ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

Shadowed Realities: An Investigation of UI Attacks in WebXR

Chandrika Mukherjee†, Reham Mohamed‡, Arjun Arunasalam†, Habiba Farrukh§, and Z. Berkay Celik†

† *Purdue University, {cmukherj, aarunasa, zcelik}@purdue.edu*
‡ *American University of Sharjah, raburas@aus.edu*
§ *University of California, Irvine, habibaf@uci.edu*

Abstract

WebXR is a standard web interface for extended reality that offers virtual environments and immersive 3D interactions, distinguishing it from the traditional web. However, these novel UI properties also introduce potential avenues for dark design exploitation. For instance, the absence of iframe-like elements in WebXR can be exploited by third parties, such as ad service providers, to inject JavaScript scripts and induce unintentional clicks or extract sensitive user information.

In this work, our objective is to identify and analyze the UI properties of WebXR vulnerable to exploitation by both first and third parties and to understand their impact on user experience. First, we examine vulnerable UI properties and propose five novel attack techniques that exploit one or more of these properties. We systematically categorize both existing and newly identified attacks within the advertising domain, to create a comprehensive taxonomy. Second, we design a user study framework to evaluate the impact of these attack categories employing dark designs on user experience. We develop a logging system to collect spatial data from 3D user interactions and integrate it with different WebXR applications that have different interaction needs. Additionally, we develop a set of metrics to derive meaningful insights from user interaction logs and assess how dark designs affect user behavior. Finally, we conduct a 100-participant between-subjects study using our user-study framework and survey.

Our findings suggest that most of these dark patterns go largely unnoticed by users while effectively achieving their intended goals. However, the impact of these designs varies depending on their category and application type. Our comprehensive taxonomy, logging framework, metrics, and user study results help developers review and improve their practices and inspire researchers to develop more robust defense mechanisms to protect user data in immersive platforms.

1 Introduction

WebXR is an open standard interface for extended reality (XR) platforms that enables users to explore virtual environments, manipulate digital objects, and engage with multimedia content directly through their web browser, eliminating the need for additional software or applications [48, 54]. The WebXR JavaScript API [62] offers a unified framework for AR/VR devices, allowing developers to access data from headsets and controllers and manage display output on compatible hardware. WebXR creates engaging experiences by offering users a more tactile and immersive way to interact with web content. Its web-based nature allows it to reach a wider audience without requiring additional software or devices. As a relatively new technology, WebXR presents unique opportunities for marketing campaigns to stand out, naturally drawing advertisers seeking innovative methods for brand promotion [64].

In standard web, a website publisher offers website space to ad service providers (e.g., Google, Facebook, etc.) to monetize their content. They use APIs from these providers, which embed ads using iframes [33] to isolate JavaScript execution from different origins. However, in WebXR, the absence of iframe-like isolation forces publishers to share portions of the immersive scene, potentially compromising control over individual content elements. This lack of isolation allows entities in the ad ecosystem to access the WebXR API, collect sensitive object data, and track user movements, posing significant privacy and security risks. Moreover, in XR, UI features such as consistent 360°unawareness, 3D object placement, and transparent objects can be exploited to trap users into unintended actions or apply dark patterns.

Prior works have extensively studied dark patterns across platforms such as traditional web, mobile, and IoT environments [11, 13, 27, 28, 41, 42], showing how manipulative interfaces can drive unintended user actions. However, these works primarily focus on 2D interfaces. Recent works [36, 61] have begun to explore dark patterns in XR environments, where immersive interfaces and spatial freedom heighten user vulnerability. Yet, these works overlook UI vulnerabilities unique to XR. Recent research [16, 39] has demonstrated attacks that exploit the absence of iframe isolation in WebXR, considering various entities within the ad ecosystem as potentially malicious. For instance, an interactive ad object is kept hidden inside another enticing object of the same size and shape, redirecting clicks to the ad, and generating revenue for the publisher. However, these works do not fully investigate the vulnerable UI properties, thereby overlooking other dark designs that could have potential security and privacy implications for users. Additionally, these works do not investigate the impact of these attacks on user experience.

Thus, it is essential to systematically analyze UI properties enabling dark patterns to safeguard immersive WebXR environments from malicious exploitation. Moreover, there is a limited understanding of how dark patterns affect user behavior and interactions. Given the variety of apps in WebXR, it is also important to determine whether the impact of these dark patterns differs based on the user's interaction specific to the app's context. Hence, in this paper, we focus on investigating various dark designs within the WebXR ad ecosystem, identifying the contributing sensitive UI properties, and analyzing their impacts on user behavior. Specifically, we aim to answer the following research questions:

RQ1: How do different UI properties contribute to the dark pattern designs in WebXR ad ecosystem?

RQ2: How do different deceptive design practices within WebXR impact the overall experience of users and their interaction with the app content?

To address these questions, we analyze nine existing attacks [16,39] in WebXR ad ecosystem through the lens of dark patterns [42]. In our analysis, we identify 14 UI properties, including two newly identified properties, that can potentially contribute to dark design practices. Through the manipulation of new and existing UI properties, we introduce five novel attacks within the WebXR ad ecosystem. Following this, we propose a taxonomy of attacks within the WebXR ad ecosystem, structured around the objectives of malicious entities.

To understand the impact of different attack categories outlined in the taxonomy, we conduct a between-subjects user study with 100 participants. First, we develop a logging framework for WebXR that collects spatial data from user interactions, allowing us to distinguish unintentional actions triggered by dark patterns and evaluate whether user behavior is influenced by these manipulative patterns. Second, we propose a set of four metrics derived from user interaction logs to assess user focus, disengagement from primary task, and unintentional interactions caused by the presence of dark patterns. Finally, we customize four WebXR apps of varying interaction levels (gaming, shopping reading, travel), integrating all the 14 attacks from the taxonomy and the logging framework to design our user study framework with 56 apps with dark designs and four apps as the control group.

Our study reveals that most attack categories employing dark designs go largely unnoticed by users. With the strategic placement of ads in dark pattern-integrated environments, users do not report significant UX discomfort related to ads. Attacks exploit these security-sensitive UI properties to achieve their malicious goals without disrupting the core functionality of the application, contributing to a consistent user experience, potentially masking the presence of these attacks. While most attacks in our taxonomy do not significantly impact user involvement and maintain a consistent user experience within the WebXR environment, their deceptive design still achieves the intended malicious outcomes. Moreover, we observe that user involvement varies depending on the level of interaction required by the app type. We also find that these attack categories are effective in coercing users into performing unintended actions, regardless of the app.

In summary, we make the following contributions:

- We identify security sensitive UI properties that contribute to dark designs in the WebXR ad ecosystem and propose five new attacks that exploit these properties.
- We develop a taxonomy of attacks that employ dark designs in the WebXR ad ecosystem, thoroughly highlighting the contributing UI properties.
- We develop a logging framework and interaction metrics for WebXR as part of our user study framework to aid in understanding the impacts of dark patterns on users.
- We conduct a 100-participant between-subjects user study to understand user perceptions of these dark patterns and their impact on user interaction behavior.

Responsible Disclosure. Given that our study on dark patterns in WebXR proposes new attacks and identifies contributing security-sensitive UI properties, we have disclosed both the attacks and potential mitigation strategies to Meta, WebXR, and A-Frame.

2 Background

WebXR. WebXR [62], is an open standard interface for accessing virtual reality (VR) and augmented reality (AR) experiences through web browsers on Android, iOS, VR (e.g., Meta Quest 2/3), and MR Head-Mounted Displays (HMDs) such as Apple Vision Pro and HoloLens 2. Unlike the standard web, which presents everything on a flat screen with the mouse as the primary interaction tool, WebXR offers a significantly more immersive experience. It enables users to engage in more realistic interactions in a 360° view with gaze cursor and hand controller support. A controller provides precise input and haptic feedback, enhancing control in immersive environments, while a gaze cursor enables hands-free, intuitive navigation via eye or head tracking, improving accessibility. These cursors support multiple events for different purposes, as we detail in Appendix A.1.

Developers leverage libraries such as Three.js [57], A-Frame [1], and Babylon.js [8] to develop immersive WebXR applications [46,54,58]. A-Frame is a preferred framework for WebXR development due to its ease of use and comprehensive feature set. It simplifies the creation of complex 3D VR and AR scenes with familiar HTML syntax and an intuitive, reusable entity-component system.

Dark Patterns. Dark patterns refer to user interface elements designed to deceive or manipulate users into performing actions unintentionally [13]. Researchers have thoroughly explored dark patterns in various computing platforms, including mobile [43], IoT [35, 38] and social media [27]. Recent studies have examined dark patterns emerging in XR systems [30, 36, 61]. These works highlight that existing dark patterns in other domains can have amplified effects in XR due to their super-realistic presentation, increased user engagement through XR device sensors, and a higher level of immersion provided by various HMDs.

Given the array of dark patterns in various domains, researchers have proposed generalized dark pattern taxonomies and systematic categorizations using six main attributes - Asymmetric, Covert, Deceptive, Information-Hiding, Restrictive, and Disparate Treatment [41, 42]. For instance, Covert (`C`) patterns subtly guide users toward specific actions while concealing the underlying influence mechanism. Deceptive (`D`) patterns foster false beliefs through misleading information or omissions. Information Hiding (`IH`) patterns obscure or delay the presentation of crucial information. Restrictive (`R`) patterns limit user choices. We use these dark patterns' attributes to investigate dark designs in the WebXR domain and to determine the contributing UI properties.

Research on dark patterns [35, 41, 56] indicates that these manipulative design strategies can result in a loss of autonomy – where users are influenced to make decisions they might not have made independently. Dark patterns impede informed choices and diminish user autonomy by obscuring or denying access to essential information. For example, cookie banners often undermine autonomy by employing tactics such as pre-selected options and confusing language to steer users towards accepting cookies [29]. Although cookies do not inherently cause loss of autonomy, websites can restrict user control to achieve this outcome. We investigate how malicious actors in the WebXR advertising ecosystem exploit various UI properties to subvert user decision-making, causing loss of autonomy. This manipulation can result in unintentional clicks and views, leading to security and privacy issues such as privacy leakage, financial loss, and malware downloads.

