Understanding Users' Perceptions of Client-Side Modifications and Anti-Cheat Engines in Social XR

Iman Mohammadi*
University of California, Irvine
Irvine, California, USA
imohamm1@uci.edu

Mutahar Ali*
University of California, Irvine
Irvine, California, USA
mutahara@uci.edu

Habiba Farrukh
University of California, Irvine
Irvine, California, USA
habibaf@uci.edu

Abstract

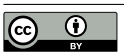
Social XR platforms allow users to interact in immersive, shared virtual environments. To support real-time interaction and immersion, these platforms require each user's device (client) to maintain the shared environment state to render it. Prior work has shown that this reliance on client-side state enables malicious users to modify their clients in ways that undermine other users' security and privacy. Platforms respond with defenses such as server-side validation, integrity attestation, and anti-cheat engines, but these mechanisms can themselves introduce privacy, security, usability, and accessibility concerns. Despite these tradeoffs, how users experience harms from client modifications or perceive the defenses intended to prevent them remains largely unexplored.

To address this gap, we conduct a large-scale analysis of ~5M Reddit posts from four popular XR-related subreddits. Through qualitative analysis, we find that users report a range of concerns about malicious modifications, including disruption, asset theft, and evasion of safety features to harass or stalk other users. Unlike on other platforms, users perceive these harms as violations of their sense of presence and personal identity because of the embodied and immersive nature of social XR. Users' attitudes toward anti-cheat engines are divided, with some crediting them for reducing harm, while others raise concerns about performance degradation, privacy implications, false positives, and the loss of beneficial modifications, such as accessibility tools. Using automated labeling and regression analysis, we find that user concerns shift in response to specific platform events (e.g., anti-cheat updates, public bypasses), with anti-cheat deployments coinciding with significant spikes in discussions of their effectiveness and tradeoffs. Based on our findings, we synthesize recommendations for platforms, users, and policymakers to improve safety and privacy in social XR while preserving user trust, accessibility, and transparency.

CCS Concepts

• **Security and privacy** → **Human and societal aspects of security and privacy**; *Software and application security*; • **Human-centered computing** → *Virtual reality*.

*These authors contributed equally to this work.



This work is licensed under a Creative Commons Attribution 4.0 International License.
CCS '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2871-6/2026/11
https://doi.org/10.1145/3830454.3846683

Keywords

social XR, extended reality, client modifications, anti-cheat engines, security and privacy perceptions

ACM Reference Format:
Iman Mohammadi, Mutahar Ali, and Habiba Farrukh. 2026. Understanding Users' Perceptions of Client-Side Modifications and Anti-Cheat Engines in Social XR. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3830454.3846683

1 Introduction

Social Extended Reality (XR) platforms, such as VRChat [82] and Rec Room [61], let users meet and interact in shared, immersive virtual environments through embodied avatars [29, 71]. They have become mainstream venues for socializing, live events, education, and professional training, with VRChat hosting tens of thousands of concurrent users and Rec Room reporting millions of monthly active users [35, 74]. To meet the latency these interactions demand [22], each user's device, the client, holds and renders a local copy of the shared space, including the world, the objects in it, and the avatars of everyone present [3]. This places the state that governs a user's experience on a machine the platform does not control, opening the door to *client-side modifications* (mods), unauthorized changes a user makes to the software running on their own device [87].

Client modifications are a long-standing threat in online games, where they enable cheating and disruption [34], and in mobile and financial applications, where they bypass licensing or access memory [38, 75]. Recent work has demonstrated them in social XR as well, showing how modified clients can manipulate the shared state they hold to surveil other users and bypass safety features [3, 47]. A substantial body of work examines security and privacy in social XR, including attacks on sensor data and shared state [51, 52, 72, 73, 76], harassment and embodied harms users experience [8, 30, 68], privacy expectations [1, 33, 67], and the safety features platforms provide [13–15, 84]. However, how users in social XR experience client modifications remains unstudied. This matters because the embodied nature of social XR can amplify harms from modified clients, as behaviors such as virtual stalking and groping trigger psychological reactions comparable to physical-world encounters [8, 68].

Several defenses against client modifications exist, including server-side validation of client actions, integrity attestation that verifies the client binary has not been tampered with, and anti-cheat (AC) engines that inspect the running client for signs of tampering [18]. Social XR platforms have adopted these defenses, including AC engines [60, 80]. Prior work on AC engines has studied how they work, how quickly attackers find ways around them,

and what level of system access they have [18, 21, 42, 78]. How effective users find these defenses against client modifications, and what concerns they hold about running them, remain unexplored. Understanding this is important for social XR, where these engines protect users' personal safety, not only fair play as in online games.

To address this gap, we ask the following research questions:

- **RQ1:** What concerns do users articulate about client modifications in social XR?
- **RQ2:** What are users' perceptions of the platform defenses deployed against client modifications?
- **RQ3:** How do users' concerns and perceptions evolve over time, and how do major events in the social XR ecosystem influence them?

To answer these questions, we conduct a large-scale analysis of ∼5M Reddit posts from four popular XR-related subreddits (`r/VRChat`, `r/RecRoom`, `r/virtualreality`, `r/VRGaming`). Our analysis proceeds in four stages. Using keyword filtering and stratified sampling, we manually annotate 1,250 posts to fine-tune a DeBERTa-based binary classifier [36] that identifies content related to client modifications and platform defenses. Applying this relevance classifier to the full corpus yields ∼ 71K related posts. We then iteratively sample and annotate a subset of these posts through qualitative thematic analysis to develop a codebook capturing users' concerns and perceptions about client modifications and defenses. We use this codebook with an LLM-based multi-label classifier to label all ∼ 71K posts, and finally conduct an interrupted time-series regression analysis [7] to investigate how users' concerns shift in response to major events in the social XR ecosystem over time.

We find that the embodied and immersive nature of social XR changes how harms are experienced, with users articulating a wide range of concerns about how client modifications can be weaponized against other users, the platform, and content creators. The most prevalent is disruption of users' experiences (48.1% of labeled posts), ranging from full instance crashes that remove users from live events to object manipulation that induces lag. This is followed by asset theft (26.6%), which users describe as a violation of personal identity, since an avatar is their persistent face and body in a virtual space. Users also complain that modified clients allow malicious users to bypass built-in safety features (2.9%), such as block and votekick, enabling persistent stalking and harassment. Users who install modifications themselves, meanwhile, often fall victim to credential-stealing payloads bundled with the mods (3.1%).

Users' attitudes toward defenses, mainly AC engines, are deeply mixed. A subset (9.7%) credits the AC engines for reducing harms after deployment, while a larger share (25.3%) finds them ineffective against determined modders and reports that they break the app, degrade performance (3.6%), and remove mods users rely on, including accessibility controls (8.8%). Our longitudinal analysis further identifies 10 events significantly associated with shifts in this discourse, showing that concerns about modification harms persist across them and attitudes toward AC engines change with each deployment, public bypass, and update.

Our study extends efforts to understand users' security, privacy, and safety perceptions in immersive social platforms, and highlights how these perceptions shift in response to both platform defense decisions and the modding community's responses to them. Our findings help platforms weigh usability, privacy, and effectiveness tradeoffs when deploying defenses, and anticipate how users will react. We conclude by synthesizing recommendations for key stakeholders, including for platforms to adopt defense-in-depth strategies that combine AC with server-side validation and to absorb popular non-malicious mods as first-party features, for users to avoid unvetted modified clients, and for policymakers to establish guidelines for social XR use in sensitive contexts, such as K-12 education and professional training.

In this paper, we make the following contributions.

- We conduct the first large-scale analysis of user discourse on client-side modifications and platform defenses in social XR, analyzing ∼5M posts from four social-XR-related subreddits.
- We characterize users' concerns about client modifications and their perceptions of platform defenses, identifying both the harms users experience and the tradeoffs they perceive.
- We investigate how users' concerns and perceptions evolve through an interrupted time-series regression analysis, identifying 10 events in the social XR ecosystem that significantly shift user discourse.
- We provide actionable recommendations for platforms, users, and policymakers to improve user safety and privacy in social XR while preserving accessibility and trust.

2 Background and Related Work

Social XR Platforms. Extended Reality (XR) is an umbrella term for immersive technologies that render interactive 3D environments through head-mounted displays and motion tracking [48]. Social XR refers to multi-user XR applications in which users meet and interact in real time inside shared immersive environments called *worlds* [45]. Each user is represented by an *avatar* whose movements mirror the user's own, producing a strong sense of presence in which virtual interactions are perceived as physical [29, 71]. To support this immersion, each user's device (the client) maintains and renders a local copy of the shared world state, synchronized continuously with other clients [3]. Both avatars and worlds are largely user-generated, built from arbitrary meshes, shaders, animations, and scripts that other users' clients download and render [81]. To scale participation, platforms create instances, which are independent sessions of a world whose state is synchronized among the group of users in it. VRChat [82] and Rec Room [61] are two of the most popular social XR platforms.

XR Security and Privacy. Prior work has studied several security and privacy concerns in XR. From a technical perspective, prior works have explored side-channel and inference attacks on XR sensor data, including de-anonymization from motion, attribute and keystroke inference, and location leakage [25, 51, 52, 72], as well as security weaknesses in the firmware of standalone headsets [70]. In social XR specifically, prior work has demonstrated attacks that enable adversaries to join private world instances [79], capture keystrokes via unprotected traffic [76], manipulate collaborative AR state [73], and exploit deceptive UI [50].

From a user-facing perspective, a parallel body of work has examined how users experience and perceive these threats. Harassment and embodied harms (e.g., virtual groping, verbal abuse, stalking) are widely shown to cause psychological distress comparable to

physical-world encounters [8, 30, 68]. A related line of work examines these harms for younger users and how their parents perceive and manage them [11, 20]. Users also express general privacy concerns, though they often lack awareness of how granular tracking data can reveal sensitive attributes [1, 33, 67].

Safety Features in Social XR Platforms. To prevent harms such as harassment, social XR platforms provide built-in safety features that let users control who can contact, observe, and interact with them [89]. Users can exercise these features individually and collectively, or the platform can enforce them. For example, the *block* feature makes two users mutually invisible and unable to interact, and *mute* silences another user's voice. Together, they are the primary means of cutting off unwanted contact [14, 84]. The *safety zone* feature instead works on proximity rather than identity, automatically hiding and muting any user within a set distance of the user's avatar [89]. Any user can use these features individually.

In contrast, *votekick* is an instance-level moderation tool that lets users collectively kick a disruptive user by majority vote [15]. At the platform level, *private and invite-only instances* act as access control, restricting who can join a space so that users can gather without unwanted intrusion [89], and *trust rank* ties an account's capabilities to a reputation that rises with time and community standing, limiting what low-trust accounts can do [13].

Prior research has evaluated these safety features and found them widely adopted but largely reactive, bypassable, and limited against persistent harassment [13–15, 84].

Client Modifications. Client modifications ("mods") are unauthorized changes a user makes to the software running on their own device to alter its behavior [87]. Common techniques include packet editing, memory manipulation, code injection (e.g., DLL injection), and input scripting [18, 34]. Mods have been documented in multiplayer games [18, 34], mobile and financial applications [38], and web-based social XR platforms [47], with goals ranging from in-game advantage [18] to bypassing purchases or license checks [90] to extracting sensitive information from app memory [75]. Recent work shows that in social XR, modified clients can manipulate the shared state they hold to surveil other users, tamper with virtual objects, and bypass safety features [3, 47]. These works establish what modified clients are capable of, not how users experience and reason about client modifications. While a recent study examined how players make sense of suspected cheaters in online games [9], no comparable account exists for social XR.

Defenses Against Client Modifications. Online platforms have developed several defenses against client modifications. These defenses include server-side validation of client actions, integrity attestation to verify that the client binary has not been tampered with, and Anti-Cheat (AC) engines that monitor the application's runtime environment for signs of modification [18].

Among these, ACs are the most widely deployed in commercial multi-user applications, including social XR platforms [18]. They detect tampering by using techniques such as file integrity checks, memory hashing, system call monitoring, process scanning, and anti-debugging [18]. ACs are grouped by the privilege level at which they run. Early engines ran at the user level (e.g., Valve Anti-Cheat), with application privileges and no visibility into kernel memory, scanning process memory and files for signatures of known cheats [18]. As evasion techniques improved, many vendors moved to kernel-level engines (e.g., Easy Anti-Cheat (EAC) [24], Riot Vanguard) that run as signed drivers with system-wide visibility [18, 21]. Higher privilege improves detection of sophisticated tampering at the cost of broader access to the user's system.

