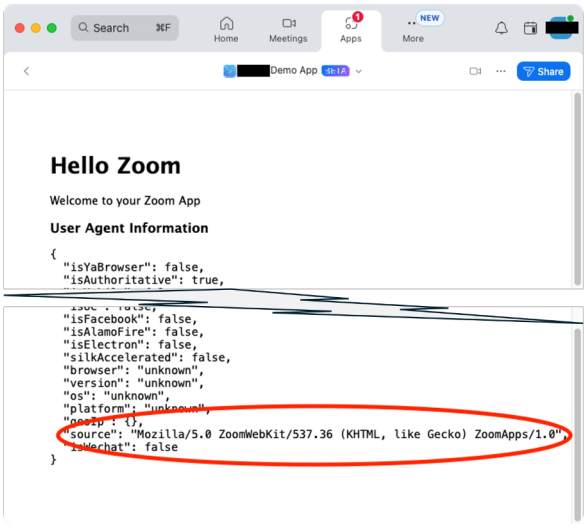

Fig. 16: Our App with Verification Bypassed in Zoom

granted to the app in order to get or change information about the Zoom user or their client. This information could include personal, private information [32]. This kind of threat could also potentially access meeting recordings and other sensitive information, by simply accessing the data through the permissions granted to the app by Zoom [33].

V. Discussion

A. Limitations

One of our original goals was to evaluate the code of Zoom marketplace apps to determine which scopes were actually used versus those that were simply requested. By comparing the used scopes to the list of allowed scopes in each app's metadata, we hoped to identify cases of over-privilege—where unused scopes are still granted. However, since we could not access the source code for these apps, we were unable to make this determination. While Zoom's policies require app developers to remove any unused scopes, there is no independent verification of whether developers actually comply with this rule. As neither Zoom nor our team reviewed the underlying source code, we could not conclusively determine the extent to which these apps have excessive privileges.

B. Privacy Implications

Based on this work, we concluded that users of Zoom Marketplace Apps should be mindful of these apps' impacts on their personal privacy. While Zoom has policies in place that require disclosure of these apps' data practices, we found that the company has little way of enforcing these policies, short of continuously monitoring the developers and publishers of these third-party apps.

VI. Related Work

In recent years, there has been an increased focus the privacy of third-party app and add-ins available for popular user devices and platforms [34], [26], [35], [29], [36], [37], [38], [39]. Much of the prior work focuses add-ins that are available for Android, Alexa [40], and other hardware-based platforms.

More recently, Liu and Biczók [9] proposed a method of managing interdependent privacy issues such as sharing other users' contact and calendar information without their permission, across third-party apps. Their work is relevant because they explored the privacy issues that arise when third-party apps are used to integrate users and services across multiple platforms. They considered the privacy implications of apps for Android, but also considered those of extensions for the Firefox, Opera web browsers and the Zoom marketplace. Their work did not examine the contents of Marketplace apps' policies, which we offer in this work.

Goenka at al. [41] studied trends in Zoom's marketplace apps over the period of a year. Their work documented trends in the amount of published apps, and the amount and kinds of permissions that were granted. They also identified concerns with the amount of user data that was collected by these apps, along with possible non-compliance with applicable regulations. However, they did not perform fine-grained mapping of OAuth scope–implied data practices against privacy policy text. Also, similar to Liu and Biczók [9], they also did not analyze the contents of Marketplace apps' privacy policies.

The "Laws of Zoom" [42] is particularly relevant to us, as it presents an in-depth study of the API that is used by third-party developers to access Zoom's functions from their marketplace apps. It highlights the many types of information that may (or may not) be accessed by these apps. They also make a case for the "closed" nature of the Zoom application programming interface (API).

Kagan et al. [43] explore privacy concerns that arise due to images of Zoom meetings being shared, posted or leaked online. As much of Zoom's meeting metadata is available to marketplace apps by way of the API [42], then understanding the privacy and security concerns implied by the relationship of this metadata to the meeting images. Nevertheless, this work studies privacy leakage via participant images in meeting collage snapshots, not third-party app permissions or policies.