3 Motivation and Threat Model

Motivation. As noted in Section 2, WebXR differs significantly from the standard web in immersiveness and input methods. Unlike the standard web, WebXR lacks iframe-like isolation, allowing third-party entities to access APIs and retrieve data related to user interaction and virtual environment. WebXR, easily accessible through the web, enhances user engagement and has the potential to transform the ad ecosystem with innovative brand promotion methods [64]. The ad ecosystem consists of three main entities: publishers, advertisers, and ad service providers. Publishers monetize content by providing ad space to ad service providers, while advertisers bid for clicks and impressions to target relevant audiences and maximize campaign performance. Ad service providers display ads based on bids and relevance. In WebXR, publishers allocate parts of their scene areas for ads, unlike the standard web where iframes isolate third-party scripts.

Prior works [16, 39] have explored how entities in the ad ecosystem deceive users to benefit malicious actors by manipulating UI properties in WebXR. These studies propose various attacks with objectives such as generating revenue from ad clicks and impressions or extracting sensitive user information. These attacks identify the exploitation of WebXR UI properties, including the absence of iframes, object placement in the user's blind spot, same-space objects, transparency, and synthetic input. However, beyond these factors, the ability to programmatically capture screenshots without user consent, register clicks on the first clickable entity even if visually obscured, and render objects sequentially in WebXR provides malicious entities with opportunities to integrate dark patterns and manipulate users into performing unintended actions.

Therefore, it is essential to systematically investigate the UI properties that enable dark patterns in WebXR to protect immersive WebXR environments from malicious exploitation. In this paper, our goal is to explore vulnerable UI properties enabling novel attacks in the WebXR ad ecosystem and proposing a taxonomy categorizing attacks by objectives of adversaries and exploited UI properties.

To ensure safe and secure adoption of WebXR, it is also necessary to understand the impact of these attacks on user perception and their interaction behavior. Users may or may not perceive malicious intent, and the consequences can vary based on the app type and level of user interaction. Thus, we aim to investigate whether users can identify a loss of autonomy during interaction with WebXR interfaces that embed dark patterns. We also examine how different attack categories and apps differently impact the user experience.

Threat Model. The absence of iframe allows any entity in the ad ecosystem to directly access WebXR APIs and inject listeners for target events (detailed in Appendix A.1) to infer users' private information, alter scene content with false details, or trigger dynamic events on 3D objects, thereby disrupting primary operations. For instance, a malicious ad service provider could dynamically introduce invisible controllers and ray-casting events in a gaze-cursor-enabled environment, overriding the gaze-cursor's focus initiation. This manipulation suppresses critical visual cues or safety warnings, ultimately impairing the user's ability to make informed decisions. In our threat model, we assume that any of the three entities in the ad ecosystem can act as an adversary.

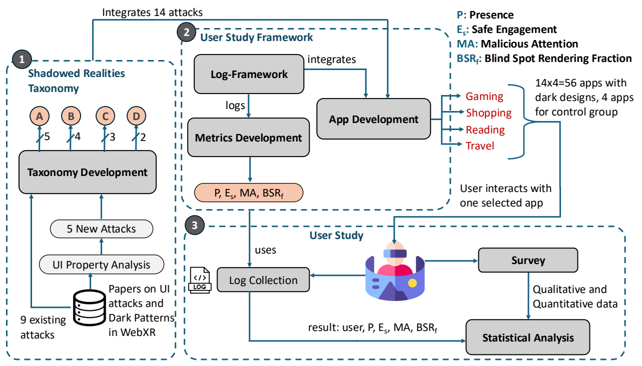


Figure 1: Overview of our method for analyzing the impact of UI attack categories on users within WebXR.

Malicious ad service providers may exploit the lack of code isolation and covert, deceptive, information-hiding, and restrictive dark design attributes to boost ad revenues, causing advertisers to pay for illegitimate activities. These actions can also erode trust in the ad market by creating unfair competition with other ad service providers. Malicious advertisers, meanwhile, may engage in "malvertising" to artificially increase their domains' popularity. Similarly, a malicious publisher could earn money from illegitimate ad clicks and impressions by strategically placing genuine ads within the scene.

Malicious activities by any party within the ad ecosystem can have detrimental effects on other entities, including users. Inadvertent clicks on malicious objects can lead to data theft, financial loss, and user profiling. Users may experience significant device performance issues, including slowdowns and crashes, as a consequence of malvertisement.

4 Approach Overview

Figure 1 presents an overview of our study methodology. To achieve our goal, we develop a taxonomy (❶) of attacks in the advertising context in WebXR, establishing four distinct attack categories. To do so, we uncover the 12 UI properties associated with the attacks proposed in previous works [16, 39] and propose two additional properties and five novel attacks within the ad ecosystem in WebXR.

To evaluate the impact of different attack categories on user behavior, we design a comprehensive user study framework (❷). This includes a logging framework designed to capture user interactions within the WebXR environment. The logging framework tracks various 3D user interactions, including users' positions, orientations, gaze-cursor movements, and controller inputs. By analyzing these logs, we propose four interaction metrics to derive valuable insights from user behavior: Presence (P), Safe Engagement (E_s), Malicious Attention (MA), and Blind Spot Rendering Fraction (BSR_f). Additionally, we evaluate user behavior across different app types.

Finally, we conduct a between-subjects user study (❸) with 100 participants to assess how the attack categories in our taxonomy impact user perception and interactions compared to a WebXR environment without attacks.

5 Taxonomy of WebXR UI Attacks

To answer RQ1 (Section 1), we introduce an attack taxonomy for UI attacks in the WebXR ad ecosystem. Attacks within each category are marked by the sensitive UI properties they exploit and their corresponding dark design attributes (D, IH, R, C). We map existing attacks to this taxonomy, and leverage sensitive UI properties to propose new attacks.

5.1 Systematic Literature Review

We collected papers on UI-based attacks specific to AR/VR by querying Google Scholar in November 2023 using the Scholarly[1] Python library. Leveraging CSRankings [19] and Google Scholar metrics[2], we identified top 23 venues in Engineering & Computer Science, specifically in Computer Security, and HCI, because of its relevance to attack threat models and AR/VR. The four key phrases "UI attack", "UI manipulation", "dark pattern", and "deception" were paired with each XR term ("virtual reality", "augmented reality", "mixed reality", "extended reality") and combined with selected venues, producing total 368 keywords resulting in 228 unique articles published since 2014.

Our study focused on UI-based attacks or manipulation strategies in AR/VR. Two authors reviewed and filtered papers based on abstracts, excluding surveys, SoK papers, studies unrelated to XR, and those not addressing UI-based deceptive design. This yielded 13 articles with high inter-rater agreement (Cohen's Kappa [31], $\kappa = 0.83$). We further excluded studies on emotional manipulation (e.g., hyper-personalization, false memory implantation), multi-user scenarios, those requiring application installations and dependent on ML models. Ultimately, we identified four papers [16, 17, 39, 61] proposing UI manipulation attacks in XR. Two of the studies [17, 61] focused on attacks utilizing UI properties that were already covered by Cheng et al. [16] and Lee et al. [39]. Hence, we selected attacks from the latter to identify security-sensitive UI properties and analyze their impact on users.

5.2 WebXR UI Properties

Prior works [16, 39] have exploited UI properties in WebXR to launch attacks with four main objectives: (1) perpetrating click fraud, (2) engaging in impression fraud by exploiting users' lack of awareness, (3) disrupting or compromising app's functionality, and (4) extracting users' sensitive data. By comprehensively studying the previous attacks, we determine 12 WebXR UI properties that contribute to these attacks.

[1] https://scholarly.readthedocs.io/en/stable/index.html
[2] https://scholar.google.com/citations?view_op=top_venues

Table 1: Security-sensitive UI properties (🟥 highlights two new properties that we identified).

ID	UI Property	Purpose	Use Case
P_1	Absence of iframe	Isolates third-party script execution.	Third-party content integration.
P_2	Transparency	Controls object opacity.	Visual effects (e.g., depth, fading, dimensionality, motion, flowing water, shattering glasses).
P_3	Synthetic input	Enables custom event emission on objects.	Dynamic object interactions.
P_4	Same space overlapping objects	Multiple objects occupy same space.	Complex scene design.
P_5	Event registration with only the event name	Event listeners registered by event name.	Generic event handling.
P_6	Cursor event sharing	Controllers added programmatically register events triggered by users through physical controllers.	Dynamic controller integration.
P_7	Blind spot	Objects persist behind user.	Strategic placement of objects out of direct view e.g., for exploration, navigation aids, dynamic lighting or shadow effects.
P_8	Auxiliary screens	Active display on HMD / connected device.	Debugging, user view sharing.
P_9	Scene entry/exit detection	Event listeners for scene transitions.	VR-friendly UI adjustments.
P_{10}	Sequential rendering	Maintains rendering order.	Optimizes performance.
P_{11}	Click reception by the first rendered object of fully overlapping objects	First rendered object receives click (if overlapping and of same size and shape).	Simulates real-world occlusion.
P_{12}	Click received by first clickable intersected entity	Clicks first intersected clickable object.	Prioritizes nearby objects in normal settings and simulates realistic interactions.
P_{13}	3D capture	Capturing programmatic screenshots of the 3D environment.	Automated testing, media sharing.
P_{14}	Gaze-fusing override	Prioritizes controller input over gaze.	Faster, more precise interactions.

These properties are used to employ dark pattern attributes, including deceptive (D), information hiding (IH), restrictive (R), and covert (C) in WebXR to achieve malicious intentions. We detail the description and usage of each of these UI properties in Table 1, annotated as P_1-P_{14} along with highlighting two new properties P_{13} and P_{14}.

Although these properties can be exploited for malicious purposes, each of them has legitimate use cases. For instance, sequential rendering (P_{10}) optimizes UI performance by sequentially loading content, but can be exploited to obscure objects. As in clickjacking [16], a malicious actor uses P_{10} to load a genuine ad beneath a bait object, collecting clicks to generate ad revenue. Similarly, the auxiliary screen (P_8) enables view sharing by displaying the user's VR perspective on a secondary screen (e.g., browser of a connected desktop), helping developers test and improve the interface. Yet, in the AAD [39] attack, adversaries exploit P_8 to display ads unnoticed by the user, generating revenue from ad impressions.