Popular social XR platforms have also adopted AC engines. In July 2022, VRChat integrated EAC [80], a commercial kernel-level engine maintained by a third-party vendor, on whom the platform depends to respond when a bypass appears. Rec Room, on the other hand, deployed EAC in April 2020 and later, between December 2024 and May 2025, replaced it with Referee, its own proprietary user-level engine, which runs with application privileges and can be updated independently by Rec Room [60, 62].

Prior research on AC engines has been largely technical and focused primarily on gaming. It evaluates how reliably engines detect modifications and how readily attackers evade them [18], and examines the behavior of the engines themselves. For example, Dorner and Klausner found that AC engines exhibited rootkit-like behavior [21]. Similarly, other works have raised privacy and integrity concerns about the deep system access that kernel-level engines require [42, 78]. Recently, an interview study of game developers reported that practitioners consider standard anti-cheat tooling only partially effective [39]. None of these works examine how end users perceive these defenses.

Research Gap. Prior work has treated client modifications and their defenses largely as technical problems, focusing on what modified clients can do, how reliably they are detected, and what system access defenses require. How social XR users experience these modifications and defenses remains understudied. First, no work has examined how users experience client modifications on social XR platforms: which of these capabilities they actually encounter, what safety, security, and privacy consequences they associate with them, and whether modified clients compound existing harms on the platform. This is important because, while modifications on other platforms primarily affect program behavior (e.g., unfair gameplay advantages in games), in social XR they intersect with users' identity, presence, and personal safety. An avatar is the user's face and body in shared space, and interactions are immersive, so what happens in a session is perceived as physical. Harassment is therefore a persistent harm on these platforms, with behaviors such as virtual groping or stalking triggering visceral psychological reactions akin to physical assault, which is why platforms ship the safety features described above. Social XR is also used for a wide range of activities, from socializing and live events to education, so a modification that disrupts a session or circumvents these features affects users in settings, such as education and professional training, that have no equivalent on the platforms where modifications have been studied.

Second, how users perceive defenses against client modifications has not been examined on any platform, social XR or otherwise. As described above, prior work on these defenses, most notably AC engines, is technical, covering detection, evasion, and whether such engines act as rootkits. It remains unclear what concerns users themselves have about these defenses, what tradeoffs they perceive, whether they find them effective on social XR platforms, and what alternative approaches they consider acceptable.

Our work addresses both gaps by examining how users on the largest social XR platforms (i.e., VRChat, Rec Room) experience client modifications and perceive defenses against them.

3 Methodology

To understand users' concerns and perceptions about malicious client modifications in social XR and how existing platform defenses shape users' experience, we conduct a multi-stage analysis of online discussions on Reddit [64]. Online discussion platforms (e.g., Reddit, X, Stack Overflow) have been widely used to study user security and privacy perceptions, including for IoT [40], conversational AI [2], online abuse [85], and safety features in social XR [13–15]. Analyzing Reddit discussions allows us to (i) study users at a scale infeasible for interviews or surveys, (ii) capture in-situ accounts as users articulate them to peers rather than to researchers, and (iii) track how perceptions evolve around platform events, which point-in-time methods cannot capture [12, 59].

Figure 1 presents an overview of our analysis. We first collect a large dataset of Reddit posts and annotate a sample to identify content related to malicious modifications and anti-cheat engines ❶. Next, we train a classifier to identify relevant posts across the full corpus ❷. Then, we conduct a qualitative thematic analysis to develop a codebook capturing users' perceptions of malicious modifications and platform defenses and use this codebook for automated labeling of all relevant posts ❸. Finally, we perform a regression analysis to investigate how users' concerns shift in response to major events in the social XR ecosystem ❹.

3.1 Data Collection

We compiled a list of popular subreddits related to social XR using Reddit's search engine. From this list we selected four for our analysis: `r/VRGaming`, `r/virtualreality`, `r/VRChat`, and `r/RecRoom`. We select `r/virtualreality` (~2.1M members, ~4.3K weekly posts) and `r/VRGaming` (~159K members, ~2.6K weekly posts) because they are the two largest general-purpose XR communities on Reddit and discussions of modifications and defenses often extend beyond individual platforms, with users comparing experiences. We select `r/VRChat` (~213K members, ~3.4K weekly posts) and `r/RecRoom` (~63K members, ~1.7K weekly posts) because they are the two most popular social XR platforms by active user base, with VRChat averaging tens of thousands of concurrent users on Steam [74] and Rec Room reporting millions of monthly active users across PC, mobile, and console [35], and because both platforms have deployed dedicated anti-cheat engines in response to malicious modifications (Section 2). Their contrasting approaches and deployment timelines enable both cross-platform comparison and longitudinal analysis.

For each of these subreddits, we collected all posts (including original submissions and comments) from its inception through January 2026, using publicly available Reddit dumps [5], yielding a raw corpus of approximately 5M posts. We removed duplicates, empty and deleted posts, and filtered out bot-generated content using known bot signatures (e.g., the presence of the word "bot" in the username) following prior work [2, 14], leaving a final dataset of ~2.1M posts, including ~263K submissions and ~1.83M comments.

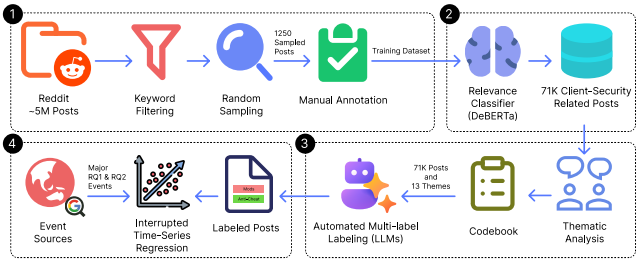


Figure 1: Methodology Overview

3.2 Identifying Relevant Posts

Given the scale of the dataset (~ 2.1M posts), identifying posts that discuss malicious modifications or platform defenses is a significant challenge. An intuitive approach is to perform a keyword search to find relevant posts. However, keyword filtering alone can suffer from low coverage, missing relevant posts that use alternative phrasing, and high false positives [53], especially in platform-specific subreddits where users discuss gameplay bugs, general mods, and platform updates in overlapping terms. Hence, we adopt a hybrid approach: we first use keyword filtering and manual annotation to create a labeled dataset of client security-related posts, then use this dataset to train an automated relevance classifier.

Keyword Filtering and Sampling. We compiled candidate keywords by reviewing literature on security issues in social XR [8, 30, 67, 84] and client-side modification techniques [18, 34, 47, 87], then refined them iteratively by searching Reddit posts, reviewing matched results, and identifying new relevant terms. We provide the complete keyword list in our artifacts [49]. To ensure balanced representation, we organized our keywords into three groups aligned with our research questions, targeting (1) modified clients, cheats, and their effects (e.g., *modified client*, *crasher*, *avatar rip*), (2) anti-cheat mechanisms and enforcement (e.g., *Easy Anti-Cheat*, *EAC*, *Referee*, *kernel-level*), and (3) the broader security and privacy framing (e.g., *exploit*, *malware*, *privacy*, *surveillance*). For `r/virtualreality` and `r/VRGaming`, we retained a post only if it also contained at least one social XR term (e.g., `VRChat`, `Rec Room`).

We then drew 250 posts from each group, 250 posts with no keyword match, and 250 `r/RecRoom` posts on or after December 19, 2024 that matched at least one keyword group, to ensure adequate coverage of the newer Referee deployment. Since the first two groups are narrow, the general group and the keyword-free posts keep the classifier from learning to spot our keywords rather than the topic itself. Our final seed corpus consisted of 1,250 posts.

Relevance Annotation. To construct a reliable seed dataset for training our classifier, we manually annotated the 1,250 sampled posts, labeling each as relevant or not. Two authors (experts in XR security) jointly reviewed an initial subset of 250 posts to create a detailed annotation guide, defining clear boundaries between relevant and irrelevant content. Using this guide, the authors then independently labeled the corpus, achieving substantial inter-rater agreement (Cohen's Kappa, $\kappa = 0.76$) [17], and resolving all remaining disagreements through discussion following standard practice

for thematic analysis [44]. The final labeled dataset included 486 XR client security-related and 764 unrelated posts.

Relevance Classification. We split the 1,250 labeled posts into train and test samples (80-20 split) and trained a binary classifier to identify client-security-related posts across the full corpus. We evaluated multiple models for this task, including fine-tuned transformer models (i.e., DeBERTa [36], RoBERTa [41]) and prompted LLMs (i.e., Gemini 2.5 Flash, Flash-Lite, and Pro [19]). We selected DeBERTa as the final relevance classifier given its best overall performance among the evaluated models, with an F1-score of 0.91. Full results are included in our artifacts [49].

Applying the classifier to all ∼ 2.1M posts yielded ∼ 71K client-security-related posts from 22,314 unique authors, spanning September 2014 through January 2026, of which 72.3% are from `r/VRChat`, 25.9% from `r/RecRoom`, and the remainder from `r/virtualreality` and `r/VRGaming`. We randomly sampled 200 posts from these ∼ 71K and manually verified their labels, finding an accuracy of 0.97.

3.3 Data Analysis

3.3.1 Qualitative Analysis. To identify users' perceptions of client modifications and platform defenses, we conducted a thematic analysis [10], starting from an initial codebook informed by our keyword exploration and the relevance annotation process (Section 3.2). To refine the codebook, two authors iteratively sampled and coded posts in batches of 50 drawn from the ∼ 71K client-security-related posts, after each batch introducing codes for emerging perceptions, refining existing ones, and updating code descriptions. We stopped when no new themes emerged across two consecutive batches, reaching thematic saturation [66] at 486 posts. We then grouped similar codes into broader themes aligned with our research questions. We do not present intercoder agreement as all posts were reviewed together [44]. The complete codebook with label definitions is included in our artifacts [49], and we describe each label with examples in our findings (Section 4 and Section 5).

3.3.2 Automated Thematic Labeling. Analyzing how users' perceptions evolve requires labeling all ∼ 71K client-security-related posts. Given the scale of our dataset, manually annotating all posts was infeasible, so following prior work that scales qualitative labels via classification [2, 85], we used an LLM-based multi-label classifier to apply the codes from our qualitative analysis (Section 3.3.1) to all client-security-related posts. Importantly, we used LLMs only for post-hoc labeling, not to develop codes.

Since each post can express multiple concerns simultaneously (e.g., a user may describe avatar theft and criticize anti-cheat performance in the same post), we treat each codebook theme as an independent binary classification task, each with its own prompt and a few labeled examples. We evaluated multiple LLMs for this task, including Gemini 3.1 Flash-Lite [32] and GPT-5.4 Mini [56], using 80 of the 486 coded posts for prompt development (30 as few-shot examples and 50 for iterative refinement) and the remaining 406 for evaluation. Gemini 3.1 Flash-Lite achieved the best overall performance, with a weighted F1-score of 0.89. Per-label metrics are reported in Appendix A Table 1; the per-label prompts and the results for GPT-5.4 Mini are included in our artifacts [49].

After applying the classifier to all client-security-related posts, to assess its performance on unseen data, we drew a further 250 posts from the ∼ 71K, disjoint from the 486, and manually coded them to generate ground truth labels. The classifier achieved a weighted F1-score of 0.94 (weighted precision 0.91, recall 0.98) and an exact-match accuracy of 0.93 on these posts.

3.3.3 Temporal Analysis. To analyze how user discourse on client modifications and defenses changes over time in response to platform updates and events in the modding and AC ecosystems, we conduct an interrupted time-series regression analysis (ITSA) [7]. We first collected a list of relevant events from official announcements (e.g., VRChat developer updates [83]), XR and security news sites (e.g., Road to VR [65], Hacker News [86]), community wikis (e.g., Rec Room Fandom [62]), and academic disclosures [47, 51, 76]. Two authors reviewed the results after filtering with relevant keywords from our qualitative analysis (Section 3.3.1) to identify a candidate set of 63 events, including AC deployments and bypasses, mod releases and tools, ban waves, and security disclosures. The complete list is included in our artifacts [49].