Liao et al. [29] is particularly relevant to our work. They sought to measure the effectiveness of privacy policies for voice assistant applications, such as *skills* for Amazon's Alexa and *actions* for Google's Assistant. Like us, they were unable to obtain the source code of the apps that were the subject of their study. They chose to compare the data practices evident in each *skill*'s or *action*'s descriptions to those that were found in the corresponding privacy policies (if any policy existed). In our case, we were able to gather the data practices that are permitted by each Zoom marketplace app. We then used Liao et al. [29]'s approach to confirm each practice's inclusion in its app's associated privacy policy. Our work further validates the efficacy of Liao et al.'s methods, while also demonstrating their general applicability beyond the policies of voice assistant applications.

Given our focus on the privacy policies themselves, we also

considered the previous work of Andow et al. [26]. Their work identified common types of issues that occur within privacy policies. In their work, they defined these issues as *contradictions* and *narrowing definitions*. They defined *contradictions* as instances when a policy states that the app or developer performs a given data practice, but then the policy elsewhere states that the data practice is not performed (or the opposite). In cases of *narrowing definitions*, they defined these as instances where a data practice is broadly stated in one case, but then partially contradicted elsewhere (or the opposite). They studied the privacy polices of applications that were published on the Google Play store. In our case, we chose to use their approach to identify the same types of issues in the policies associated with Zoom marketplace apps. Similar to Liao et al., our work further validated and demonstrated the general applicability of PolicyLint beyond its original use case.

While our approach builds upon established methodologies, our work addresses a fundamentally different architectural context and research scope than prior studies. PolicyLint [26] focuses solely on textual static analysis of internal contradictions within Android app privacy policies from Google Play, without considering the actual permissions granted to apps. Similarly, Liao et al. [29] examined voice assistant applications (Alexa skills and Google Actions) by comparing voice applications' descriptions against privacy policies, however lacking access to formal permission structures such as OAuth scopes on Zoom environment. In contrast, our study audits 2,733 Zoom Apps, combining static policy text analysis with OAuth scope–policy comparison and a proof-of-concept runtime check of OAuth credential enforcement in the Zoom client.

VII. Conclusion

In this study, we conducted a comprehensive privacy compliance analysis of 2,733 Zoom Marketplace apps by systematically examining the alignment between their granted OAuth permissions and privacy policy disclosures. We found that while 87% of apps provided accessible privacy policies with only 14% containing contradictions, a staggering 99.7% failed to disclose all data practices permitted by their OAuth scopes, with contact information access being the most frequently overlooked practice. Additionally, we demonstrated that Zoom apps can execute within the client even with invalid authentication credentials, potentially allowing malicious actors to exploit previously granted OAuth scopes after the app's approval. These findings show a disconnect between Zoom app permissions and developer disclosures, calling for stricter approval checks and better user privacy awareness.

Acknowledgment

This work is supported by the National Science Foundation (NSF) under Grant No. 2523814, 2448205, 2239605, 2228616, the South Carolina Research Authority, and partially based upon the work supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR) (a U.S. Department of Transportation National University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR, and the U.S. Government assumes no liability for the contents or use thereof.

References

[1] Xueqian Cao. Analyzing the popularity of zoom video communications using marketing mix theory. page 301–305. Atlantis Press, December 2021.

[2] Axel Volmar, Charline Kindervater, Sebastian Randerath, and Aikaterini Mniestri. *Mainstreaming Zoom: Covid-19, Social Distancing, and the Rise of Video-Mediated Remote Cooperation*, page 99–133. Springer Fachmedien, Wiesbaden, 2023.

[3] Zoom User Surge. https://www.cnbc.com/2020/04/03/how-zoom-rose-to-the-top-during-the-coronavirus-pandemic.html.

[4] Zoom sued for overstating, not disclosing privacy, security flaws. *Reuters*, April 2020.

[5] Zoom common criteria certification. https://explore.zoom.us/en/trust/legal-compliance/common-criteria/.

[6] Zoom Marketplace Launch. https://medium.com/zoom-developer-blog/official-launch-of-the-zoom-marketplace-df24dabd3e4f.