5.3 Shadowed Realities Attack Categorization

We develop our taxonomy, as detailed in Table 2, by focusing on four distinct categories of attacks, each aligned with the primary objectives of malicious actors operating within the WebXR ad ecosystem. This systematic classification clarifies the specific UI properties associated with each category, providing a structured framework to understand the various potential dark patterns in WebXR. Thus, our Shadowed Realities taxonomy introduces the four following categories:

Click Manipulation. This category includes attacks that use dark pattern attributes to trick users into unintended clicks, typically to generate fraudulent ad revenue. For instance, in the GCJ attack [39], a malicious ad service provider exploits the absence of iframe (P_1) to inject a JS script into the hosting origin, creating a fake cursor positioned near the real one. Subsequently, transparency (P_2) is used to hide the real cursor, deceiving (D) users into perceiving the fake cursor as genuine. Exploiting default event registration with only event name (P_5), the app cannot distinguish between clicks from the real and fake cursor, preserving functionality while generating ad revenue. The adversary thus conceals (IH) these malicious activities from both users and other entities in the ad ecosystem by leveraging P_1 and P_5. With the shared goal of generating ad revenue through fraudulent clicks, similar attacks like CCJ [39] and clickjacking [16] also belong to this category.

Peripheral Exploitation. This category leverages users' blind spots or areas outside their immediate focus to generate illicit ad impressions or clicks without the users' awareness. For instance, in the Input Forgery attack [16], a malicious publisher identifies whether the ad is located in the user's blind spot (P_7). The attacker then exploits the synthetic input property (P_3) to programmatically initiate clicks on the ad within the blind spot. Leveraging default event registration with only the event name (P_5), these synthetic clicks appear genuine to the system. By employing P_5 and P_7, this attack hides (IH) malicious activity from users and other entities within the ad ecosystem. P_3 creates a restrictive (R) dark pattern, as it denies the user control over the synthetic clicks. Similarly, attacks such as BST [39] and AAD [39] exploit user unawareness to generate ad revenue, placing them in this taxonomy section.

Functionality Disruption. This category encompasses attacks leveraging dark pattern attributes to deliberately disrupt app functionality, leading to potential user frustration, confusion, or unintended actions. For example, in the Object Erasure attack [16], a malicious competitive ad service provider exploits the absence of iframe (P_1) to extract information about a competitor's target XR content by injecting a JS script. It then employs the transparency (P_2) and same space overlapping (P_4) properties to modify or remove the target content, thereby erasing critical information such as safety warnings and disrupting the service's functionality. This attack exploits P_1 to conceal information (IH) from both the user and other entities within the ad ecosystem. P_2 and P_4 impose a restrictive (R) dark pattern attribute, depriving users of control over the altered content. These properties also deceive (D) users, creating false beliefs about competitor's XR content. Similarly, Denial-of-Service [16] attack use the same properties to disrupt the normal functionality of apps.

UI-based Privacy Leakage. This category covertly accesses or extracts sensitive user information, compromising privacy and enabling identity theft or other malicious actions. In the Intercepting User Inputs attack [16], a malicious ad service provider or advertiser (third-party entity) extracts private user information (e.g., passwords) for targeted marketing purposes. It exploits the absence of iframe (P_1) to inject a JS script into the host origin, introducing multiple transparent (P_2) and over-

Table 2: Taxonomy of attacks in WebXR ad ecosystem.

Category	Attack Name	Malicious Entity	Dark Design Attributes				Security Sensitive UI Properties													
			Covert	Deceptive	Info. Hiding	Restrictive	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}
Click Manipulation	GCJ [39]	Ad Service Provider		✓	✓		✓	✓			✓									
	CCJ [39]	Ad Service Provider		✓	✓		✓				✓	✓	✓							
	Clickjacking: Leveraging Inconsistency between Rendering and Interaction Orders [16]	Publisher	✓		✓					✓						✓	✓			
	Visual Overlapping	Publisher	✓	✓	✓				✓									✓		
	Sequential Rendering	Publisher		✓	✓			✓	✓	✓						✓		✓		
Peripheral Exploitation	BST [39]	Ad Service Provider			✓		✓						✓							
	AAD [39]	Ad Service Provider			✓		✓							✓	✓					
	Input Forgery: Leveraging Synthetic User Input [16]	Publisher			✓	✓			✓		✓		✓							
	Malvertising	Advertiser			✓	✓	✓	✓						✓	✓			✓		
Functionality Disruption	Denial-of-Service: Leveraging Invisibility [16]	Competitive Ad Service Provider			✓	✓	✓	✓		✓								✓		
	DoS through Overriding	Competitive Ad Service Provider		✓	✓	✓	✓	✓												✓
	Object Erasure: Leveraging Invisible Meshes [16]	Competitive Ad Service Provider		✓	✓	✓	✓	✓		✓										
UI-based Privacy Leakage	Intercepting User Inputs: Combining Invisible Objects and Synthetic User Input [16]	Any Third-Party Entity		✓	✓	✓	✓	✓	✓	✓	✓							✓		
	GUI Switch	Advertiser	✓	✓	✓		✓								✓				✓	

lapping (P_4) meshes over the keypad. These meshes capture user input (P_{12}), while the malicious script generates synthetic input (P_3) on the submit button using the captured data. The button authenticates the input without source validation, exploiting default event registration with only the event name (P_5), concealing the attack from users and other entities. P_1 and P_5 hide malicious activity (`IH`) from users and ad ecosystem entities. P_3 deceives users (`D`) by making the system appear normal. P_2, P_4, and P_{12} impose restrictive (`R`) dark pattern, forcing interaction through a transparent layer.

While all attacks in Table 2 involve information hiding (`IH`), it is not always essential for success but enhances effectiveness. For example, the BST attack [39] relies on P_1 and P_7 to hide ads for its success. Conversely, in the Intercepting User Inputs attack [16], P_1 and P_5 enhance UI manipulation but are not crucial for success. The attack relies primarily on transparent meshes to capture input and generate synthetic clicks on the PIN pad. Table 8 in the Appendix details how malicious entities within ad ecosystem exploit WebXR UI properties, their objectives, and associated consequences.

5.4 New Attacks in WebXR Ad Ecosystem

Beyond 12 sensitive properties employed by previous attacks, we further identified two additional UI properties (P_{13}, P_{14}) that can contribute to dark designs in WebXR. We now detail five new attacks that exploit combinations of the identified 14 security-sensitive UI properties. Based on their malicious objectives, we include these attacks in our taxonomy in Table 2 and visually illustrate their attack methods in Figure 2.

5.4.1 Malvertising

This attack involves advertisers exploiting sensitive UI properties to craft seemingly benign ads with malicious intent, integrated into legitimate web pages (shown in Figure 2-(a)).

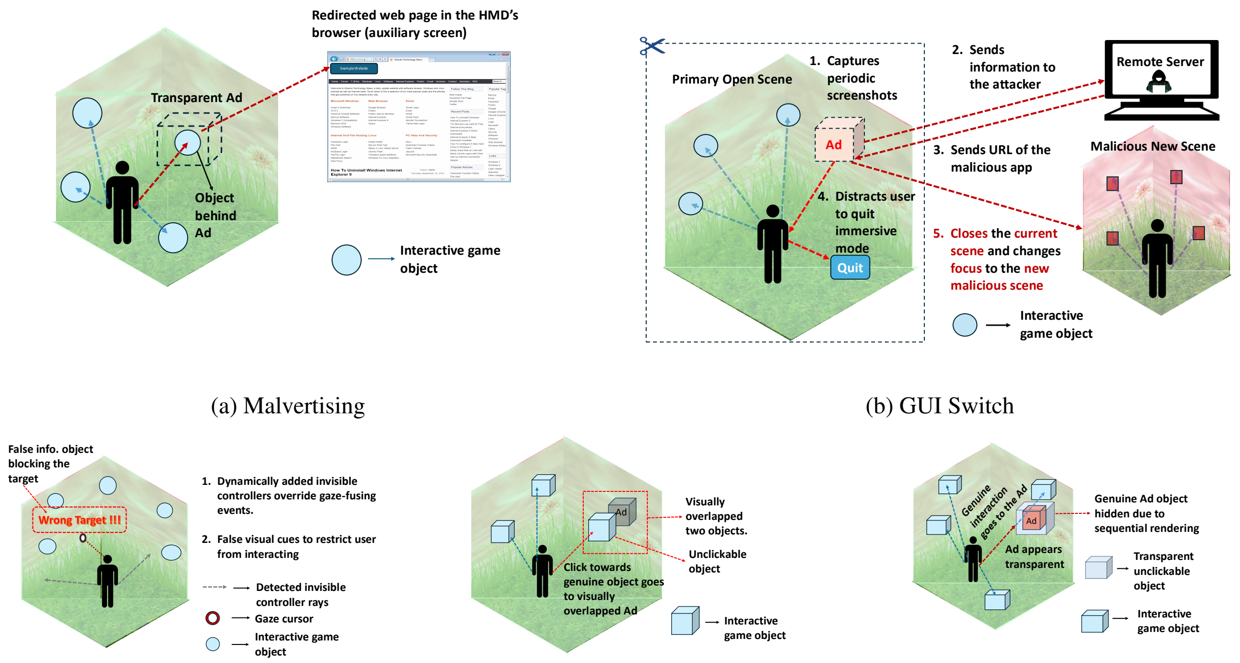
Attack Overview. Advertisers frame partially or fully transparent ads, visually overlaying them on other content. Even uninterested users, aiming to interact with underlying elements, inadvertently click the transparent ad. The transparency of the ad is then removed, preserving the apparent functionality. Upon entering the immersive mode, the 2D web page persists as an auxiliary screen in the HMD browser. The deceptive click triggers a redirection on this hidden screen, potentially also leading to drive-by downloads of malware. Users interacting with the immersive 3D environment remain unaware of the auxiliary screen's activity. Leveraging the detection of scene entry and exit in immersive environments, the attacker observes the user transitions and strategically terminates redirections to remain undetected. This can boost SEO rankings or video views, with advertisers potentially gaining first-party access to cookies for further tracking.

Exploited Properties. This attack relies on several UI properties: the absence of an iframe (P_1), transparency (P_2), click received by the first clickable intersected entity (P_{12}), a hidden auxiliary display (P_8), and scene entry/exit detection (P_9). By concealing redirection within the 2D auxiliary screen and removing user choice, this attack exhibits both information hiding (`IH`) and restrictive (`R`) attributes. Its reliance on the hidden auxiliary display places Malvertising under *Peripheral Exploitation* within our taxonomy.