We then ran an ITSA using this list of events and our labeled dataset (Section 3.3.2) to identify events associated with shifts in user discussions. For every $\langle event, thematic\ label \rangle$ pair, we fit a segmented regression on a window centered on the event date [7], using Newey-West HAC at lag 7 [54] for autocorrelation in daily post counts and Benjamini-Hochberg FDR (BH-FDR) correction [6] across all event-label tests, excluding pairs with near-zero samples. We consider a shift significant if $q < 0.05$. We choose BH-FDR because stricter family-wise approaches, such as Bonferroni correction [37], inflate false negatives and would be overly conservative for our number of comparisons [58]. Because window length is a modeling choice rather than a property of the data, we repeated the analysis at window sizes of ±7, ±14, and ±30 days for robustness.

A significant shift need not be attributable to the event, as it may reflect an overlapping event or a pre-existing trend. We therefore manually review each significant shift and report an association only where posts around the event date reference the event. Substantive review of significant findings is a recommended practice when assessing evidence about specific hypotheses [58]. This gives us a set of 10 events significantly associated with a shift in at least one codebook theme. Appendix B Table 2 lists these events, and we discuss the findings in Section 6. Since overlapping events and trends cannot be fully ruled out, we describe our results as temporal associations rather than causal effects.

4 RQ1: Users' Concerns on Client Modifications

Users articulate a wide range of concerns about how client modifications can be weaponized against other users, the platform, and content creators. We organize these concerns into five themes: disruption of users' experiences (48.1% of labeled posts), asset theft (26.6%), evasion of safety features (2.9%), access to private information (3.8%), and security side effects of mods (3.1%). Figure 2 shows the prevalence of each theme across the labeled corpus.

4.1 Disrupting XR Experience

A recurring concern across discussions is that client modifications disrupt others' experiences, ranging from outright crashes to subtle interference that degrades social interaction.

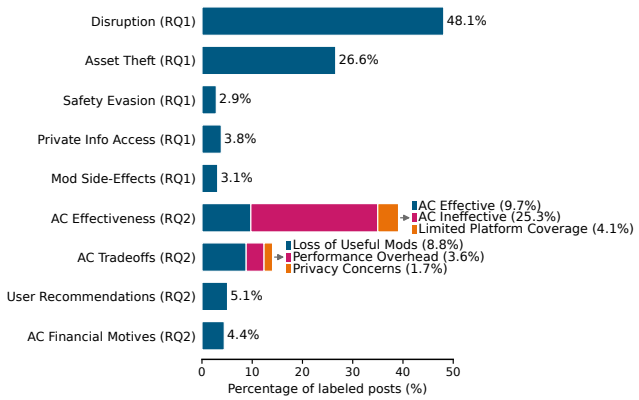


Figure 2: Prevalence of RQ1 & RQ2 themes on client modifications' harms and perceptions of anti-cheat engines. Percentages are over posts carrying at least one label; labels are not mutually exclusive, so bars sum to more than 100%.

4.1.1 Crashing. The most severe form of disruption discussed by users is crashing, in which a malicious user with a modified client forces an entire virtual world instance to terminate unexpectedly. Users describe how mods enable this by sending malformed or unexpected updates over the network that the receiving client cannot parse correctly, exploiting weaknesses in how clients handle untrusted peer input. For instance, a user noted, *"[The modified clients] send malformed Photon (VRChat's networking library) packets, and since they aren't handled by the client, it causes a crash"* (Q1).

Crashing Social and Live Events. Users describe the consequences as most severe at live social events, where dozens or hundreds of attendees gather in a single instance. A single malicious actor running a crash mod could collapse an event that organizers and attendees had invested significant effort into preparing. For instance, one VRChat event host reported: *"I'm dealing with crashers at my events. There's a small group of us [organizers] who pour a ton of time and effort into throwing fun nights for people . . . and then you get folks whose whole aim is to cause trouble and crash everyone. I don't get it"* (Q4). Similarly, a Rec Room user described hosting events as *"an unending fight against crashing players"* (Q13).

Crashing as a Harassment Tool. Users also frequently describe crashing as a tool for targeted harassment and cyberbullying, where an attacker repeatedly crashes a specific user's client to drive them out of a virtual world instance or off the platform entirely. Unlike event-level crashes, which are indiscriminate, these attacks single out individuals: *"there is one person who keeps telling me I should end my own life . . . he has crashed me several times and has sent me threats to kill me because he thinks I've been stealing his games . . . I've tried to block him, but he just keeps creating new accounts"* (Q43). Verbal abuse, repeated crashing, and account-level evasion compound, as blocking the harasser does not stop them, because they create new accounts, and the technical capability to crash the victim is preserved across each new identity.

Creative Crashing Techniques. Users also describe the creative and evolving nature of crash techniques. Crash payloads were not always delivered through obvious actions like joining an instance; in some cases, they could be triggered by interactions as routine as viewing another user's profile: *"this guy [said] he'd crashed someone, so I asked him how, and he told me to not look at his bio . . . when I did, my game crashed"* (Q23). This example illustrates how crash vectors can be embedded in surfaces that users do not normally treat as risky to load, blurring the line between passive browsing and active engagement with a malicious peer.

4.1.2 Degradation of User Experience. Beyond outright crashes, users discuss a range of modifications that degrade rather than terminate the experience. These do not cause a direct crash but make the virtual world unusable in ways that are harder to attribute and easier to sustain over time.

Inducing Lag. A common form of degradation that users report is lag-inducing object spam, where mods spawn many in-world objects to overwhelm the renderer: *"lately, the events my friends and I host keep getting hacked, where someone either crashes the whole instance or spawns a lot of random objects all over the world, causing the game to lag"* (Q3). Users describe object spam and crashing as alternative outcomes of the same kind of attack: when the instance does not crash outright, it is left too laggy to use.

Forced Teleportation. Another disruption was forced teleportation, where a mod manipulates another user's position within a world or across instances. Users report mods that spawn portals at a target's exact in-world coordinates, moving them to a different world or instance without their consent. For instance, one user reported *"Very often, I'll see crashers drop a portal and teleport everyone else in the instance right on top of it. Anyone who has the stupid auto-accept-portal setting on gets teleported out of the instance"* (Q67).

Object Manipulation. A related form of disruption involves modified clients that manipulate world objects. This can be done to take or hog ownership of interactable objects. For example, a user noted that *"hackers can forcibly grab whatever item you happen to be holding"* (Q12). Another form of object manipulation controls objects en masse, causing every interactable item in an instance to swarm or fly around a target. The victim is rendered unable to grab, use, or destroy these objects, and in some cases, this can also overload the victim's client, subsequently crashing it. For example, a user described it: *"there is a hack where every object in the world gets teleported close to you, and when the objects come close enough, your client starts trying to claim ownership of the objects, which requires doing some physics calculations and overwhelms the victim's machine with simultaneous ownership-transfer events"* (Q77).

4.2 XR Asset Theft

Several discussions in our analysis highlight how client modifications enable the theft of digital assets, including user-created avatars and worlds, which are popular features in social XR. Users call this "ripping". Because clients must download avatar packages from peers locally to render them, a modified client can intercept those packages and write them to disk: *"When you equip your avatar, the package for the avatar which you uploaded [to VRChat] is downloaded on everyone else's device . . . a modified client just grabs that package and saves it on their computer. From there, they can load it*

into Unity, and now your whole project is theirs" (Q56). This explanation reframes ripping as an architectural rather than incidental property of the platform. The same delivery mechanism that makes avatars visible to other users is what makes them extractable.

Identity Theft Through Avatars. We find that users not only describe avatar theft as digital piracy or intellectual property theft, but also as a violation of their personal identity. On social XR platforms, avatars function as users' persistent face and body in shared space, and users often invest significant time in customizing them. Modified clients allow attackers to obtain an exact copy of these avatars. For example, one user articulates the embodiment directly: *"For three-plus years now my avatar has been my face and my identity in VR. I've poured hundreds of hours into building it … losing it would feel like having my identity stolen and someone parading around as me in real life"* (Q29). Another user framed ripped avatars as a form of impersonation and identity theft, where a stolen avatar lets the ripper pass as the original owner in shared spaces: *"Personal avatars are a form of self-expression as well, so taking one is basically identity theft. There's more to it than that, but that's the core of it. Picture walking down the block and coming face to face with a copy of yourself"* (Q34).

IP and Creativity Theft. Users describe ripping as a sustained financial and intellectual-property harm to creators who build avatars and worlds for others. First, ripping affects paid creators directly, since extracted avatars often contain assets purchased from third-party storefronts that creators paid to license. For instance, one creator described *"People are ripping avatars just to get the paid assets used in building them for free. With everything I've put in from the Unity store and outfits from DeviantArt, my avatar is probably worth about $50. Creators want controlled distribution of their assets, and because enough paid assets have been passed around, several creators have already taken their work down permanently"* (Q9). As a result of ripping concerns, some creators withdraw their work from the marketplace, reducing the supply of high-quality assets for everyone. Second, ripping produces a chilling effect on creator participation, with experienced creators citing it as a reason not to invest time on the platform: *"Being a computer graphics artist myself, I can't justify putting time and effort into this platform (VRChat) until they deal with the hacking. I would hate to accept a commission, make someone a really nice avatar, and then have another user with a modified client rip it"* (Q51).

Asset theft extends beyond avatars to user-built worlds, which represent comparable creative investment (scene, lighting, audio, interactions) and often combine the work of several contributors. Users describe both private worlds and entire creator portfolios being ripped: *"The design content inside my private world shouldn't be accessible to anyone I haven't authorized. It's not anything lewd, just models that I've been working on for a long time. The last thing I want is some random kid running a modified client ripping my projects and calling them his own"* (Q28). A second user reports the simultaneous theft of avatars and worlds as a recurring experience: *"I found that some of my older avatars had been ripped, and so had the recent ones. They'd even stolen my old VRChat worlds"* (Q72). The community-as-creator dynamic of social XR amplifies these harms. Unlike platforms where content is produced by a separate professional studio, a substantial fraction of the social XR user base is the creator population, which collapses the distinction between user-harm and creator-harm in user accounts: *"the same people playing are also the ones creating the avatars … it makes sense why there's so much resentment toward people stealing avatars and worlds"* (Q19).

4.3 Evading Social XR Safety Features

Many users express concerns that client modifications enable malicious users to evade built-in safety features, which they attribute to client-side enforcement or reliance on identifiers reported by the client. Because these features are how users manage their own safety in a shared, embodied space (Section 2), their evasion strips users of any means of protecting themselves. Where prior work studies the effectiveness of these features [13–15, 84], users here describe them failing outright against a modified client.

Block Evasion. Users describe how modified clients undermine VRChat's block control. While blocking is intended to prevent the blocked user from interacting with or seeing the blocker, modified clients can ignore some or all of these enforcements. As one user explained, *"The trouble with blocking [modified clients] is that it doesn't always work … the administrative powers sit on the client side"* (Q44). Another user reported that they continued to be tracked across instances after blocking: *"What gets to me is that a user I already blocked can still somehow track me down and there's nothing I can do about it … they still show up on the player list for me every time they join the same world instance"* (Q60). This allows attackers to continue harassing and bullying victims despite being blocked.

Votekick Evasion. Users report that modified clients also defeat community-level safety features such as votekick, which is an in-instance moderation tool that lets users collectively kick a malicious actor by majority vote. For example, one user noted *"people running mods can tweak the game so it stops them from ever being kicked out of a lobby"* (Q7). This may even extend to platform-level moderation. For example, one user described an in-instance attacker who openly invoked this immunity: *"He was showing off, going on about how nobody could vote kick him and not even the mods could remove him thanks to his [modified] client"* (Q38).

Forced Entry into Private Instances. Beyond defeating individual safety actions, users report that modified clients defeat the access-control mechanisms that they rely on for protection from harassment, including private and invite-only instances. Modified clients can join invite-only world instances without invites, allowing attackers to surveil or follow targets across instances they should not be able to reach. One user reported a stalking incident: *"A friend of mine got harassed and stalked in VRC by someone running a mod that let him sneak into invite-only world instances"* (Q45). Another summarized the phenomenon as: *"On VRChat, people [running modified clients] can slip into private instances while remaining invisible, and some have used that to stalk other players"* (Q15).

4.4 Accessing Private Information

Users also report that modified clients can be used to access private information about other users (3.8% of labeled posts). Modified clients allow malicious users to see information the interface does not reveal, such as who is present in a private instance, friend lists, and whether they are blocked. For example, one user described a stalker running a modified client to see which friends they were

with inside an invite-only world (Q82). Another questioned why blocked status was visible at all: *"why does it show you who has blocked you? That is ripe for misuse, and it is not something you have any reason to know in the first place"* (Q84).