[7] Zoom Marketplace. https://marketplace.zoom.us/.

[8] Zoom agrees to step up security after new york probe. https://www.securityweek.com/zoom-agrees-step-security-after-new-york-probe/, May 2020.

[9] Shuaishuai Liu and Gergely Biczók. Idpfilter: Mitigating interdependent privacy issues in third-party apps. (arXiv:2405.01411), may 2024. arXiv:2405.01411.

[10] Katharina Sophie Dassel and Stefan Klein. To zoom or not: Diverging responses to privacy and security risks. *Journal of Business Research*, 161:113772, June 2023.

[11] Dima Kagan, Galit Fuhrmann Alpert, and Michael Fire. Zooming into video conferencing privacy. *IEEE Transactions on Computational Social Systems*, 11(1):933–944, February 2024.

[12] Wenhong Chen and Yuan Zou. Why zoom is not doomed yet: Privacy and security crisis response in the covid-19 pandemic. *American Behavioral Scientist*, page 00027642231155367, February 2023.

[13] Carlin Sack. Introducing zoom apps: Use the apps you love, right in zoom.

[14] We now have more than 1,000 apps on the zoom app marketplace to help you work better. https://www.zoom.com/en/blog/we-now-have-more-than-1000-apps-on-the-zoom-app-marketplace/.

[15] App Marketplace. Zoom app marketplace.

[16] Setting up your zoom app. https://dev.to/zoom/setting-up-your-zoom-app-4h35, June 2022.

[17] Zoom apps sdk. https://appssdk.zoom.us/classes/ZoomSdk.ZoomSdk.html.

[18] Oauth scopes (overview). https://developers.zoom.us/docs/integrations/oauth-scopes-overview/.

[19] Inc. Zoom Video Communications. Zoom app marketplace app review process.

[20] Submission checklist - zoom developers. https://developers.zoom.us/docs/distribute/app-submission/submission-checklist/.

[21] Yunang Chen, Yue Gao, Nick Ceccio, Rahul Chatterjee, Kassem Fawaz, and Earlence Fernandes. Experimental security analysis of the app model in business collaboration platforms. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2011–2028, 2022.

[22] Kwaku Nyante. Domain validation explanation.

[23] Saharsh Goenka, Adit Prabhu, Payge Sakurai, Mrinaal Ramachandran, and Rakibul Hasan. Who's watching you zoom? investigating privacy of third-party zoom apps. *arXiv preprint arXiv:2504.04388*, 2025.

[24] Zoom marketplace api. https://developers.zoom.us/docs/api/rest/reference/marketplace/methods/#overview.

[25] https://requests.readthedocs.io.

[26] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. {PolicyLint}: Investigating internal privacy policy contradictions on google play. In *28th USENIX security symposium (USENIX security 19)*, pages 585–602, 2019.