5.4.2 GUI Switch

In this attack, malicious advertisers exploit dark pattern attributes to induce GUI confusion, forcing users to unwittingly interact with a malicious WebXR environment, thereby granting themselves first-party privileges (Figure 2-(b)).

Attack Overview. Leveraging the absence of an iframe, the adversary injects malicious JS script within the host page through an ad. It then captures periodic programmatic screenshots of the immersive environment, as a data URL without the user's explicit understanding to potentially identify page content. The capture of periodic screenshots leads to performance degradation, manifesting as slowdowns in the environment's loading time, particularly when the user is actively moving or rotating their head-mounted display (HMD). The


Redirected web page in the HMD's browser (auxiliary screen)
Transparent Ad
Object behind Ad
Interactive game object
Primary Open Scene
1. Captures periodic screenshots
2. Sends information to the attacker
Remote Server
3. Sends URL of the malicious app
Ad
4. Distracts user to quit immersive mode
Quit
Malicious New Scene
5. Closes the current scene and changes focus to the new malicious scene
Interactive game object
False info. object blocking the target
Wrong Target !!!
1. Dynamically added invisible controllers override gaze-fusing events.
2. False visual cues to restrict user from interacting
Detected invisible controller rays
Gaze cursor
Interactive game object
Ad
Visually overlapped two objects.
Unclickable object
Click towards genuine object goes to visually overlapped Ad
Interactive game object
Genuine interaction goes to the Ad
Ad
Genuine Ad object hidden due to sequential rendering
Ad appears transparent
Transparent unclickable object
Interactive game object

(a) Malvertising (b) GUI Switch

(c) DoS through Overriding (d) Visual Overlapping (e) Sequential Rendering

Figure 2: New attacks employing dark designs in WebXR ad ecosystem.

malicious code then employs visual or auditory distractions, further degrading the user experience. Frustrated users, driven by the false distraction, exit immersive mode, triggering scene exit detection at which point the attacker replaces the original page with a deceptive replica, forcing re-entry into a similar-looking compromised environment. This deceptive switch gives the attacker's page first-party status, enabling 3D data collection and user tracking.

Exploited Properties. This attack combines multiple UI properties: no iframe (P_1), periodic programmatic screenshots (P_{13}) to capture user's 3D immersive environment, scene entry/exit detection (P_9), and distractions. It deceives users into believing that the original page is malfunctioning (`D`) and hides malicious activity (`IH`). In addition, it covertly manipulates user behavior through distractions (`C`). The ultimate goal of extracting sensitive data about the primary service and virtual content classifies this attack under *UI-based Privacy Leakage* in our taxonomy.

5.4.3 DoS through Overriding

In this attack, a malicious adversary, such as a competing ad service provider, exploits UI dark patterns to disrupt the functionality of legitimate content in a gaze-based multi-ad service provider WebXR environment (Figure 2-(c)).

Attack Overview. Exploiting the lack of iframe isolation, the adversary introduces invisible controllers and associated raycasting (lines emitted from the controllers to interact with virtual objects) events targeting the competitor's XR content. This exploits the gaze-fusing override property, prioritizing these hidden controller raycasting events over the user-intended gaze-fusing events. This event is triggered when the user initiates focus on an object using the gaze cursor. However, taking advantage of the override, the adversary displays false information that blocks any interaction with the target XR content. Thus, essential visual cues or safety warnings from the authentic gaze-fusing event may be suppressed, hindering the user's ability to make informed decisions.

Exploited Properties. The attack combines the absence of an iframe (P_1), gaze-fusing override (P_{14}), and transparency (P_2) to achieve its disruptive goals. By concealing virtual controllers and their effects, it employs information hiding (`IH`). It shows false information to mislead users, thereby applying deception (`D`). Furthermore, by blocking the feedback from the gaze-fusing event, it provides users with limited/no choice, thereby applying the restrictive dark attribute (`R`). As the primary objective is to disrupt the functionality of a competing service, this attack falls under the category *Functionality Disruption* in our taxonomy.

5.4.4 Visual Overlapping

In this attack, the adversary is the publisher or developer of the WebXR environment and manipulates the user interface to encourage interaction with strategically placed objects to visually overlap with authentic ads. The user's intended click is then hijacked by the visually obscured ad, generating illegitimate ad revenue for the publisher (Figure 2-(d)).

Attack Overview. The adversary strategically places ads behind seemingly interactive objects, enticing users to click on them. However, these seemingly interactive objects are rendered non-interactive or unclickable. Because the "first clickable object receives the click" property, the click is intercepted by the hidden clickable ad. If the ad redirects to the auxiliary screen, the ad feedback remains unnoticed by the user immersed in the 3D environment. The primary functionality of the service remains unaffected, as the publisher can use the raycaster's origin and direction to determine whether it intersects with the bounding box (geometry) of the bait object. Consequently, the publisher can emit synthetic input on the bait object to maintain the illusion of normal interaction.

Exploited Properties. This attack renders the target object unclickable, ensuring the click is captured by the next intersected object (P_{12}), here the ad. Additionally, the publisher employs synthetic input (P_3) on the bait object to preserve its functionality. It employs deception (D) by misleading users about the consequences of their interactions. It is also covert (C) in its manipulation of user behavior for the adversary's gain. Furthermore, it hides this malicious practice from other entities (IH) in the ad network. Its primary goal of generating illegal ad clicks places it firmly within the *Click Manipulation* category of our taxonomy.

5.4.5 Sequential Rendering

This attack, also perpetrated by a malicious publisher, leverages sequential rendering and transparency to generate illegitimate ad clicks (Figure 2-(e)).

Attack Overview. The adversary creates a transparent, unclickable object and loads a genuine ad within it. Using sequential rendering, this ad, rendered later, becomes visually transparent, allowing users to see through it. However, because the "first clickable object receives the click" property, any interaction intended for objects behind the ad results in an unintentional click on the ad itself. The publisher maintains functionality by triggering synthetic clicks on the intended objects, leaving the user unaware of the deception.

Exploited Properties. This attack utilizes sequential rendering (P_{10}), first-clickable object (P_{12}), transparency (P_2), synthetic input (P_3), and same space (P_4) properties. It creates a false belief about user interactions or deceives (D) them and hides the mechanism from users and other entities (IH) in the ad ecosystem. Primarily aimed at generating illegal ad revenue, it falls under *Click Manipulation* in our taxonomy.

Table 3: Interactions within WebXR applications.

Interaction	Description
$\mathtt{HC}^{\mathtt{T_{obj}}}$	Click initiated by the human on $\mathtt{T_{obj}}$.
$\mathtt{HF}^{\mathtt{T_{obj}}}$	Focus initiated by the human on $\mathtt{T_{obj}}$.
$\mathtt{HC}^{\mathtt{DP_{obj}}}$	Click initiated by the human on $\mathtt{DP_{obj}}$.
$\mathtt{HF}^{\mathtt{DP_{obj}}}$	Focus initiated by the human on $\mathtt{DP_{obj}}$.
$\mathtt{UC}^{\mathtt{T_{obj}}}$	Click initiated unintentionally/programmatically on $\mathtt{T_{obj}}$.
$\mathtt{UF}^{\mathtt{T_{obj}}}$	Focus initiated unintentionally/programmatically on $\mathtt{T_{obj}}$.
$\mathtt{UC}^{\mathtt{DP_{obj}}}$	Click initiated unintentionally/programmatically on $\mathtt{DP_{obj}}$.
$\mathtt{UF}^{\mathtt{DP_{obj}}}$	Focus initiated unintentionally/programmatically on $\mathtt{DP_{obj}}$.

6 User Study Framework

To analyze the impact of diverse attack categories in our taxonomy (Table 2), we propose a user study framework. First, we introduce our log-framework, designed to capture granular interaction data from various WebXR environments, including user position, orientation, spatial coordinates, distance from cursors, and other relevant information. Subsequently, we develop interaction metrics, a suite of analytical methods, to extract meaningful insights from the collected logs. This enables us to address our research questions to analyze the impact of dark pattern integration in the WebXR environment. Finally, we present our collection of apps, which integrates the identified attacks in our taxonomy and the log-framework to construct a comprehensive user study environment.

6.1 WebXR Interaction Logging

Existing web analytic tools and libraries (e.g., Google Analytics [25], Firebase [23]) capture 2D web events (e.g., clicks, keyboard inputs, touch interactions, page navigation) to log users' interactions. However, there is a fundamental difference between events and the information captured within the standard web and WebXR. For instance, WebXR introduces new input systems, such as gaze and controllers, which must be logged individually. In our user study, we analyze the impact of the attack categories defined in our taxonomy on user interaction behavior. Therefore, to capture detailed user interactions, in WebXR apps, we log spatial coordinate information, along with the user's device orientation and position. For this purpose, we leverage A-Frame and Three.js to capture various event details that enable us to determine whether the dark patterns impact user interaction behavior.

6.1.1 Interaction Terminology

Table 3 presents the eight granular interactions we consider in our logging framework. To derive these interactions, we first consider that the categories in the taxonomy (Table 2) employ two types of interactions - click (C) and focus (F). To click on an object, the user must first hover over or focus on it using gaze or controller-emitted rays. A click is initiated by pressing the controller trigger or, with the gaze cursor, maintaining focus on the object for a specified duration.

These interactions can be classified as either human-intended (H) or human-unintended/uncontrolled (U). For example, synthetic clicks generated without the user's knowledge or transparent overlays blocking interactions with virtual content can cause unintended clicks on objects linked to dark patterns. In our user-study framework, users interact with various apps to complete tasks. Objects related to the assigned task are defined as $\mathtt{T_{obj}}$, while those associated with dark patterns are defined as $\mathtt{DP_{obj}}$. Interactions with either can be intentional or unintentional via click ($\mathtt{C}$) or focus ($\mathtt{F}$).

6.1.2 Components of Log-Framework

We design the log-framework with a flexible, independent component structure, ensuring adaptability. Embedding the component's name in any scene object activates it. The log-framework starts from a single entry point, activating four other components.

First, environment scanner continuously monitors WebXR scene for new objects. It links the $\mathtt{DP_{obj}}$ interaction logger with all $\mathtt{DP_{obj}}$ and the $\mathtt{T_{obj}}$ interaction logger with all $\mathtt{T_{obj}}$ to track their interactions separately.