Earlier posts describe users' IP addresses being obtained by others in the same instance, with one user reporting that tools had been built specifically to collect them (Q85). Users report these addresses were then used to dox them or to flood their home connections until the connection dropped. One described people locating them mid-stream to cut the stream off: *"when I streamed regularly, people would often find me in game to crash my stream"* (Q52).

4.5 Security and Privacy Side Effects of Mods

Our analysis also revealed how client modifications harm users who install these modifications themselves. Modified clients are typically distributed outside platform-controlled channels, where there are no security checks, so users run whatever the distributor shipped. One user warned, *"steer clear of client mods. There's no telling what their code is actually doing, and a few can even pull down extra code from some remote source. Honestly, you'd be no safer than grabbing random malware with your anti-virus switched off"* (Q49).

Users report that modified clients are commonly bundled with credential-stealing payloads, most notably auth-token stealers that hijack user accounts: *"I'd put the odds at roughly 98% that [one modified client] grabs your auth token before [the other one] does"* (Q25). VRChat staff have publicly attributed account hijacks to modified-client use: *"A huge amount of account hijacks we see are the result of using modified clients or SDKs. If you run someone else's code, and they wanna steal your password … nothing stops them"* (Q64).

Users note the irony: those attempting to cheat become victims of theft themselves. For example, one user noted, *"Wait, people genuinely pay for these [modified] clients? I always figured they'd be 'free' but loaded with malware that ends up wrecking the running user's own PC with all sorts of nasty stuff"* (Q31). Other users reported that these modified clients are often *"riddled with keyloggers"* (Q6).

> **Key Takeaways (RQ1).** Users report disruption of shared sessions, theft of digital assets, evasion of safety features, and the security risks of installing mods themselves. They do not describe these as software-level failures but as violations of availability, identity, safety, and personal agency.

5 RQ2: Users' Perceptions of Platform Defenses

Of the defenses platforms deploy, anti-cheat (AC) engines are the only ones users discuss substantively, so we organize this section around perceptions of ACs and the alternative defense models users propose where ACs fall short. Figure 2 shows the prevalence of AC-related labels across labeled posts. We categorize users' perceptions into four main themes: (1) the effectiveness of ACs against the harms users associate with client modifications (covering negative sentiment at 25.3% of labeled posts, positive sentiment at 9.7%, and limited cross-platform coverage at 4.1%), (2) the tradeoffs ACs introduce for ordinary users (loss of useful mods at 8.8%, performance overhead at 3.6%, and privacy concerns at 1.7%), (3) the recommendations users make for alternative or complementary defense models (5.1%), and (4) the commercial motives users attribute to platforms' AC adoption (4.4%). Notably, while prior work focuses on privacy and kernel access [21, 42, 78], effectiveness, performance, and loss of useful mods were more frequent concerns in our data than privacy, which appeared in only 1.7% of posts.

5.1 Anti-Cheat Effectiveness

Users describe ACs as effective in reducing certain attacks, while also identifying gaps that limit their overall protection.

AC Seen as Effective. A subset of users credit the AC for reducing harms that were common before its deployment, while describing it as deterrence rather than full prevention. For instance, a VRChat user noted about VRChat's Easy Anti-Cheat (EAC), *"EAC was never going to stop modding completely, but it did deter a large number of users"* (Q58). Other users name specific attack classes that became less common after EAC deployment, particularly indiscriminate object spawning to disrupt victims: *"You can criticize EAC all you want, but ever since its deployment I haven't once seen someone surrounded against their will by every object in the room"* (Q74).

Users also describe AC protection as depending on how quickly the platform responds when a bypass appears. For example, one user explained how EAC was fast to react after a bypass was released for VRChat: *"They patched that bypass some time yesterday, and it is now detected by EAC. Users using it were banned, and EAC will probably soon get a client update that shuts it down"* (Q11).

AC Seen as Ineffective. Many users are skeptical and describe ACs as inherently bypassable, arguing that protection only lasts until someone builds a bypass. For instance, one user commented *"For anyone who actually knows what they're doing, getting past EAC isn't hard. Every game that's relied on it has ended up bypassed sooner or later, and usually fast. It's [like] Digital Rights Management (DRM). Basically, all it does is postpone the inevitable"* (Q40). Another user explicitly noted how ACs are ineffective against crashers: *"Blocking the people who crash you doesn't work. They can still keep tabs on you, still see you, still rip your [avatars], and still crash you"* (Q27).

Limited Cross-Platform Coverage. Users frequently note that AC engines do not run on every supported platform, leaving specific, well-known bypasses open. For example, one user noted that a common bypass exploits the Android version of VRChat, which EAC does not cover: *"Modding hasn't gone anywhere, since Easy Anti-Cheat doesn't run on Android; you can just run the Android build on PC through an emulator. Nice work, VRChat team. Modding is still doable, and the only real change is that every player's experience is now worse"* (Q21). Since users can join a VRChat world from any supported device, even if the AC prevents modding on PC, the harms persist unless all platforms are protected.

User-level vs. Kernel-level AC. We find that users acknowledge the differences between user-level and kernel-level ACs adopted by different social XR platforms. For instance, users mention that Rec Room's in-house, user-level AC, Referee, is not as effective as the previously adopted and retired commercial, kernel-level AC, EAC: *"Referee just needs to be scrapped completely. Rec Room clearly doesn't have the staff to run an in-house anti-cheat. Cheating is worse than it has ever been precisely BECAUSE of Referee; it wasn't great under EAC either, but it was noticeably better. Just move to Battleye (a commercial kernel-level anti-cheat) and be done with it"* (Q53). The

user also noted the advantage of a commercial AC: its vendor has the bandwidth to maintain it against vulnerabilities and bypasses.

5.2 Anti-Cheat Tradeoffs

Beyond questioning AC effectiveness, users describe kernel-level ACs as imposing concrete tradeoffs that often fall on users who never installed a modification themselves. These tradeoffs are reported mostly for EAC. Rec Room's user-level Referee draws effectiveness complaints (Section 5.1), but we find few user complaints on performance, usability, or privacy, consistent with the difference in privilege level between the two engines (Section 2).

Performance Overhead. Users attribute concrete performance regressions to AC engines, comparing frame rates before and after AC adoption on the same hardware and in the same worlds: *"After the recent [EAC] updates, I've noticed my framerate has fallen from over 100 FPS to somewhere around 50-70 FPS, and walking towards a mirror drops me to 30 FPS and this is just in the default home world"* (Q36). Users running a vanilla version of the app with no prior mod usage report the same regression in performance: *"The latest update has made the game genuinely unplayable for me, thanks a lot. And to be clear, this isn't a 'I can't run my mods' post. Not once in my entire time playing have I ever run any kind of mod. I've cheerfully stayed oblivious to this whole side of the game and have always [stuck] with the Vanilla (unmodified) version of the app"* (Q69). Several users demand that platforms fix performance problems before deploying the AC. One VRChat user noted: *"Maybe sort out the performance hit that comes from running EAC on top of an [app] that's already a pain to run in the first place"* (Q17).

Usability Issues. Beyond performance regressions, users report cases where the AC prevents the application from running at all. For instance, a VRChat user on a cloud PC describes the loss of virtual machine (VM) compatibility: *"Please stop letting EAC block Shadow PC. Not all of us use [Shadow PC], but a fair number do, and for VRChat but also plenty of other EAC-supported games as well. Right now [Shadow PC] is being flagged as a VM and blocked, so a lot of us just can't play VRChat, which is a real bummer"* (Q47). Linux users describe similar exclusion: *"Anti-cheat and Linux have never gotten along, and since I'm on Linux, this update pretty much locks me out of playing entirely"* (Q62).

Other users describe AC failures that render the application unusable, even on stock systems with no mods: *"Every time I try, my PC crashes and shows a blue screen with a heap corruption error in kernel mode. The EAC patch has left the game completely unplayable for me with or without mods"* (Q8). Another user reported that the AC prevented the app from launching without any actionable error: *"Hadn't touched VRChat in a while, ran it moments ago, and it won't open at all"* (Q33). In such cases, users are confused and have no way to debug or resolve the failure, making the platform unusable.

Loss of Useful Mods. Another tradeoff users raise is that the AC removed modifications relied on by disabled and resource-constrained users. First, the AC ironically banned defensive mods (e.g., anti-crasher tools) that protected users from the very harm the AC was deployed to mitigate: *"What worries me is that crasher avatars will make a comeback. People used to rely on mods as a shield against being crashed, and now there's basically nothing left to stop crashers"* (Q70).

Second, the AC removed accessibility mods that disabled users relied on: *"…some players with disabilities relied on mods for accessibility reasons and after the update, they cannot play at all"* (Q26). Users point to the asymmetry between the loss of useful mods and the reduction of malicious mods caused by AC by comparing with other AC-protected games where cheating persists: *"…EAC disproportionately affects more legitimate players, like the ones running anti-crasher mods, than it stops the genuinely malicious users. This is evidenced by the fact that cheating is still common in Fortnite, Halo MCC, and Elden Ring, and every one of those runs EAC"* (Q55).

Third, the AC removed quality-of-life (QoL) mods that, in some cases, the platform later reabsorbed as official features. For instance, one user complained about mods enabling avatar search getting banned: *"When they banned Avatar search along with a handful of QoL mods, features that honestly should have been in the game from day one, I nearly walked away from VRC for good. Thankfully, even though I still think EAC is far too invasive, around 90% of what those mods did for me has since shown up as built-in features"* (Q39).

Kernel-level ACs and Privacy. Users also report concerns about the system-level access required by kernel-level ACs, such as EAC adopted by VRChat. On data collection, some users cite EAC's End-User License Agreement (EULA), grounding their privacy concerns in the engine's stated terms: *"Is EAC actually spying on me? This is straight from the EAC EULA: Easy Anti-Cheat is storing information regarding cheating methods used in the Game. By installing, copying, or otherwise using this Software Product, you give your consent that Easy Anti-Cheat may gather, store, and publish data …"* (Q75). Others identify specific data points the engine reports: *"Whenever their driver starts up, EAC reads off your hard drive's serial number and your MAC address"* (Q14). Users object to being required to accept a kernel-level component to participate, and suspect it collects information they have not meaningfully consented to.

On security, users describe the engine itself as a system-wide attack surface. A vulnerability in EAC becomes a vulnerability in the user's entire system, because the engine runs with kernel privileges: *"With software running at the kind of privilege level a [kernel-level] anti-cheat has to have, any bug that slips by could be catastrophic, potentially causing a lot of damage without the end-users' knowledge"* (Q50). Another user framed the disproportionality of kernel-level anti-cheat for a social platform: *"there's no reason a social chat app should require a surveillance rootkit"* (Q66).

5.3 User Recommendations

Beyond evaluating the effectiveness and tradeoffs of ACs, users also propose alternative defense models against client modifications that they perceive as more suitable for social XR. We identify three categories of recommendations provided by users: (1) official mod support, (2) stronger account-level enforcement, and (3) architectural fixes that move enforcement off the client. We discuss the feasibility of these recommendations in Section 7.2.

Official Mod Support. The most common recommendation from users is for platforms to provide an official modding API, allowing vetted accessibility and quality-of-life modifications while excluding malicious ones. A transitional version of the proposal is that the platform should retain commonly used non-malicious mods until equivalent functionality becomes native. For instance, a user

suggested *"At the very least, they could identify the most popular non-malicious mods and allow people to run them until the platform can actually add those functionalities as built-in features. If they still want to ban them once that's done, I still wouldn't support it, but if they're set on going through with this plan, that's the best way to handle it"* (Q22). Users cite other games as precedent for an alternative defense model that incorporates rather than excludes modding: *"What I can't figure out is why they don't have official mod support? Similar to World of Warcraft, Team Fortress 2 or any other game that ships with mod support built in?"* (Q59). This reframes mod policy as a configurable feature rather than a binary switch.

Account-level Enforcement. Other users recommend increasing the cost of getting banned rather than the cost of running mods. They suggest banning accounts based on hardware identifiers (HWID) and IP addresses. Several users also suggest increasing account-creation friction by making the platform paid. Both VRChat and Rec Room are currently free on all major XR stores, which users describe as enabling cheap account churn after enforcement: *"there is a need for stricter bans, including HWID bans, IP bans, and permanent account bans, even for the minor violations … turning [VRChat] into a paid game would [also] cut the crap in here too"* (Q5).