[27] Roy T. Fielding and Julian Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Number RFC 7231. June 2014.
[28] Privacy policy analysis. https://github.com/benandow/PrivacyPolicyAnalysis?tab=readme-ov-file.
[29] Song Liao, Christin Wilson, Long Cheng, Hongxin Hu, and Huixing Deng. Measuring the effectiveness of privacy policies for voice assistant applications. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 856–869, 2020.
[30] OpenJS Foundation and Contributors. Express — fast, unopinionated, minimalist web framework for node.js. https://expressjs.com/. Accessed: 2025-07-30.
[31] Security best practices - zoom developers. https://developers.zoom.us/docs/distribute/security-best-practices/.
[32] Zoom account api. https://developers.zoom.us/docs/api/rest/reference/account/methods/#operation/accountSettingsUpdate.
[33] Integrations granular scopes zoom developers. https://developers.zoom.us/docs/integrations/oauth-scopes-granular/.
[34] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. Dangerous skills got certified: Measuring the trustworthiness of skill certification in voice personal assistant platforms. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.
[35] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. Actions speak louder than words:{Entity-Sensitive} privacy policy and data flow analysis with {PoliCheck}. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 985–1002, 2020.
[36] Le Yu, Xiapu Luo, Jiachi Chen, Hao Zhou, Tao Zhang, Henry Chang, and Hareton KN Leung. Ppchecker: Towards accessing the trustworthiness of android apps' privacy policies. *IEEE Transactions on Software Engineering*, 47(2):221–242, 2018.
[37] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven Bellovin, and Joel Reidenberg. Automated analysis of privacy requirements for mobile apps. In *2016 AAAI Fall Symposium Series*, 2016.
[38] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering*, pages 25–36, 2016.
[39] HHK Fawaz, RLF Schaub, and KGS Karl. Polisis: automated analysis and presentation of privacy policies using deep learning. Technical report, Technical report, EPFL, 2017.
[40] Mohammed Aldeen, Jeffrey Young, Song Liao, Tsu-Yao Chang, Long Cheng, Haipeng Cai, Xiapu Luo, and Hongxin Hu. End-users know best: Identifying undesired behavior of alexa skills through user review analysis. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(3):1–28, 2024.
[41] Saharsh Goenka, Adit Prabhu, Payge Sakurai, Mrinaal Ramachandran, and Rakibul Hasan. Who's watching you zoom? investigating privacy of third-party zoom apps. (arXiv:2504.04388), April 2025. arXiv:2504.04388 [cs].
[42] Kim Albrecht. Laws of zoom. *Digital Society Volume 53*, page 136.
[43] Dima Kagan, Galit Fuhrmann Alpert, and Michael Fire. Zooming Into Video Conferencing Privacy. *IEEE Transactions on Computational Social Systems*, 11(1):933–944, February 2024. Conference Name: IEEE Transactions on Computational Social Systems.

VIII. Appendix

A. PolicyLint Semantic Taxonomy

We employ PolicyLint which flags two principal classes of semantic anomalies—*contradictions* and *narrowing definitions*—each subdivided into specific subtypes. Tables I and II illustrate these subtypes with concrete examples from our retrieved polices. Below, we summarize the meaning of each label:

a) Contradiction Subtypes (C1–C5): A contradiction occurs when a policy both permits and forbids the same or related data practice in different clauses:

- **C1 (Exact Contradiction):** The same data item and subject appear in opposing statements. *E.g.*, "we will never share your contact data" vs. "we may share contact information."
- **C2 (Data-Subsumption Contradiction):** The same subject appears with a broad data category and a narrower, forbidden data subtype. *E.g.*, "we may collect personal information" vs. "we do not collect email addresses."
- **C3 (Entity-Subsumption Contradiction):** A broad entity and a subsumed (narrower) entity appear with opposing permissions on the same data item. *E.g.*, "we disclose email addresses" vs. "our partners do not collect email addresses."
- **C4 (Both-Subsumption Contradiction):** Both the subject and data category differ by subsumption and are used in opposing clauses. *E.g.*, "we share personal information with partners" vs. "third parties may not use personal information."
- **C5 (Cross-Generalization Contradiction):** Non-overlapping subjects and data categories are used in contradictory statements. *E.g.*, "advertisers may collect usage data" vs. "we will not share personal information."

b) Narrowing-Definition Subtypes (N1–N4): A narrowing definition arises when an initially broad permission is immediately qualified by a restrictive clause:

- **N1 (Exact-Category Narrowing):** The same subject appears with a broad allowance and a disallowed subtype. *E.g.*, "we may disclose personal information" vs. "we do not share email addresses."
- **N2 (Data-Subtype Narrowing):** A broad data category is allowed by one subject and a specific subtype is forbidden by a different subject. *E.g.*, "we collect personal information" vs. "our agents do not collect email addresses."
- **N3 (Entity-Subtype Narrowing):** A specific data item is allowed by a broad subject and forbidden by a subsumed subject. *E.g.*, "we share email addresses" vs. "third parties may not use email addresses."
- **N4 (Both-Subtype Narrowing):** Both subject and data category differ by subtype in adjacent permission and restriction. *E.g.*, "we use personal information" vs. "we do not use email addresses."

By grounding each inconsistency flag in this taxonomy, readers can readily interpret PolicyLint's output (Sections III-D) and understand precisely which patterns of semantic obfuscation are present in a given policy.