Second, $\mathtt{T_{obj}}$ and $\mathtt{DP_{obj}}$ interaction loggers monitor the user's focus, loss of focus, and click events on objects using gaze-cursor and controller-emitted rays, capturing the object's identity, event time, and input source. The $\mathtt{DP_{obj}}$ interaction logger also tracks position changes of $\mathtt{DP_{obj}}$, while distinguishing between intentional and unintentional interactions. For example, in Visual Overlapping attack, where a non-clickable $\mathtt{T_{obj}}$ visually overlaps a $\mathtt{DP_{obj}}$ in the foreground, if the cursor/ray intersects both objects simultaneously, the interaction is deemed unintentional, and intentional otherwise.

Third, cursor interaction logger individually collects interaction data from gaze-cursor and controller rays, including intersection position, distance from the cursor, object identities, and event time. It also provides an overview of simultaneous interactions with multiple objects using the same cursor/ray.

Lastly, camera information logger estimates the user's position from the main camera's location and captures its facing direction, rotation, Field-of-View (FoV), and time data.

6.2 WebXR Interaction Metrics

We introduce metrics to analyze user's task engagement and unintended interactions with the WebXR environment from logs captured by our log-framework. A line of prior work has proposed different ways of measuring a user's virtual presence in an immersive environment [15, 22, 26]. Presence, being a subjective experience, is most commonly measured using post-immersion questionnaires. However, post-immersion questionnaires cannot measure the presence's time variance [26] and are influenced by users' prior experience [15,22]. In contrast, behavioral measurement provides a non-intrusive, cost-effective, and continuous temporal tracking of user engagement. Thus, we propose interaction metrics to define users' active participation in immersive tasks.

Presence ($\mathtt{P}$). We define this metric as the sustained focus on the primary activity when performing a given task.

$$\mathtt{P} = \frac{\sum \mathtt{focus}_{\mathtt{dur}}^{\mathtt{T_{obj}}}}{\mathtt{total\ immersion\ time}} \tag{1}$$

The user interacts with various $\mathtt{T_{obj}}$ as part of the immersive environment. We define the duration of the focus ($\mathtt{focus}_{\mathtt{dur}}^{\mathtt{T_{obj}}}$) as the time interval between removing the focus from a $\mathtt{T_{obj}}$ and initiating the focus on the same object. The logger records the identifiers of cursor (left or right controller, gaze cursor), intersected $\mathtt{T_{obj}}$, and the corresponding timestamp when a cursor enters or leaves a $\mathtt{T_{obj}}$. This generates time intervals, each defined by a start and end timestamp, representing the beginning and end of focus. However, since multiple $\mathtt{T_{obj}}$ can be intersected by multiple cursors simultaneously, these intervals may overlap. To address this, we sort the intervals by start time and merge any overlapping intervals, yielding discrete intervals representing user interaction with any $\mathtt{T_{obj}}$. We quantify the presence ($\mathtt{P}$) as the fraction of time spent engaging with various $\mathtt{T_{obj}}$ in immersive mode.

Safe Engagement ($\mathtt{E_s}$) We define this metric as the fraction of time that the user interacts exclusively with $\mathtt{T_{obj}}$.

$$\mathtt{E_s} = \frac{\sum \mathtt{focus}_{\mathtt{dur}}^{\mathtt{T_{obj}}\ \mathtt{exclusive}}}{\mathtt{total\ immersion\ time}} \tag{2}$$

Given the nature of the cursor, when a user focuses on one object, it simultaneously focuses on other objects intersecting with it. When there is overlap between $\mathtt{T_{obj}}$ and $\mathtt{DP_{obj}}$, such as a transparent object that hinders interactions with a legitimate $\mathtt{T_{obj}}$, both acquire the focus of the cursor. We consider a time interval as valid if the cursor focuses only on $\mathtt{T_{obj}}$ without any overlap with $\mathtt{DP_{obj}}$. Subsequently, we identify disjoint intervals of exclusive focus on $\mathtt{T_{obj}}$ using a method similar to computing $\mathtt{P}$. Thus, comparing $\mathtt{E_s}$ with $\mathtt{P}$ shows the impact of dark patterns on user presence.

Malicious Attention ($\mathtt{MA}$). This metric measures the success of dark patterns by assessing loss of autonomy based on user actions beyond their control.

$$\mathtt{MA} = \frac{\#[\mathtt{UC}^{\mathtt{DP_{obj}}}]}{\#[\mathtt{UC}^{\mathtt{DP_{obj}}}] + \#[\mathtt{HC}^{\mathtt{T_{obj}}}] + \#[\mathtt{HC}^{\mathtt{DP_{obj}}}]} \tag{3}$$

User clicks on $\mathtt{DP_{obj}}$ drive malicious gains, as seen in *Click Manipulation*, where authentic user clicks are received by ads, and in *Functionality Disruption*, where $\mathtt{DP_{obj}}$ blocks clicks on $\mathtt{T_{obj}}$. Again, users may also intentionally click $\mathtt{DP_{obj}}$ out of genuine interest. The logger records different events on $\mathtt{T_{obj}}$ and $\mathtt{DP_{obj}}$ such as human-intended or unintentional focus and clicks. We calculate the count (#) of the following click events to calculate $\mathtt{MA}$, We consider $\mathtt{HC}^{\mathtt{T_{obj}}}$ since intentional clicks on $\mathtt{T_{obj}}$ can redirect to $\mathtt{DP_{obj}}$, while ignoring $\mathtt{UC}^{\mathtt{T_{obj}}}$ as it does not aid dark pattern success. Thus, $\mathtt{MA} \in [0,1]$ represents the fraction of unintended clicks on $\mathtt{DP_{obj}}$.

Blind Spot Rendering Fraction ($\mathtt{BSR_f}$). This metric estimates the proportion of $\mathtt{DP_{obj}}$ outside the FoV, measuring how often they stay out of sight across interactions.

$$\mathtt{BSR_f} = \frac{\sum_{\mathtt{i}} \#[\mathtt{DP_{obj}\ outside\ FoV}]_{\mathtt{i}}}{\sum_{\mathtt{i}} \#[\mathtt{DP_{obj}}]_{\mathtt{i}}} \tag{4}$$

In *Peripheral Exploitation*, the success of attacks depends primarily on positioning $\mathtt{DP_{obj}}$ in blind spots and not on clicks. Thus, ($\mathtt{MA}$) alone is insufficient to measure the success. During each interaction, the logger records the Field of View (FoV), camera position and direction, and gaze position. Additionally, it tracks any position changes of $\mathtt{DP_{obj}}$ over time. This data determines whether a $\mathtt{DP_{obj}}$ lies within the user's FoV by calculating the angle between the user's gaze direction and the relative direction of the $\mathtt{DP_{obj}}$. Here, $\mathtt{BSR_f} \in [0,1]$, $\#[\mathtt{DP_{obj}\ outside\ FoV}]_{\mathtt{i}}$ represents the total number of $\mathtt{DP_{obj}}$ that are outside the FoV during the $\mathtt{i}^{th}$ interaction, and $\#[\mathtt{DP_{obj}}]_{\mathtt{i}}$ represents the total number of $\mathtt{DP_{obj}}$ during the $\mathtt{i}^{th}$ interaction.

A high $\mathtt{BSR_f}$ value indicates that a large proportion of the $\mathtt{DP_{obj}}$ is located outside the user's FoV during interaction with the interface. However, legitimate objects may sometimes be rendered outside the FoV. For example, while exploring a city in 3D mode, a user might look to their left to view an object, causing objects on the right, like a park or another building, to fall outside their FoV. The logger exclusively tracks the position changes of $\mathtt{DP_{obj}}$ to calculate the $\mathtt{BSR_f}$ metric, and not any $\mathtt{T_{obj}}$. As a result, there are no false positives for $\mathtt{BSR_f}$ in our study. However, a high $\mathtt{BSR_f}$ score across multiple interactions can indicate deliberate design choices intended to mislead. This can serve as a red flag for regulatory bodies and auditors.

6.3 User Study Applications

Our framework examines the influence of app interaction demands on user behaviors within WebXR. We considered ecological validity in the app design. We curated four apps: gaming [60], reading [2], shopping [4], and travel [5] to simulate real-world scenarios with common XR interactions like gaze tracking and controller raycasting. Tasks, such as viewing clothes, shooting targets, reading a book, and exploring locations, were designed to be accessible without specialized skills or excessive effort. Interaction intensity varies by app, with gaming requiring fast actions, reading slower engagement, and shopping or traveling in between.

We implemented the 14 attacks from our taxonomy in each app, creating $(14 \times 4) = 56$ apps that incorporate dark designs and four apps as a control group without any attack for user study. Using A-Frame [1], we developed and customized the apps and hosted them on Glitch [24] to facilitate distribution during user study. Appendix A.2 provides app design details. Each app included the log-framework to capture user interaction data. Developers can integrate the logger into A-Frame apps by specifying all $\mathtt{T_{obj}}$, $\mathtt{DP_{obj}}$, and the target attack.

Despite our constructed apps and metrics, user behavior significantly affects attack success. These apps and metrics reflect this variability. For instance, the Visual Overlapping attack relies on users clicking unclickable objects in front of ads. Without this behavior, the attack fails or results in a lower $\mathtt{MA}$ score. In contrast, attacks in *Peripheral Exploitation* consistently succeed as dark pattern objects stay outside the user's FoV. Our user study framework, including the logger, metrics, and applications, effectively captures variations in user actions and attention in the WebXR environment.

7 WebXR UI Attack User Study

We conducted a between-subjects user study involving five groups (four categories of attacks and a control group) to answer RQ2 (Section 1), which we divide into the following three sub-research questions:

SQ1: How do WebXR UI attacks influence the quality of users' experience?

SQ2: Do these attacks influence user attention and alter interaction behavior, diverting them from their tasks?

SQ3: Do these attacks achieve their intended objectives?

7.1 Study Design

We conducted an in-person between-subjects user study involving 100 participants, each equipped with a Meta Quest 2 headset. We randomly distributed the participants into five groups, each with 20 participants. Each group of 20 participants interacted with only one category (either attacks employing dark patterns or the control group). We then divided the 20 participants into four groups, each with five participants. Each participant in each group interacted with one of the four apps. We also conducted an online survey using Qualtrics [50] to capture the user experience after each experiment. Our study was approved by our institutional IRB. We detail the ethics considerations of our study in Section 12.