Server-side Validation. Users technically familiar with social XR apps' networking architecture recommend that the platform validate security-sensitive actions on the server rather than rely on client-side scanning to detect their misuse. Users frame AC adoption as treating a symptom rather than the underlying cause: *"The funny thing is EAC was a patch over an open wound, the API hooks they never fixed. We're still being crashed by the same things we were in 2021"* (Q41). They suggest how the server could perform necessary checks to prevent malicious users from disrupting others' experience. For example, one user noted, *"The server should trust the client less. There are barely any server-side checks against malicious behavior. If it actually verified things like one person grabbing every item at once, that would fix it"* (Q76).

5.4 Financial Motives for Anti-Cheat

A smaller set of posts (4.4%) discusses why platforms adopted AC engines in the first place, attributing the decision to commercial motives. Users read the ban on mods as a way to keep functionality behind a paywall, since a mod that adds a feature for free removes the platform's ability to sell it: *"You can't charge extra for a feature when modders have already shipped it for nothing"* (Q78). Others tie the timing of AC deployment to the platform's paid offerings, with one user concluding *"this beta is about as close to microtransactions as it gets … so now we can see what EAC was really for"* (Q65).

Some users attribute adoption to investor demands, describing them as met at their expense. One user wrote, *"They brought in EAC to keep investors happy … once you answer to investors instead of the people actually playing, that is when a game loses its soul and dies"* (Q79). Other users accept the same premise but defend the decision, arguing that the platform needs the AC to satisfy investors: *"VRChat has to pass security and financial audits regularly, and an anti-cheat is required for that; fail those and the investors walk away, and a creator economy handling real money is off the table"* (Q80).

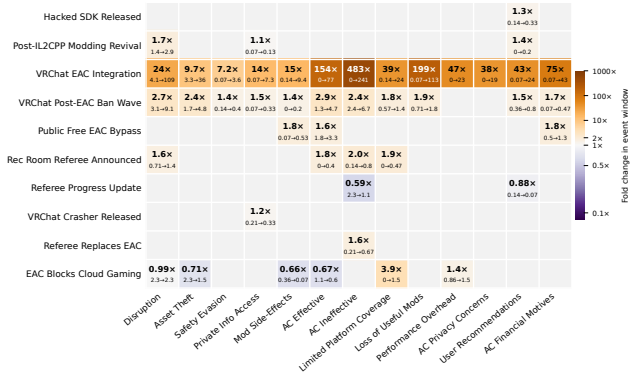


Figure 3: Association between the 10 verified events and 13 codebook themes in a ±14-day window. Each cell reports the fold change in mean daily posts after versus before the event. Values above 1× indicate increases, below 1× decreases. Grey cells were either not statistically significant or were filtered during manual inspection of posts around the event.

> **Key Takeaways (RQ2).** User attitudes toward ACs are mixed: a subset gives them credit for reducing harms after deployment, while others frame them as a contested defense whose trade-offs (privacy, performance, usability) fall disproportionately on users who never installed a mod. A few attribute AC adoption to commercial rather than security reasons.

6 RQ3: Evolution of Users' Perceptions

Our analysis (Section 3.3.3) identifies 10 events, including platform AC decisions, release of mods and bypasses, and updates to ACs, that are significantly associated with shifts in user discourse on at least one codebook theme. Figure 3 shows the results at the ±14-day window. Each significant cell reports the fold change in mean daily posts after the event relative to before, computed with Laplace smoothing ($\varepsilon = 0.5$), and the pre- and post-event means. We repeated the analysis at window sizes of ±7 and ±30 days and found that 42 out of the 45 associations at ±14 days hold in at least one other window, indicating that they are not artifacts of the choice of window size. Where an association holds across windows, its direction is unchanged. Full results are included in our artifacts [49].

VRChat's Adoption of Commercial Kernel-Level AC. We find that the adoption of EAC (a commercial kernel-level AC) by VRChat in late July 2022 [80] coincided with a rise in discussions on the merits and demerits of AC engines and the harms of mods, with significant association across all themes (Figure 3). Some users were happy with the EAC adoption since it would mean the end of malicious mods that enable disruption and asset ripping. One user noted, *"[EAC] is a step in the right direction. It won't stop every modified client, though it should cut down the angsty kids who desynchronize a room whenever they're crossed"* (Q18). It also raised the privacy, accessibility, and performance concerns detailed in Section 5.2. Several users voiced intentions to leave the platform over kernel-level access (*"Any game that puts kernel-level anti-cheat on my machine is one I'll skip"* (Q32)) or the removal of useful mods

(*"Adopting EAC means losing mods like avatar search, the portable mirror, face tracking, bhaptics, mute text-to-speech, advanced safety, and avatar hider"* (Q63)). The announcement also enumerated the harms EAC was meant to address, prompting users to discuss them.

Later, in December 2022, VRChat issued account terminations and hardware ID bans, coinciding with increased posts both for and against EAC (AC Effective 2.9×, AC Ineffective 2.4×) and renewed discussions on the harms and utility of mods, and the tradeoffs of a kernel-level AC engine. Some users reported false bans on platforms without EAC support. For instance, one user noted, *"Quest users are getting banned when EAC doesn't even run on Quest devices"* (Q30).

Rec Room's Migration from Commercial Kernel-Level AC to In-house User-Level AC. Users initially expressed strong support for Rec Room's migration from a commercial kernel-level AC to a proprietary in-house AC (Referee), then reported the same harms once EAC was retired. Rec Room introduced EAC in April 2020. Unlike VRChat's adoption of EAC, this received little discussion in our data, and we find no significant association with any codebook theme. However, concerns about cheating persisted in the following years. For example, one user noted, *"Recently, the anti-cheat was not detecting cheaters; it needs some improvements"* (Q54).

Later, in December 2024, Rec Room announced Referee, a proprietary, user-level, in-house AC framed as a future replacement for EAC. We observe a significant rise in both positive and negative sentiment toward anti-cheat (AC Effective 1.8×, AC Ineffective 2.0×) following this announcement, with users anticipating the migration and writing negatively about EAC: *"I honestly can't wait to be rid of EAC lol; [I'm] just hoping Referee ends up being as good as they're making it out to be"* (Q10). At the same time, Linux users debated cross-platform implications: *"[It is] kind of frustrating that Rec Room dropped Linux support without saying a word"* (Q42).

In March 2025, Rec Room published its first Referee progress update, reporting that a server-side patch had blocked a large number of cheating attempts [60]. We observe a decrease in negative sentiment toward AC engines (AC Ineffective 0.59×) after this event, as users expressed optimism about Referee's effectiveness. For instance, one user noted, *"So glad we've got Referee now, [it's] easily the best anti-cheat we've ever had"* (Q71).

The migration was completed in May 2025, and EAC was retired. We observe a rise in negative sentiment toward AC engines (AC Ineffective 1.6×) after this retirement, with users directly attributing the perceived increase in mods to the loss of kernel-level enforcement. As one user noted, *"It's because of them pulling Easy Anti-Cheat off PC, and the replacement evidently isn't working"* (Q46).

Public Release of Mods, Tools, and AC Bypasses. We find that the public release of modified clients, mod tools and methods, and AC bypasses coincides with an increase in discussions about the harms of mods and the anti-cheat measures meant to contain them. In February 2018, a hacked VRChat SDK was released, enabling asset theft and disruption. This was followed by a rise in user recommendations for fixes (1.3×), such as *"cutting back the SDK"* and *"shipping an overlay menu and a report button"* (Q2, Q37).

Before 2020, VRChat ran on Unity's Mono scripting backend, allowing straightforward modding. In April 2020, migration to IL2CPP [77] compiled its code into C++ and stripped metadata, breaking all existing mods and mod loaders. Developers quickly pivoted to mod loaders with IL2CPP support (e.g., MelonLoader [46]), restoring code-injection capabilities. After the first mod [23] release in May 2020, we observe a rise in discussions on disruption (1.7×).

Similarly, a free public EAC bypass (January 2023) restored crashers and ban-evaders, with users describing a return to pre-EAC conditions: *"Feels like we've been dragged straight back to how it used to be, with an EAC bypass anyone can grab"* (Q68). It also coincided with a rise in discussions of anti-cheat's financial and monetization framing (1.8× for the AC Financial Motives theme), as users questioned the value of a paid engine that could be so readily circumvented. Other users defended EAC, with one noting that *"The nice thing about relying on a third-party anti-cheat is that the moment someone figures out a bypass, you've suddenly got an outside company (Epic Games) with a real incentive working on the fix"* (Q24), and another reporting that this bypass was patched within a day.

More recently, in May 2025, the release of an open-source modified client coincided with a rise in discussions of privacy violations against other users, particularly the IP harvesting and network-level attacks described in Section 4.4 (Q35, Q52).

Platform and AC Updates. We find that updates to the AC infrastructure are associated with an increase in discussions about platform compatibility and performance, even when the platform disclaims responsibility for the change. In December 2025, an EAC update blocked access for cloud gaming services (e.g., Shadow PC [69]) and users relying on them to access VRChat were unable to use the platform. We observe a rise in posts discussing limited cross-platform coverage (3.9×). For example, one user described *"I know a dozen-plus people who cannot play VRChat because they were using Shadow PC to access it"* (Q73). Another user started an appeal *"to convince [VRChat] into undoing the change"* (Q20). The platform denied responsibility: *"this isn't something on our end . . . please reach out to Shadow for help"* (Q57).

> **Key Takeaways (RQ3).** Users' concerns about the harms of client modifications are persistent, but sentiment toward AC engines shifts with each event in the modding and AC ecosystem. Sentiment is shaped by what the engine restricts, what harms emerge despite it, and what alternatives appear.

7 Discussion

Our analysis of user discourse on Reddit reveals users' concerns about client modifications in social XR and their perceptions of platforms' efforts to mitigate them. We discuss the key insights from our findings and synthesize recommendations for stakeholders.

7.1 Key Insights

Modified Clients Worsen User Harms in Social XR. The harms reported by users from client modifications have parallels in client-side software protection and online games. Client-side state is modifiable, cheating produces disruption and unfair advantage [34], creators have concerns about piracy or intellectual-property theft (Section 4.2), and modified clients distributed outside official channels carry malware and credential-stealing payloads (Section 4.5). However, we find that in social XR, the same attacks are experienced

differently because users are embodied through avatars in a live shared space. For example, embodiment changes what asset theft means. An avatar is the user's persistent face and body in shared space [29], so users describe a ripped avatar as identity theft and impersonation (Section 4.2). Similarly, interactions in social XR are perceived as physical due to immersion [71], so disruption attacks like induced lag and crashing (Section 4.1) are not merely performance issues but can cause real harm, such as motion sickness, seizures, and injury, or eject people from live social events.

Together, embodiment and immersion make harassment a persistent harm in social XR [8, 30, 68], which is why these platforms ship safety features through which users manage contact themselves [89]. Prior work finds those features widely adopted but reactive [13–15, 84]; we find that modified clients defeat them (e.g., unblockable clients, forced entry into private instances), so they fail even for users who apply them proactively.

ACs Emerge as the Dominant but Contested Defense. Online platforms have developed several defenses against client modifications, including server-side validation, integrity attestation, and AC engines [18]. Among these, ACs are the only ones users discuss substantively in our data. One interpretation is that effective non-user-facing defenses, when deployed, do not give users a reason to discuss them. Our findings show that ACs are partially effective at mitigating concrete harms users associate with client modifications (e.g., indiscriminate object spawning, casual crashing), but introduce new concerns about kernel-level surveillance, performance overhead, usability, and cross-platform support. A recurring source of frustration is the blanket nature of third-party commercial ACs (e.g., EAC): ACs ban both malicious mods and non-malicious useful mods (e.g., accessibility tools, anti-crasher mods), often leaving non-modding users worse off than before.

Mods and ACs in a Cat-and-Mouse Cycle. A recurring dynamic in our findings is that client modifications and the defenses deployed against them coevolve rather than reach equilibrium. We observe this cycle directly in our longitudinal analysis (Section 6): when VRChat's migration to the IL2CPP scripting backend broke all existing mods in April 2020, the modding community pivoted to an alternative within weeks, restoring code-injection capabilities. When a free public EAC bypass surfaced in January 2023, the platform issued a patch and ban wave within 24 hours, only for new bypasses to surface in subsequent months. Users themselves articulate this dynamic by comparing ACs to Digital Rights Management (DRM); both raise the cost of attack but cannot eliminate it: *"[EAC is like] DRM . . . all it does is postpone the inevitable"* (Q40).

This has two implications for defense design. First, AC adoption buys time; it does not settle the problem. Platforms that adopt AC must respond to new bypasses, not treat deployment as finished. Second, the cycle shifts the cost of each defense iteration onto non-modding users. Performance regressions, false positives, and accessibility losses fall on users the AC was deployed to protect.

7.2 Recommendations

For Platforms. Recommendations for platforms can be split into two groups: (i) recommendations from users (Section 5.3) and (ii) recommendations from parallel technologies and the broader security and privacy literature. Users recommend official first-party mods, stronger account-level enforcement and bans, and server-side validation. Official first-party mods can help reduce community backlash against ACs by integrating the most popular third-party safety and accessibility mods as built-in features, and stricter bans (e.g., hardware ID and IP bans) can deter users from modding or attempting to bypass ACs. While server-side validation is the most effective defense, it is also the least tractable given that social XR admits a large range of valid actions and XR platforms have stringent latency requirements to maintain perceptual stability [22].

Beyond user-recommended defenses, we synthesize recommendations based on the broader security literature. First, platforms should adopt defense-in-depth strategies and combine multiple defenses such as integrity checks and remote attestation alongside AC to prevent tampering and implement server validation for select sensitive actions (e.g., safety features). Second, platforms should be transparent about the security rationale motivating AC adoption. Users attribute monetization motives to AC decisions (Section 5.4). Platforms can mitigate this perception by clearly communicating the threats they aim to address and the tradeoffs involved.

For Users. Users should avoid installing unvetted modified clients, which are common vectors for credential theft, malware, and account hijacking (Section 4.5). For features users value in third-party mods (avatar search, performance optimization, accessibility), users should check whether the platform offers comparable first-party features. Users concerned about harms from modified clients should consider using unlisted private instances when possible, though this trades community size and event access for safety.

For Policymakers. For social XR use cases with sensitive impact (e.g., K-12 education, professional training), policymakers should establish guidelines on which platforms are permissible. Our findings show that mainstream XR platforms carry known harms from modified clients that may be unacceptable in regulated contexts.

7.3 Limitations and Future Work

First, we focus on two social XR platforms (VRChat and Rec Room). While they are two of the most popular platforms by active user base [35, 74], findings may not generalize to other platforms. Second, we only analyze data from Reddit, whose users are typically more technical and less diverse than the general user base. Our corpus nonetheless spans a wide range of technical sophistication, from posts discussing kernel access and mod internals to posts asking why the app will not launch after an update or reporting a crash without knowing its cause. We also checked other platforms (e.g., Steam Community, Discord) through keyword searches and observed similar concerns, though a multi-platform analysis would strengthen our findings. Moreover, as with any pseudonymous online forum, we cannot characterize user demographics. Third, our automated labeling pipeline uses a DeBERTa-based relevance classifier and an LLM-based multi-label classifier. While both achieved high accuracy and F1-score, labeling errors could introduce noise by shifting a theme's measured frequency. However, error rates are mostly low and bounded (e.g., disruption at 0.95 F1, asset theft at 0.93). We also manually confirmed every significant event-label association (Section 3.3.3), so the findings in Section 6 are not artifacts of labeling errors. Finally, our candidate event list for the longitudinal analysis is not exhaustive. Our goal, however, is not to

compile the most comprehensive event list but to study how user concerns and discourse change across event types.

Our work also surfaces directions for future research. First, alternative defenses, including server-side validation and behavioral detection, have been proposed in the gaming domain [16, 88], but their applicability to social XR is unexplored and presents distinct challenges, such as a larger search space of valid actions and stricter latency requirements for XR [22]. Second, the privacy implications of commercial kernel-level AC engines, widely debated by users (Section 5.2), remain empirically uncharacterized. Rigorous technical evaluation of what these engines collect would help users and platforms make informed decisions about whether the kernel-level access ACs require is justified by the harms they mitigate.

8 Conclusion

We conducted a large-scale analysis of user discourse on Reddit to examine perceptions of client modifications and defenses against them in social XR. Our analysis shows that client modifications worsen known harms in social XR, such as harassment and stalking, and introduce new ones, such as disruption and asset theft. Because users are embodied through avatars and present in a live shared space, they experience these harms as violations of identity and safety. While AC engines are partially effective, they introduce privacy, usability, and performance tradeoffs that fall on users who never installed a modification. User sentiment toward ACs isn't static; it shifts with events in the modding and AC ecosystem.

Acknowledgments

We thank our anonymous reviewers and shepherd for providing us with valuable feedback that helped improve our paper. This research has been partially supported by the National Science Foundation (NSF) through grant CNS-2544561. The findings and conclusions presented in this paper are solely those of the authors and do not necessarily represent the views of the NSF.

References

[1] Devon Adams, Alseny Bah, Catherine Barwulor, Nureli Musaby, Kadeem Pitkin, and Elissa M Redmiles. 2018. Ethics emerging: the story of privacy and security perceptions in virtual reality. In *Symposium on Usable Privacy and Security (SOUPS)*.

[2] Mutahar Ali, Arjun Arunasalam, and Habiba Farrukh. 2025. Understanding Users' Security and Privacy Concerns and Attitudes Towards Conversational AI Platforms. In *IEEE Symposium on Security and Privacy (SP)*.

[3] Mutahar Ali and Habiba Farrukh. 2026. Relay and Betray: Exploiting Client-Side Authority in Multi-User Mixed Reality. In *USENIX Security Symposium*.

[4] John W Ayers, Theodore L Caputi, Camille Nebeker, and Mark Dredze. 2018. Don't quote me: reverse identification of research participants in social media studies. *NPJ Digital Medicine* (2018).

[5] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *International AAAI Conference on Web and Social Media (ICWSM)*.

[6] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)* (1995).

[7] James Lopez Bernal, Steven Cummins, and Antonio Gasparrini. 2017. Interrupted time series regression for the evaluation of public health interventions: a tutorial. *International Journal of Epidemiology* (2017).

[8] Lindsay Blackwell, Nicole Ellison, Natasha Elliott-Deflo, and Raz Schwartz. 2019. Harassment in social virtual reality: Challenges for platform governance. *Proceedings of the ACM on Human-Computer Interaction* (2019).

[9] Arianna Boldi and Amon Rapp. 2024. "Is It Legit, To You?". An Exploration of Players' Perceptions of Cheating in a Multiplayer Video Game: Making Sense of Uncertainty. *International Journal of Human–Computer Interaction* (2024).

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* (2006).

[11] Jiaxun Cao, Anupam Das, Pardis Emami-Naeini, et al. 2024. Understanding parents' perceptions and practices toward children's security and privacy in virtual reality. In *IEEE Symposium on Security and Privacy (SP)*.

[12] Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You can't stay here: The efficacy of Reddit's 2015 ban examined through hate speech. *Proceedings of the ACM on Human-Computer Interaction* (2017).

[13] Qijia Chen, Jie Cai, and Giulio Jacucci. 2024. "People are way too obsessed with rank": Trust system in social virtual reality. *Computer-Supported Cooperative Work (CSCW)* (2024).

[14] Qijia Chen, Seyed Mahed Mousavi, Giuseppe Riccardi, and Giulio Jacucci. 2025. Investigating the use and perception of blocking feature in social virtual reality spaces: A study on discussion forums. *Proceedings of the ACM on Human-Computer Interaction* (2025).

[15] Qijia Chen, Qunfang Wu, and Giulio Jacucci. 2025. Democratic moderation: Exploring the use and perception of votekicking in social virtual reality. In *Conference on Human Factors in Computing Systems (CHI)*.

[16] Minyeop Choi, Gihyuk Ko, and Sang Kil Cha. 2023. BotScreen: Trust Everybody, but Cut the Aimbots Yourself. In *USENIX Security Symposium*.

[17] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* (1960).

[18] Sam Collins, Alex Poulopoulos, Marius Muench, and Tom Chothia. 2024. Anti-cheat: Attacks and the effectiveness of client-side defences. In *Workshop on Research on Offensive and Defensive Techniques in the Context of Man At The End Attacks (CheckMATE)*.

[19] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).

[20] Elmira Deldari, Diana Freed, Julio Poveda, and Yaxing Yao. 2023. An investigation of teenager experiences in social virtual reality from teenagers', parents', and bystanders' perspectives. In *Symposium on Usable Privacy and Security (SOUPS)*.

[21] Christoph Dorner and Lukas Daniel Klausner. 2024. If it looks like a rootkit and deceives like a rootkit: A critical examination of kernel-level anti-cheat systems. In *International Conference on Availability, Reliability and Security (ARES)*.

[22] Mohammed S Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. 2018. Toward low-latency and ultra-reliable virtual reality. *IEEE Network* (2018).

[23] emmVRC. 2026. emmVRC mod for VRChat. https://github.com/emmVRC. Accessed: 29-Apr-2026.

[24] Epic Games Inc. 2025. Easy Anti-Cheat — Industry-leading anti-cheat service. https://www.easy.ac/en-US. Accessed: 2025-09-11.

[25] Habiba Farrukh, Reham Mohamed, Aniket Nare, Antonio Bianchi, and Z Berkay Celik. 2023. LocIn: Inferring semantic location from spatial maps in mixed reality. In *USENIX Security Symposium*.

[26] Casey Fiesler, Nathan Beard, and Brian C Keegan. 2020. No robots, spiders, or scrapers: Legal and ethical regulation of data collection methods in social media terms of service. In *International AAAI Conference on Web and Social Media (ICWSM)*.

[27] Casey Fiesler, Michael Zimmer, Nicholas Proferes, Sarah Gilbert, and Naiyan Jones. 2024. Remember the human: A systematic review of ethical considerations in reddit research. *Proceedings of the ACM on Human-Computer Interaction* (2024).

[28] Aline Shakti Franzke, Anja Bechmann, Charles Melvin Ess, and Michael Zimmer. 2020. Internet research: Ethical guidelines 3.0.

[29] Guo Freeman and Divine Maloney. 2021. Body, avatar, and me: The presentation and perception of self in social virtual reality. *Proceedings of the ACM on Human-Computer Interaction* (2021).

[30] Guo Freeman, Samaneh Zamanifard, Divine Maloney, and Dane Acena. 2022. Disturbing the peace: Experiencing and mitigating emerging harassment in social virtual reality. *Proceedings of the ACM on Human-Computer Interaction* (2022).

[31] Google. 2026. Abuse Monitoring, Gemini API. https://ai.google.dev/gemini-api/docs/usage-policies Accessed: 2026-08-10.

[32] Google DeepMind. 2026. Gemini 3.1 Flash Lite. https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/ Accessed: 2026-09-12.

[33] Hilda Hadan, Derrick M Wang, Lennart E Nacke, and Leah Zhang-Kennedy. 2024. Privacy in immersive extended reality: Exploring user perceptions, concerns, and coping strategies. In *Conference on Human Factors in Computing Systems (CHI)*.

[34] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. 2022. Cheating and detection method in massively multiplayer online role-playing game: systematic literature review. *IEEE Access* (2022).

[35] Scott Hayden. 2022. Virtual Social Platform 'Rec Room' Hits 3 Million Monthly Active VR Users. https://www.roadtovr.com/rec-room-monthly-active-vr-users-3-million-peak/. Road to VR. Accessed: 2026-04-26.

[36] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using electra-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*.

[37] Yosef Hochberg and Ajit C Tamhane. 1987. *Multiple comparison procedures*. John Wiley & Sons, Inc.

[38] Taehun Kim, Hyeonmin Ha, Seoyoon Choi, Jaeyeon Jung, and Byung-Gon Chun. 2017. Breaking ad-hoc runtime integrity protection mechanisms in Android financial apps. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*.

[39] Philip Klostermeyer, Sabrina Klivan, Sandra Höltervennhoff, Alexander Krause, Niklas Busch, and Sascha Fahl. 2024. Skipping the security side quests: A qualitative study on security practices and challenges in game development. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[40] Jingjie Li, Kaiwen Sun, Brittany Skye Huff, Anna Marie Bierley, Younghyun Kim, Florian Schaub, and Kassem Fawaz. 2023. "It's up to the Consumer to be Smart": Understanding the Security and Privacy Attitudes of Smart Home Users on Reddit. In *IEEE Symposium on Security and Privacy (SP)*.