Pilot Study. Following an IRB-approved protocol, we conducted a pilot study with five participants recruited from the university. They completed the screening questionnaire prior to attending the one hour in-lab study. We made several modifications to the applications based on observations from the pilot study. For example, we added functionality to the left controller to navigate to the previous page, allowing users to easily read back and forth. In the travel app, we added thumbstick rotation based on participant feedback, making it easier to find and click on target objects for teleportation.

With *Click Manipulation* having the most attacks (five), participants in the pilot study interacted with five scenarios in one hour, completing an experience questionnaire after each. Due to reported fatigue after mainly four scenarios and a preference for three, we limited the main experiments to three scenarios per participant. We randomly selected three attack scenarios for each user from a single attack category, integrated within the assigned app type. For instance, a participant assigned to *Click Manipulation* and the shopping app interacted with three *Click Manipulation* attack scenarios within the shopping app. To balance, *UI-based Privacy Leakage* participants interacted with both scenarios, and one was repeated randomly as third. For the control group, the same scenario was repeated three times per user.

Following our pilot study, we revised the survey to improve clarity, reduce bias, and encourage detailed responses. For instance, we updated the "user-friendliness" question to prompt users to describe helpful or problematic features.

Online Survey. We designed a survey divided into four sections to gather information on user experiences after interacting with each of the three scenarios. The first three sections of the survey consist of Likert scale items and open-text questions that allow users to provide explanations or further information about their experiences after engaging with the selected WebXR app. In the last section of the survey, participants respond to demographic questions and share their experiences and familiarity with AR/VR and computer technology. The complete questionnaire is available in Appendix A.3.

7.2 Recruitment

We recruited participants over 18 years of age from our university through an IRB-instructed procedure. Each participant spent a maximum of one hour completing the set of three experiments along with the survey. Our study was advertised as a perception study of UI patterns in WebXR. We recruited 100 participants for the main user study. The study participants comprised 61% males, 36% females, and 3% who chose not to disclose their gender. 71% of participants identified as Asian, 21% as White, 2% as Black or African American, and 6% as belonging to the "Other" category. 50% of participants were aged 18–24, 44% were aged 25–34, 4% were aged 35–44, and 2% were 45 or older. 65% of participants had technical skills from education in computer science, software development, or related fields. 80% of participants had minimal to extensive AR/VR experience. Appendix A.4 provides more details on AR/VR experience of the participants.

7.3 Data Analysis

Qualitative Analysis. We thematically analyzed survey responses from the participants. Two authors independently performed thematic analysis applying a deductive-inductive coding approach to extract themes pertaining to user experience. As detailed in A.3, we asked each user five open-text questions for each experiment, gaining $5 \times 3 = 15$ responses per user. These survey responses allow us to measure UX discomfort as an indicator of the negative impact of dark patterns on WebXR UI. This discomfort highlights the potential loss of autonomy that users experience, even when unaware of the manipulation. For example, the GCJ attack can create a sense of cursor misalignment, the GUI Switch attack can cause delayed rendering issues, and the DoS attack can prevent users from performing intended actions. Additionally, some attacks in our taxonomy involve overlapping objects in the same space, which can cause flickering or rendering inconsistencies [16], which can also lead to user discomfort. We quantify UX discomfort using the metric `#[UD]`, reflecting the challenges faced by users.

Two authors independently performed deductive coding for (15×100) open-text responses. If the user mentions any discomfort related to the UI, we mark them as 1; otherwise, we mark them as 0. Then, we compute the total number of 1s to compute `#[UD]` per user as a proxy for the evaluation. For all texts labeled 1, two authors conducted an inductive analysis of responses to develop a codebook of themes related to different UX discomforts. Once we generated the themes, we reviewed the responses again and applied our codebook, assigning each response labeled 1 to a specific theme. Authors achieved a high level of inter-rater agreement (Cohen's Kappa [31], $\kappa = 0.81$), before resolving disagreements.

Quantitative Analysis. We leveraged statistical significance tests to measure differences between quantitative data, and associations between categorical data (e.g., extracted themes association with attack category). For quantitative data, we derived the metrics data for each user by averaging the results from the three experiments. The Shapiro–Wilk test [53] and Bartlett's test [9] indicated non-normality and non-homogeneity of the data, respectively. Thus, we proceeded with non-parametric tests to test for significant differences.

With two independent variables, "Group" (attack category and control) and "App", we used the Scheirer–Ray–Hare (`SRH`) [52], a non-parametric alternative to 2–way ANOVA [45], to test significance across groups and apps. When significance was detected, we performed further analysis using Dunn's post-hoc test. We also employed the Kruskal–Wallis (`KW`) [37] (a non-parametric version of the 1-way ANOVA), to test for difference within one attack/app category. To test the association between groups and themes derived from inductive analysis, we employed the Chi-Square (χ^2) test [59] and applied an adjusted residual analysis to determine the contributing factors to the χ^2 test.

For significance tests, we assume a null hypothesis (H_0) of no significant differences/association, and an alternative hypothesis (H_1) that suggests there is a significant difference/association. We reject (H_0), and accept (H_1) when the p-value is less than a threshold value (α). For each conducted test,

Table 4: Participants' self-reported difficulty and user friendliness scores (Mean ± Stdev).

Likert Scale	*Click Manipulation*	*Peripheral Exploitation*	*Functionality Disruption*	*UI-based Privacy Leakage*	*Control Group*
Difficulty	1.40(±0.6)	1.30(±0.73)	3.55(±1.14)	1.55(±0.76)	1.25(±0.55)
User Friendliness	4.30(±0.73)	4.55(±0.69)	2.95(±1.19)	3.80(±0.95)	4.75(±0.44)

we use the G*Power [21] Analysis to derive a threshold (α) that is required to achieve 80% power and a moderate effect size of $\eta^2 = 0.11$ (under our sample size and groups). For statistically significant results, we note the p-value and effect size ($\eta^2 > 0.14$, large effect size for significant results).

7.4 Results and Findings

7.4.1 SQ1: UI Attack Influence on Users' Experience

We analyzed the 15 open-text responses per user to understand UI attack influence on user experience.

Distribution of UX Discomforts. `SRH` test on `#[UD]` indicates that "Group" has a statistically significant impact (**p-value** $< \mathbf{0.001}$ $\eta^{\mathbf{2}} = \mathbf{0.34}$) on `#[UD]`. In contrast, "App" does not exhibit a significant main effect, suggesting that it does not independently influence `#[UD]`. Furthermore, we find no significant interaction effect between "Group" and "App", implying that the effect of one independent variable on `#[UD]` does not depend on the other's level. These results allow us to reject the null hypothesis that UX discomfort levels are consistent across all groups. The results of Dunn's post hoc test indicate that *Functionality Disruption* is significantly different from the other attack categories and the control group, with no significant differences observed among the other attack categories or between them and the control group.

Table 4 shows the mean and standard deviation of the users' Likert scale scores for difficulty and user-friendliness. We calculate the overall experience score (`#[Exp]`) as the ratio of difficulty level to user-friendliness, reflecting the inverse relationship where higher difficulty typically results in lower user-friendliness and vice versa. `SRH` test on `#[Exp]` yields similar results to those obtained using `#[UD]`, indicating a significant difference (**p-value** $< \mathbf{0.001}$, $\eta^{\mathbf{2}} = \mathbf{0.4}$) across groups, thus confirming the comparability of the deductive analysis with the Likert scale score.

Our results show that *Functionality Disruption* significantly impacts user experience, while `#[UD]` and `#[Exp]` remain comparable between other attack categories and the control group. This likely stems from *Functionality Disruption*'s attack methods, where the adversary blocks interactions with XR content (e.g., overlaying transparent objects), causing user frustration.

Types of Discomforts. Referring to `#[UD]` with label 1, our inductive analysis of the free responses generated a codebook containing 5 distinct themes related to the various UX discomforts experienced by users. In general, users encountered difficulties with visual feedback and interaction clarity (11.95%), responsiveness and interaction lag (17.64%), controller and cursor alignment issue (18.97%), pop-up or ad-related distractions (18%), and miscellaneous minor issues (33.3%), as shown in Figure 3. We found a significant association between the UX difficulty themes and the five groups [$\chi^{\mathbf{2}}(\mathbf{dof} = \mathbf{16}, \mathbf{N} = \mathbf{100}) = \mathbf{156.925}$, **p-value** $< \mathbf{0.001}$].

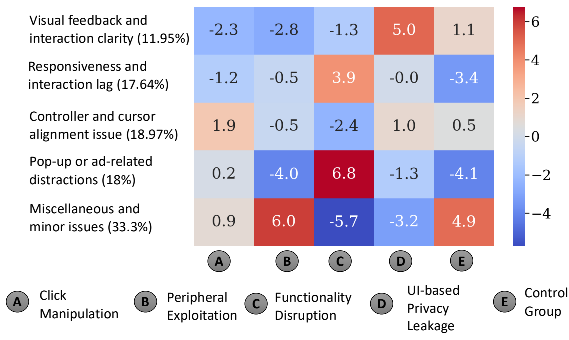


Figure 3: Five types of UX discomforts experienced by participants and their adjusted residual frequency summary (>2 indicates over-representation and <2 indicates under-representation). % on the left indicates the percentage of total participants that experienced that type of discomfort.

Adjusted residual analysis reveals significant deviations in UX discomfort themes in different categories of attacks. Noticeably, users did not report any significant issues in *Peripheral Exploitation* or the control group. Here, we note that in *Peripheral Exploitation* most of the malicious activities occur in the user's blind spot. For *Functionality Disruption*, users frequently report issues with (1) pop-up or ad-related distractions, and (2) responsiveness and interaction lag – intentional obstructions hinder their interactions and often lead them to click on objects multiple times. For *UI-based Privacy Leakage*, users had increased concerns about visual feedback and interaction clarity (e.g., a GUI switch attack may cause cursors to freeze, leading to unrecognized clicks and prompting users to report the issue).

`Finding-1:` Users exposed to *Functionality Disruption* face significant interaction challenges, frequently noting pop-up or ad-related distractions. In contrast, user experiences in other attack categories are similar to each other and align closely with the experiences in the absence of attacks, suggesting participants fail to recognize the loss of autonomy caused by adversarial UI manipulation.

7.4.2 SQ2: UI Attack Influence on User Attention and Interaction Behavior

We use the presence (`P`) and safe engagement ($\mathtt{E_s}$) metrics defined in Section 6.2 to understand the influence of dark patterns on users' interaction behavior with the task.