[41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[42] Anton Maario, Vinod Kumar Shukla, A Ambikapathy, and Purushottam Sharma. 2021. Redefining the risks of kernel-level anti-cheat in online gaming. In *International Conference on Signal Processing and Integrated Networks (SPIN)*.

[43] Annette Markham. 2012. Fabrication as ethical practice: Qualitative inquiry in ambiguous internet contexts. *Information, Communication & Society* (2012).

[44] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. *Proceedings of the ACM on Human-Computer Interaction* (2019).

[45] Joshua McVeigh-Schultz, Elena Márquez Segura, Nick Merrill, and Katherine Isbister. 2018. What's it mean to" Be Social" in VR? Mapping the social VR design ecology. In *ACM Conference Companion Publication on Designing Interactive Systems (DIS)*.

[46] MelonLoader. 2026. Universal Mod Loader for Unity Games. https://melonloader.co/. Accessed: 29-Apr-2026.

[47] Andrea Mengascini, Ryan Aurelio, and Giancarlo Pellegrino. 2024. The Big Brother's New Playground: Unmasking the Illusion of Privacy in Web Metaverses from a Malicious User's Perspective. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[48] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems* (1994).

[49] Iman Mohammadi, Mutahar Ali, and Habiba Farrukh. 2026. Understanding Users' Perceptions of Client-Side Modifications and Anti-Cheat Engines in Social XR. https://doi.org/10.5281/zenodo.22748459

[50] Chandrika Mukherjee, Reham Mohamed, Arjun Arunasalam, Habiba Farrukh, and Z Berkay Celik. 2025. Shadowed Realities: An Investigation of UI Attacks in WebXR. In *USENIX Security Symposium*.

[51] Vivek Nair, Wenbo Guo, Justus Mattern, Rui Wang, James F. O'Brien, Louis Rosenberg, and Dawn Song. 2023. Unique Identification of 50,000+ Virtual Reality Users from Head & Hand Motion Data. In *USENIX Security Symposium*.

[52] Vivek Nair, Christian Rack, Wenbo Guo, Rui Wang, Shuixian Li, Brandon Huang, Atticus Cull, James F O'Brien, Marc Latoschik, Louis Rosenberg, et al. 2024. Inferring private personal attributes of virtual reality users from ecologically valid head and hand motion data. In *IEEE VR Abstracts and Workshops (VRW)*.

[53] Preksha Nema, Pauline Anthonysamy, Nina Taft, and Sai Teja Peddinti. 2022. Analyzing user perspectives on mobile app privacy at scale. In *International Conference on Software Engineering (ICSE)*.

[54] Whitney K. Newey and Kenneth D. West. 1987. A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica* (1987).

[55] OpenAI. 2026. Data Controls in the OpenAI Platform. https://developers.openai.com/api/docs/guides/your-data. Accessed: 2026-08-10.

[56] OpenAI. 2026. GPT-5.4 Thinking. https://openai.com/index/gpt-5-4-thinking-system-card/ Accessed: 2026-09-12.

[57] OpenAI and Reddit. 2024. OpenAI and Reddit Partnership. https://openai.com/index/openai-and-reddit-partnership/ Accessed: 2026-08-14.

[58] Thomas V. Perneger. 1998. What's Wrong with Bonferroni Adjustments. *BMJ* (1998).

[59] Nicholas Proferes, Naiyan Jones, Sarah Gilbert, Casey Fiesler, and Michael Zimmer. 2021. Studying Reddit: A systematic overview of disciplines, approaches, methods, and ethics. *Social Media + Society* (2021).

[60] Rec Room Inc. 2025. Referee Update. https://blog.recroom.com/posts/refereeupdate. Accessed: 2026-04-26.

[61] Rec Room Inc. 2026. Rec Room. https://recroom.com/. Accessed: 2026-04-26.

[62] Rec Room Wiki Contributors. 2026. Timeline. https://rec-room.fandom.com/wiki/Timeline. Rec Room Wiki, Fandom. Accessed: 2026-04-26.

[63] Reddit. 2024. Expanding our Partnership with Google. https://redditinc.com/news/reddit-and-google-expand-partnership Accessed: 2026-08-14.

[64] Reddit, Inc. 2026. Reddit. https://www.reddit.com/. Accessed: 2026-04-26.

[65] Road to VR. 2026. Road to VR: Virtual Reality News. https://www.roadtovr.com/ Accessed: 2026-04-30.

[66] Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. 2018. Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & Quantity* (2018).

[67] Abhinaya SB, Abhishri Agrawal, Yaxing Yao, Yixin Zou, and Anupam Das. 2025. "What are they gonna do with my data?": Privacy Expectations, Concerns, and Behaviors in Virtual Reality. *Proceedings on Privacy Enhancing Technologies (PoPETs)* (2025).

[68] Kelsea Schulenberg, Guo Freeman, Lingyuan Li, and Catherine Barwulor. 2023. "Creepy Towards My Avatar Body, Creepy Towards My Body": How Women Experience and Manage Harassment Risks in Social Virtual Reality. *Proceedings of the ACM on Human-Computer Interaction* (2023).

[69] Shadow SAS. 2026. Shadow: Cloud Gaming PC Service. https://shadow.tech/us/ Accessed: 2026-04-30.

[70] Vamsi Shankar Simhadri, Yichang Xiong, Habiba Farrukh, and Xiaokuan Zhang. 2025. Virtual reality, real problems: A longitudinal security analysis of VR firmware. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[71] Mel Slater. 2009. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society B: Biological Sciences* (2009).

[72] Carter Slocum, Yicheng Zhang, Nael Abu-Ghazaleh, and Jiasi Chen. 2023. Going through the motions: AR/VR keylogging from user head motions. In *USENIX Security Symposium*.

[73] Carter Slocum, Yicheng Zhang, Erfan Shayegani, Pedram Zaree, Nael Abu-Ghazaleh, and Jiasi Chen. 2024. That Doesn't Go There: Attacks on Shared State in Multi-User Augmented Reality Applications. In *USENIX Security Symposium*.

[74] SteamDB. 2026. VRChat · SteamDB. https://steamdb.info/app/438100/. Accessed: 2026-04-26.

[75] Pasquale Stirparo, Igor Nai Fovino, Marco Taddeo, and Ioannis Kounelis. 2013. In-memory credentials robbery on android phones. In *World Congress on Internet Security (WorldCIS)*.

[76] Zihao Su, Kunlin Cai, Reuben Beeler, Lukas Dresel, Allan Garcia, Ilya Grishchenko, Yuan Tian, Christopher Kruegel, and Giovanni Vigna. 2024. Remote Keylogging Attacks in Multi-User VR Applications. In *USENIX Security Symposium*.

[77] Unity. 2026. IL2CPP: Intermediate Language To C++. https://docs.unity3d.com/Manual/scripting-backends-il2cpp.html Accessed: 2026-08-14.

[78] Amanda Viescinski, Tiago Heinrich, Vinicius Fulber-Garcia, and Carlos Maziero. 2025. How Risky Is It? A Closer Look at Game Anti-Cheat Software. In *IEEE Symposium on Computers and Communications (ISCC)*.

[79] Martin Vondráček, Ibrahim Baggili, Peter Casey, and Mehdi Mekni. 2023. Rise of the Metaverse's Immersive Virtual Reality Malware and the Man-in-the-Room Attack & Defenses. *Computers & Security* (2023).

[80] VRChat. 2022. VRChat 2022.2.2 Release Notes. https://docs.vrchat.com/docs/vrchat-202222. Accessed: 2026-02-11.

[81] VRChat. 2026. Creating Your First Avatar. https://creators.vrchat.com/avatars/creating-your-first-avatar/ Accessed: 13-Sep-2026.

[82] VRChat Inc. 2026. VRChat. https://hello.vrchat.com/. Accessed: 2026-04-26.

[83] VRChat Inc. 2026. VRChat Official Developer Updates. https://ask.vrchat.com/c/official/dev-updates/32 Accessed: 2026-04-30.

[84] Maheshya Weerasinghe, Shaun Macdonald, Cristina Fiani, Joseph O'Hagan, Mathieu Chollet, Mark McGill, and Mohamed Khamis. 2025. Beyond mute and block: adoption and effectiveness of safety tools in social VR, from ubiquitous harassment to social sculpting. *IEEE Transactions on Visualization and Computer Graphics* (2025).

[85] Miranda Wei, Sunny Consolvo, Patrick Gage Kelley, Tadayoshi Kohno, Tara Matthews, Sarah Meiklejohn, Franziska Roesner, Renee Shelby, Kurt Thomas, and Rebecca Umbach. 2024. Understanding Help-Seeking and Help-Giving on Social Media for Image-Based Sexual Abuse. In *USENIX Security Symposium*.

[86] Y Combinator. 2026. Hacker News. https://news.ycombinator.com/. Accessed: 29-Apr-2026.

[87] Jeff Yan and Brian Randell. 2005. A systematic classification of cheating in online games. In *ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*.

[88] Jiayi Zhang, Chenxin Sun, Yue Gu, Qingyu Zhang, Jiayi Lin, Xiaojiang Du, and Chenxiong Qian. 2025. Identify as a human does: A pathfinder of next-generation anti-cheat framework for first-person shooter games. *IEEE Transactions on Information Forensics and Security (TIFS)* (2025).

[89] Qingxiao Zheng, Shengyang Xu, Lingqing Wang, Yiliu Tang, Rohan C Salvi, Guo Freeman, and Yun Huang. 2023. Understanding safety risks and safety design in social VR environments. *Proceedings of the ACM on Human-Computer Interaction* (2023).

[90] Chaoshun Zuo and Zhiqiang Lin. 2022. Playing without paying: Detecting vulnerable payment verification in native binaries of unity mobile games. In *USENIX Security Symposium*.

Table 1: Performance of our multi-label classifier (Gemini 3.1 Flash-Lite) on the test sample of 406 posts.

Label	Prec.	Rec.	F1
Disruption	0.929	0.973	0.950
Asset Theft	0.889	0.985	0.934
Safety Evasion	0.739	0.630	0.680
Private Info Access	0.667	0.824	0.737
Mod Side-Effects	0.588	0.769	0.667
AC Effective	0.825	0.853	0.839
AC Ineffective	0.913	0.909	0.911
Limited Platform Coverage	0.760	1.000	0.864
Performance Overhead	0.692	0.818	0.750
Loss of Useful Mods	0.844	0.950	0.894
AC Privacy Concerns	0.625	0.833	0.714
User Recommendations	0.733	0.938	0.882
AC Financial Motives	1.000	0.720	0.837
Overall	0.876	0.909	**0.888**

Table 2: Events significantly associated with discourse on client modifications and anti-cheat engines, including release of mods, tools, and AC bypasses, commercial AC adoption and enforcement, migration from commercial kernel-level AC to in-house user-level AC, and AC updates. Uncolored rows are context for later events.

Date	Event
Feb 2018	**Hacked SDK Released.** A modified VRChat SDK spreads publicly and enables avatar ripping and disruption.
Apr 2020	**VRChat IL2CPP Migration.** VRChat moves from Mono to IL2CPP scripting backend [77], breaking all existing mods.
Apr 2020	**Rec Room Adopts EAC.** Rec Room deploys Easy Anti-Cheat against modified clients [62].
May 2020	**Post-IL2CPP Modding Revival.** First IL2CPP-compatible mod [23] ships on MelonLoader [46], restoring code injection.
Jul 2022	**VRChat EAC Integration.** VRChat adopts EAC, a commercial kernel-level engine [80].
Dec 2022	**VRChat Post-EAC Ban Wave.** VRChat issues account terminations and hardware ID bans.
Jan 2023	**Public Free EAC Bypass.** A free EAC bypass is released publicly, and restores crashing and ban evasion.
Dec 2024	**Rec Room Referee Announced.** Rec Room announces Referee, an in-house user-level engine to replace EAC.
Mar 2025	**Referee Progress Update.** Rec Room reports that a server-side patch blocked a large number of cheating attempts [60].
May 2025	**VRChat Crasher Released.** An open-source modified client with crashing and IP harvesting is released.
May 2025	**Referee Replaces EAC.** Rec Room retires EAC and completes the migration to Referee [62].
Dec 2025	**EAC Blocks Cloud Gaming.** An EAC update flags cloud PC services as virtual machines and blocks access.