Analysis of Presence (`P`). `SRH` test suggest that neither the

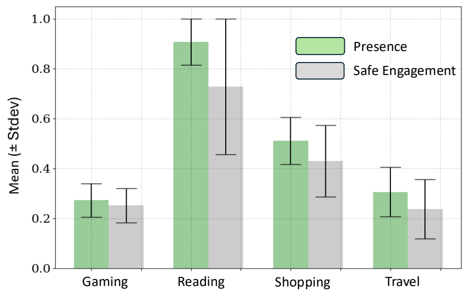


Figure 4: Presence (P) and Safe Engagement (E_s) across apps.

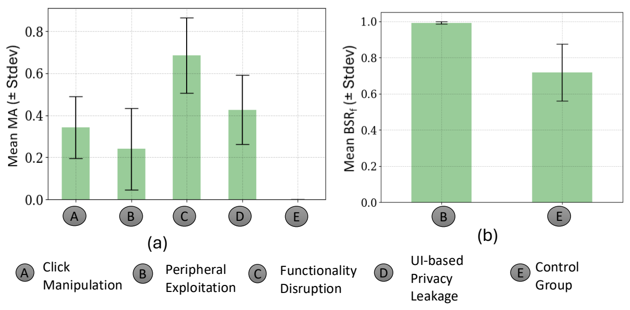


Figure 5: Success across groups based on (a) Malicious Attention (MA) and (b) Blind Spot Rendering Fraction (BSR_f).

factor "Group" nor the interaction between "App" and "Group" significantly influences P. However, "App" shows a significant impact on presence ($\textbf{p-value} < \mathbf{0.001},\ \eta^2 = \mathbf{0.78}$). Dunn's post-hoc test reveals a significant difference across all pairs of app types, except the gaming and travel pair. The mean ranks indicate that the presence level is significantly higher in reading apps, where users spend more time reading and less time interacting. Shopping apps exhibit moderate presence levels, higher than gaming and travel, as users deliberate over products before purchasing. In contrast, gaming and travel apps show similar lower presence levels, as users in both contexts engage frequently with the environment to explore more places or achieve high scores. Hence, we can conclude that P is higher in apps where users maintain steady attention.

Analysis of Safe Engagement (E_s). The SRH test reveals a significant effect of "Group" on E_s ($\textbf{p-value} < \mathbf{0.001},\ \eta^2 = \mathbf{0.27}$), contrasting P where "Group" has no significant effect. This implies that even if user's focus on the task remains comparable across different groups (P), dark pattern integration forces user to shift their engagement with the given task (E_s). The results also show that "App" significantly impacts E_s ($\textbf{p-value} < \mathbf{0.001},\ \eta^2 = \mathbf{0.4}$), similar to P. The results show no significant interaction between "App" and "Group" on E_s.

A Dunn's post-hoc test indicates that E_s for *Functionality Disruption* is significantly lower from *Peripheral Exploitation*, *Click Manipulation*, and the control group. As seen in Figure 4, reading and shopping apps have significantly greater E_s than gaming and travel apps. However, E_s is significantly more impacted by UI attacks in reading and shopping apps compared to gaming and travel. We present additional post-hoc results in Appendix A.5.

Finding-2: User's presence is solely influenced by the app type and not attack categories, further suggesting they do not recognize loss of autonomy. However, attacks employing dark patterns do have a significant effect on the safe engagement, with the *Functionality Disruption* group having significant differences from the group not exposed to dark patterns and most other attack categories.

7.4.3 SQ3: Attack Success

We use the Malicious Attention (MA) and the Blind Spot Rendering Fraction (BSR_f) metrics, defined in Section 6.2 to assess if dark patterns achieve their intended objectives.

Analysis of Malicious Attention (MA). SRH test shows that "Group" has a statistically significant impact on MA ($\textbf{p-value} < \mathbf{0.001},\ \eta^2 = \mathbf{0.71}$)[3]. "App" and the interaction of "App" and "Group" do not have a statistically significant impact on MA. Dunn's post-hoc test shows that the control group is significantly different from all attack categories. We find that *Functionality Disruption* differs significantly from other attack categories (Figure 5(a)). We also find that *Functionality Disruption* receives the most unintended clicks, supported by our previous findings for this group about significant interaction lag that causes users to click multiple times on the same object. The result also suggests that user interaction with different types of apps will be equally vulnerable to dark patterns. We present additional post-hoc results in Appendix A.5.

Analysis of Blind Spot Rendering Fraction (BSR_f). According to our previous findings, users in both the control group and *Peripheral Exploitation* did not report ad-related distractions. In our control group, a genuine ad was placed in a position where users could choose to view and interact with it if desired (e.g., ad was placed far left while all T_{obj} were in front of the user). If the user decides to focus on the assigned task, the ad may be placed outside the user's FoV, causing BSR_f to become high, as seen in (Figure 5(b)). SRH test shows that *Peripheral Exploitation* is significantly different from control group based on BSR_f ($\textbf{p-value} < \mathbf{0.001},\ \eta^2 = \mathbf{0.77}$) (Figure 5(b)). "App" and interaction of "App" and "Group" have no significant impact on BSR_f. Examining *Peripheral Exploitation* alone using KW test, we observe that there is no significant difference among the apps.

[3]For *Peripheral Exploitation*, we only include attack samples where unintended clicks occur.

`Finding-3:` In our study, all attacks achieved their malicious goals (Figure 5), leading to loss of autonomy for the participants interacting with the interface. Despite the varying level of presence across different application types (Figure 4), we observe that malicious events consistently occur, suggesting that these attacks generalize to different application types.

8 Discussion & Limitations

WebXR UI Attack Detection. Our work shows that malicious entities exploit sensitive UI properties in WebXR to manipulate users. To counter this, we introduce a logging framework for WebXR developers to detect user interactions with dark patterns. Our framework can inform WebXR app stakeholders (e.g., app developers for a gaming app) if users engage with promotional or third-party content more than necessary (as demonstrated in our user study). Here, we note that the metrics we developed can also help detect dark pattern success in immersive platforms beyond WebXR. However, using our metrics beyond WebXR requires pre-identifying potential dark pattern interactions. Future work will explore extending our framework and metrics to immersive platforms such as standalone AR/VR headsets.

Developer Guidelines for Defense. It is crucial for WebXR stakeholders to prevent dark pattern manipulation and safeguard user autonomy. To achieve this, we propose developer guidelines in three key areas: input validation, run-time monitoring, and design considerations.

Developers should validate input sources (e.g., cursorId from event.detail) to ensure that interactions originate from the intended cursor, preventing attacks like GCJ and CCJ. However, attackers may still extract the authentic cursorId and use the cursor object with synthetic events on targets. Therefore, verifying other associated attributes, such as the intersection attribute from event.detail, containing the distance and the point of intersection, is crucial. By default, a synthetic event lacks these associated attributes. Accurately replicating all attributes with the same timestamp presents a significant challenge to attackers. Additionally, user-generated events have the isTrusted [34] property set to true, enabling the identification and rejection of unrecognized synthetic events on target objects. Developers should also block invalid auxiliary screen redirection, such as in the Malvertising attack, by overriding the window.open event.

As part of run-time monitoring, developers should track enter and exit to 3D mode events to identify and restrict unauthorized activities to prevent attacks such as AAD, Malvertising, and GUI Switch. Developers should monitor advertisement UI entities, focusing on position, size, and opacity changes to prevent attacks that involve them.

Design consideration is another important aspect of preventing attacks such as Sequential Rendering, Visual Overlapping, Object Erasure. Developers should avoid rendering clickable, third-party objects behind unclickable objects. If such layering is unavoidable, they should provide clear warnings near the unclickable objects. To prevent information erasure, developers should avoid loading clickable entities after rendering transparent foreground elements.

App developers can audit their apps using our UI properties list to mitigate dark design risks. Future work will focus on automated tools to detect dark patterns in XR UI designs.

UI Attacks in Multi-User Settings. Our analysis focuses on attacks in single-user settings, as most WebXR apps cater to individual users [7,46,54]. However, multi-user WebXR apps, such as group fitness [58] and social gaming [14], are increasingly being developed. In these multi-user settings, dark patterns can be particularly insidious. For instance, malicious entities may exploit social manipulation for peer pressure or social cues to coerce users into actions they might not otherwise take. Similarly, bait-and-switch tactics may lure users with appealing features that change once they are engaged. Privacy invasion may occur when personal data is collected and shared without consent, exploiting user trust. Future work will investigate how these properties may enable new dark designs in multi-user environments.

UI Attacks using Hand-Tracking. The attack scenarios in our taxonomy rely on the use of WebXR's primary interaction modes: gaze and controller cursors. However, WebXR also supports hand-tracking [3], enabling actions like gripping and pointing. Hand tracking can be exploited for dark patterns, such as making "like" or "agree" gestures overly sensitive, causing users to unintentionally approve actions. This is particularly concerning in social or commercial contexts, where slight movements might cause unintended purchases or agreements, undermining user autonomy. Future research will examine the role of hand-tracking in dark designs.

9 Related Work

User Interface Manipulation. User Interface (UI) attacks pose a significant threat, exploiting vulnerabilities in design and implementation across web [32, 55] and mobile [6, 12, 20, 40, 44] platforms. For instance, multiple clickjacking and UI redressing attacks [32, 49] have been proposed that aim to manipulate users into performing unintended actions. In smartphones, smaller touchscreens increase susceptibility to tapjacking and overlay attacks [10, 18, 65]. To counter these threats, defenses such as frame-busting techniques have been proposed to prevent unauthorized framing of web content [63]. Recent works [51,65] have proposed defense tools to improve mobile protection. However, they focus solely on 2D UI, ignoring UI attacks in immersive environments.

Researchers [16, 39] have recently explored UI attacks in XR by exploiting the lack of iframe-like elements. However, they only focus on the feasibility of such attacks, without thoroughly analyzing the vulnerable UI characteristics or examining their impact on user perception. In contrast, we provide a comprehensive analysis of vulnerable UI properties in WebXR, proposing new UI attacks, and conducting an in-depth study to understand their impact on users.

Dark Patterns in Different Platforms. The research community has long paid attention to dark patterns. For example, previous efforts have focused on dark patterns in domains such as IoT (from vendors) [35], mobile permission prompts [43], home robots [38]. Several dark pattern taxonomies also exist. Bosch et al. introduced eight privacy dark patterns [11]. Gray et al. [27] proposed five categories of dark design strategies building upon the taxonomy of Brignull et al. [13]. Mathur et al. [41, 42] introduced higher-level attributes to organize these taxonomies systematically.