Generative AI Usage

Generative AI was only used for editorial purposes (e.g., grammar fixes). All outputs were reviewed by the authors, and all ideas and technical contributions are the authors' own.

Ethical Considerations

We follow established guidelines for research on public online data [4, 26, 27], and all data were collected for non-commercial academic purposes. Our Institutional Review Board determined the study to be IRB-exempt because it uses only publicly available information and does not involve human subjects. The determination also covered our use of LLMs on the post text.

Data and Reporting. All data are public Reddit content from public dumps [5, 64]. We did not interact with users or access non-public data. Before analysis, we dropped links and account metadata and replaced usernames and post IDs with random IDs, retaining only post text, timestamp, and subreddit. We processed all data on institution-owned machines, apart from the text sent to the labeling APIs. We assign each post a pseudonymous ID [43] and paraphrase text to prevent re-identification through reverse search [4].

LLM Use. We used LLMs (Gemini and GPT) for post-hoc labeling of posts (Section 3.3.2). We sent only post text to the API and used each provider's paid API tier. Neither provider trains on paid-tier content, and each retains inputs only briefly for abuse monitoring [31, 55]. Reddit also licenses its public corpus to both providers [57, 63], so this text was already available to them through these agreements.

Community Policies and Benefits. We reviewed the rules of the four subreddits and found no restriction on using public posts for research. We contacted the moderators of the subreddits after the study. Two of them responded, showing interest in the findings, and did not raise any objections. Following Reddit's policy, we do not redistribute Reddit data or any derived products (e.g., the trained relevance classifier). Because obtaining individual consent for research on public posts at this scale is generally infeasible, established guidance emphasizes minimizing traceability and potential harm [4, 26–28, 59]. We do so by paraphrasing text, removing metadata, and reporting aggregates. Finally, understanding how users experience harms can help platforms identify and address them, and we judged this potential benefit to justify the analysis.

Open Science

To support reproducibility while complying with Reddit's content policy, we release the following in our artifacts [49]: (i) codebook; (ii) keyword list and the manual annotation guide (Section 3.2); (iii) prompts for the LLM-based classifier; (iv) full list of candidate events; (v) model comparison results for both classifiers; (vi) results for the regression analysis at different window sizes; and (vii) our code for training the relevance classifier, running the LLM-based labeling, and fitting the regression. We do not release Reddit data or the trained classifier built on it due to Reddit's content policy.

A Multi-Label Classifier Performance

Table 1 reports the performance of our LLM-based multi-label classifier on the test sample of 406 posts, including overall and per-label values for precision, recall, and F1 score per class.

B Relevant Events

Table 2 shows the relevant events in the social XR ecosystem, grouped by event class (mod and bypass releases, commercial AC adoption, AC migrations, and platform and AC updates).

Understanding Users' Perceptions of Client-Side Modifications and Anti-Cheat Engines in Social XR

Iman Mohammadi*
University of California, Irvine
Irvine, California, USA
imohamm1@uci.edu

Mutahar Ali*
University of California, Irvine
Irvine, California, USA
mutahara@uci.edu

Habiba Farrukh
University of California, Irvine
Irvine, California, USA
habibaf@uci.edu

A Artifact Appendix

A.1 Abstract

Our artifact supports a large-scale study of user discourse on client-side modifications and anti-cheat (AC) engines in social XR. It consists of two components: (i) a public component archived on Zenodo, and (ii) a private component (for review only). The public component is licensed under CC BY 4.0 and consists of the codebook and annotation materials, the prompts for our LLM-based multi-label classifier, the list of events used in the longitudinal analysis, and the comparison results across classification models and regression window sizes. It also contains the Python code for training the relevance classifier, applying it to the corpus, running the LLM-based labeling, computing theme prevalence, and fitting the interrupted time-series regression. Following Reddit's content policy, Reddit data and derivative products such as the trained relevance classifier cannot be publicly redistributed. We therefore provide these materials in a private component (for review only) consisting of the Reddit data, the manually annotated, client-security-related, and LLM-labeled posts, and the trained relevance classifier. The artifact allows reproducing classifier metrics in Section 3, theme prevalence in Figure 2, and event–theme associations in Figure 3.

A.2 Description & Requirements

The public component is a single archive (`Artifacts.zip`) on Zenodo containing:

- `README.md`: instructions for running each step.
- `codebook.csv`: codebook labels and their definitions.
- `keywords.csv`: keywords used to build the annotation sample.
- `labeling_guide.md`: guide followed by annotators.
- `labeling_prompts.py`: prompts for the multi-label classifier.
- `events.csv`: list of events used in the regression analysis.
- `requirements.txt`: Python package requirements.
- `relevance_classifier.py`: script for splitting the annotated posts and training and evaluating the relevance classifier.
- `apply_relevance_classifier.py`: script for running the trained classifier over the full corpus.
- `evaluate_relevance_classifier.py`: script for evaluating an existing relevance classifier checkpoint on the test split.
- `multilabel_classifier.py`: LLM-based multi-label classifier.
- `evaluate_labels.py`: script for evaluating the LLM-based multi-label classifier.
- `prevalence_analysis.py`: script for computing how often each codebook label appears.
- `regression_analysis.py`: script for interrupted time-series regression analysis.
- `plot_figure2.py`: script for regenerating Figure 2 from the computed prevalences.
- `plot_figure3.py`: script for regenerating Figure 3 from the retained associations.
- `results/`: results for comparison between different models for classification and different window sizes for the regression analysis, and the retained event-theme associations for each window size.

The private component is available to reviewers only and contains:

- `full_cleaned_dataset.csv`: cleaned corpus of ∼2.1M posts and comments from the four subreddits.
- `1250_relevance_annotated_posts.csv`: 1,250 posts manually annotated for relevance.
- `relevance_classifier_model/`: fine-tuned relevance classifier.
- `client_security_related_posts.csv`: the ∼71K posts identified as client-security related.
- `few_shot_examples.json`: few-shot examples used in the multi-label classifier prompts.
- `labeled_posts.csv`: the ∼71K posts with codebook labels applied.
- `250_multi-label-classifier_evaluation_posts.csv`: 250 manually annotated posts for evaluating the multi-label classifier.

A.2.1 Security, privacy, and ethical concerns. The artifact takes no destructive actions, requires no elevated privileges, and disables no security mechanisms. The private component contains public Reddit posts whose usernames and post IDs have been replaced with random identifiers and whose links and account metadata have been removed. Reviewers are asked not to redistribute or retain it after evaluation. Re-running the labeling step transmits post text to a third-party LLM API. In this case, the provider's paid tier should be used, which does not train on submitted content.

A.2.2 How to access. The public component is archived on Zenodo at https://doi.org/10.5281/zenodo.22748459 (version v4). The private component is distributed through the Artifact Credentials field of our HotCRP submission.

A.2.3 Hardware dependencies. A conventional machine with any modern CPU and about 8 GB of RAM is sufficient for validating

*These authors contributed equally to this work.

the reported results or for running the labeling, prevalence, and regression steps. Retraining the relevance classifier requires a CUDA GPU with at least 16 GB of memory.

A.2.4 Software dependencies. Python 3.10 or newer and the packages listed in `requirements.txt` (`transformers`, `torch`, `datasets`, `scikit-learn`, `statsmodels`, `google-genai`, and others). Re-running the multi-label classifier additionally requires a paid Gemini API key, supplied through the `GEMINI_API_KEY` environment variable.

A.2.5 Benchmarks. The datasets required by the experiments are the private-component files listed above. No external benchmarks.

A.3 Set Up

A.3.1 Installation. Download and extract `Artifacts.zip` from the Zenodo record, then copy the private component's files into a `data/` directory alongside the scripts, and install the dependencies:

```
pip install -r requirements.txt
export GEMINI_API_KEY=<key>   # for re-running labeling
```

A.3.2 Basic test. Run the prevalence script, which exercises the label parsing and needs no API key or GPU:

```
python prevalence_analysis.py
```

It prints the number of labeled posts, a table of the 13 themes with their counts and percentages, and writes `data/prevalence.csv`.

A.4 Evaluation Workflow

A.4.1 Major claims.

(C1): Our fine-tuned DeBERTa-v3-base relevance classifier identifies client-security-related posts with an F1-score of 0.91 ± 0.02, and reduces the ∼2.1M-post corpus to ∼71K posts. Reported in Section 3.2; validated by (E1).

(C2): Our LLM-based multi-label classifier, Gemini 3.1 Flash-Lite, applies the 13-theme codebook with a weighted F1-score of 0.94 on 250 unseen manually coded posts. Section 3.3.2; validated by (E2).

(C3): Labeling the ∼71K posts with the 13-theme codebook yields the theme prevalences reported in Figure 2 and quoted throughout Sections 4 and 5. Validated by (E3).

(C4): The ten events discussed in Section 6 are significantly associated with shifts in at least one theme, with the fold changes reported in Figure 3. Of the 45 associations at the ±14-day window, 42 remain significant at ±7 or ±30 days. Table 2 lists the events, validated by (E4).

A.4.2 Experiments. We provide the output of every costly step, so all four experiments can be validated on a CPU in minutes, with no GPU, no API key, and no cost. The GPU and API paths are optional.

(E1): *Relevance classification* [15 person-minutes using the pre-run output; ∼3 GPU-hours if run from scratch]. `python relevance_classifier.py` trains the classifier. It needs a GPU with at least 16 GB of memory and takes ∼ 10 minutes. Alternatively, we provide the trained model (`data/relevance_classifier_model/`) and its scores (`results/model_comparison/relevance_classifier.csv`). `python evaluate_relevance_classifier.py` scores a checkpoint on the test split without retraining. It recomputes the same split, so the numbers match those from the training script. `python apply_relevance_classifier.py` applies the classifier to the ∼ 2.1M posts and writes the ∼71K client-security-related posts to `data/client_security_related_posts.csv`. This needs a GPU and takes ∼ 3 hours. Alternatively, we provide the output (`data/client_security_related_posts.csv`).

(E2): *Multi-label classification* [15 person-minutes using the pre-run output; > 10 hours and API cost if run from scratch]. `python multilabel_classifier.py` applies the codebook to the ∼71K posts. It needs `GEMINI_API_KEY`, makes ∼922K API calls. It caches progress so an interrupted run resumes from the last checkpoint. Alternatively, we provide the output (`data/labeled_posts.csv`). `python evaluate_labels.py` then scores those labels against `data/250_multi-label-classifier_evaluation_posts.csv`. This runs on a CPU in under a minute. Comparison scores for both models are in `results/model_comparison/`.

(E3): *Theme prevalence* [10 person-minutes + under 1 compute-minute]. `python prevalence_analysis.py` reads `data/labeled_posts.csv` and writes `data/prevalence.csv`, giving the percentage of labeled posts carrying each theme. Percentages are over posts carrying at least one label, so they sum to more than 100%. `python plot_figure2.py` regenerates Figure 2 from `data/prevalence.csv`.

(E4): *Interrupted time-series regression analysis* [20 person-minutes + under 5 compute-minutes]. `python regression_analysis.py --window 14` reads `data/labeled_posts.csv` and `events.csv` and writes `data/its_results_w14.csv` and `data/its_results_significant_w14.csv`. Repeat with `--window 7` and `--window 30`. It runs on a CPU in about a minute per window. Figure 3 plots the `fold_laplace` column. The script reports all significant associations across the 63 candidate events (175 at ±14 days). The 45 associations for the ten events in Section 6 (Table 2) were selected from these by the manual review described in Section 3.3.3. `results/regression_analysis/retained_associations_w14.csv` lists the retained associations with their fold change at window size 14. The pre-built heatmaps for all three window sizes are in `results/regression_analysis/`. `python plot_figure3.py --window 14` regenerates Figure 3 from the retained associations.

A.5 Notes on Reusability

The pipeline is independent of our corpus and research question. The relevance classifier reduces a large corpus to a topically relevant subset, the LLM-based multi-label classifier applies a qualitative codebook at scale, and the interrupted time-series regression relates the resulting labels to dated events. Together these form a general method for large-scale qualitative and longitudinal analysis of online discourse. Further, each stage reads and writes from files, so components can be replaced independently without disturbing the rest of the pipeline.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926.