In XR, one study [61] examines dark patterns in AR, emphasizing risks to user autonomy and safety, while another [36] uses co-design workshops to identify and characterize dark patterns unique to XR. In contrast, we systematically analyze vulnerable UI properties in WebXR, introduce new dark pattern-based attacks, create a taxonomy of existing and new attacks, and evaluate their impacts on the user experience.

10 Conclusions

We introduce a taxonomy of four attack categories comprising 14 UI attacks (including five new attacks), after uncovering 14 UI properties that contribute to dark patterns in the WebXR ad ecosytem. To assess the impact of UI attacks on users, we developed a user study framework comprising a logging framework and a suite of interaction metrics. Leveraging this framework, we conducted a between-subjects study with 100 participants using this framework. We find that UI attacks are successful in reaching their malicious objectives while users often do not recognize their loss of autonomy.

11 Acknowledgments

We appreciate the valuable feedback and suggestions from our shepherd and anonymous reviewers. This research was partially funded by the National Science Foundation (NSF) through grants CNS-2144645 and IIS-2229876. The findings, conclusions, and recommendations presented in this paper are solely those of the authors and do not necessarily represent the views of the NSF.

12 Ethics Considerations

Our research identifies security-sensitive UI properties that can contribute to dark patterns and can be leveraged by any entities within the WebXR ad-ecosystem (developers, advertisers, ad service providers) to manipulate users. We also propose novel attacks exploiting these UI properties. Recognizing the potential for harm, we carefully considered the ethical implications of our work and implemented mitigation strategies throughout the research process.

Stakeholders. Our study identifies four key stakeholders at risk: end-users, developers, advertisers, and ad service providers. Users risk losing autonomy, facing security and privacy threats (e.g., privacy leakage, malware download). Developers risk reputational damage and losing user trust. Advertisers face financial losses and missing genuine engagement with their ads. Ad service providers risk losing client advertisers and developers due to reputational damage.

Mitigating Potential Harm. Despite potential drawbacks, our analysis of exploitable UI properties in WebXR is a starting point for developing future tools to automatically secure apps against potential UI attacks that leverage the ad-ecosystem. To mitigate potential harm, we provide concrete developer guidelines in the paper. This includes verifying input sources, blocking unknown redirections to the auxiliary screen, carefully placing objects with consideration for clickability and transparency, monitoring for suspicious use of enter and exit events to or from 3D mode and suspicious changes in position, size or opacity of ads. Additionally, we recommend that ad service providers implement robust verification processes for advertisements and regularly monitor developer sites to prevent misuse. To further mitigate the negative consequences of publishing the new attacks, we disclosed both the attacks and potential mitigation strategies to Meta, WebXR, and A-Frame.

Screening and Eligibility. To assess the impact of UI attacks on user perception in WebXR, we did not disclose the true purpose of the study initially, however, after the experiment, we held debrief sessions. This study was approved by our institutional IRB. We followed the IRB approved method to recruit students from our university for our in-lab study. To ensure participant well-being during the XR interaction, we asked interested individuals to complete an online screening questionnaire addressing potential discomforts by inquiring about history of motion or car sickness; the presence of conditions such as epilepsy, migraines, unexplained seizures, recent concussions, or light sensitivity; neurological or vestibular issues; uncorrected vision impairments; issues with physical mobility; a history of experiencing physical side effects when using computer or gaming controllers; the ability to move their head and upper body; and any issues affecting hand or finger movement. Individuals were considered eligible for the pilot or main study if they reported no issues in any of the aforementioned categories.

Security and Safety during User Study. Eligible participants were invited to take the in-lab experiment. The experiment's maximum duration was one hour, which was communicated to the participants beforehand. Each participant received $20 in compensation, exceeding the minimum hourly wage. Before starting the experiment, we explained the study procedure and gave them a demo of using the Meta Quest 2 device. Participants signed consent forms after being fully informed of the study procedures, and potential risks and benefits. The participants initially interacted with a demo WebXR app where they were able to practice all the interactions necessary for the primary experiments. All collected data, including surveys and interaction logs, were anonymized to protect participant privacy. The tasks themselves were designed to be simple and avoid any undue physical or cognitive strain. Importantly, the attacks were conducted within our implemented environment - they did not involve any PII, e.g., credentials, that could cause direct harm to users. Additionally, the researchers monitored the participants' well-being and informed them of their right to withdraw at any time without impacting their compensation. Thus, our ethical decisions prioritized user well-being while aiming to benefit the WebXR ad ecosystem by identifying vulnerabilities and improving security.

13 Open Science

We have provided the artifacts of our study online [47]. We have made the suite of apps, the logging framework, and the interaction metrics library publicly available.

References

[1] "A-frame - web framework for building virtual reality experiences," https://aframe.io/, [Online; accessed 28-Jan-2025].

[2] "Comic book - a-frame example," https://aframe.io/aframe/examples/showcase/comicbook/, [Online; accessed 28-Jan-2025].

[3] "Hand-tracking controls - a-frame 1.6.0 documentation," https://aframe.io/docs/1.6.0/components/hand-tracking-controls.html, [Online; accessed 28-Jan-2025].

[4] "Shopping - a-frame example," https://aframe.io/aframe/examples/showcase/shopping/, [Online; accessed 28-Jan-2025].

[5] "A-frame 360 vr tour example," https://glitch.com/~a-frame-360vr-tour, [Online; accessed 28-Jan-2025].

[6] S. Aonzo, A. Merlo, G. Tavella, and Y. Fratantonio, "Phishing attacks on modern android," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[7] "A-painter - paint in vr on the web," https://aframe.io/a-painter/, [Online; accessed 28-Jan-2025].

[8] "Babylon.js - javascript 3d engine," https://www.babylonjs.com/, [Online; accessed 28-Jan-2025].

[9] M. S. Bartlett, "Properties of sufficiency and statistical tests," *Royal Society of London. Series A-Mathematical and Physical Sciences*, 1937.

[10] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna, "What the app is that? deception and countermeasures in the android user interface," in *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[11] C. Bösch, B. Erb, F. Kargl, H. Kopp, and S. Pfattheicher, "Tales from the dark side: Privacy dark strategies and privacy dark patterns," *Privacy Enhancing Technologies (PETS)*, 2016.

[12] D. Bove, "Sok: The evolution of trusted ui on mobile," in *ACM on Asia Conference on Computer and Communications Security (CCS)*, 2022.

[13] H. Brignull, M. Leiser, C. Santos, and K. Doshi. Deceptive patterns – user interfaces designed to trick you. [Online]. Available: https://www.deceptive.design/

[14] "Castle - a web-based 3d collaboration tool," https://castle.needle.tools/, [Online; accessed 28-Jan-2025].

[15] Y. Chandio, N. Bashir, V. Interrante, and F. M. Anwar, "Investigating the correlation between presence and reaction time in mixed reality," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[16] K. Cheng, A. Bhattacharya, M. Lin, J. Lee, A. Kumar, J. F. Tian, T. Kohno, and F. Roesner, "When the User Is Inside the User Interface: An Empirical Study of UI Security Properties in Augmented Reality," in *USENIX Security*, 2024.

[17] K. Cheng, J. F. Tian, T. Kohno, and F. Roesner, "Exploring user reactions and mental models towards perceptual manipulation attacks in mixed reality," in *USENIX Security*, 2023.

[18] V. Cooper, "Tapjacking threats and mitigation techniques for android applications," *DigitalCommons@Kennesaw State University*, 2014.

[19] "Csrankings: Computer science rankings," https://csrankings.org/, [Online; accessed 28-Jan-2025].

[20] H. Farrukh, T. Yang, H. Xu, Y. Yin, H. Wang, and Z. B. Celik, "S3: Side-channel attack on stylus pencil through sensors," *ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2021.

[21] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "G* power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior research methods*, 2007.

[22] W. M. Felton and R. E. Jackson, "Presence: A review," *International Journal of Human–Computer Interaction*, 2022.

[23] "Firebase documentation: Logging events," https://firebase.google.com/docs/analytics/, [Online; accessed 28-Jan-2025].

[24] "Glitch platform," https://glitch.com/, [Online; accessed 28-Jan-2025].

[25] "Google analytics documentation," https://support.google.com/analytics/, [Online; accessed 28-Jan-2025].

[26] S. Grassini and K. Laumann, "Questionnaire measures and physiological correlates of presence: A systematic review," *Frontiers in Psychology*, 2020.

[27] C. M. Gray, Y. Kou, B. Battles, J. Hoggatt, and A. L. Toombs, "The dark (patterns) side of ux design," in *CHI conference on human factors in computing systems*, 2018.

[28] S. Greenberg, S. Boring, J. Vermeulen, and J. Dostal, "Dark patterns in proxemic interactions: a critical perspective," in *Conference on Designing interactive systems*, 2014.

[29] J. Gunawan, C. Santos, and I. Kamara, "Redress for dark patterns privacy harms? a case study on consent interactions," in *Symposium on computer science and law (CS&Law)*, 2022.

[30] H. Hadan, L. Choong, L. Zhang-Kennedy, and L. E. Nacke, "Deceived by immersion: A systematic analysis of deceptive design in extended reality," *ACM Computing Surveys*, 2024.

[31] L. M. Hsu and R. Field, "Interrater agreement measures: Comments on kappan, cohen's kappa, scott's π, and aickin's α," *Understanding Statistics*, 2003.

[32] L.-S. Huang, A. Moshchuk, H. J. Wang, S. Schechter, and C. Jackson, "Clickjacking: Attacks and defenses," in *USENIX Security*, 2012.

[33] "Html iframe element - mdn web docs," https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe, [Online; accessed 28-Jan-2025].

[34] "Event:istrusted," https://developer.mozilla.org/en-US/docs/Web/API/Event/isTrusted#specifications, [Online; accessed 28-Jan-2025].

[35] M. Kowalczyk, J. T. Gunawan, D. Choffnes, D. J. Dubois, W. Hartzog, and C. Wilson, "Understanding dark patterns in home iot devices," in *CHI Conference on Human Factors in Computing Systems*, 2023.

[36] V. Krauss, P. Saeghe, A. Boden, M. Khamis, M. McGill, J. Gugenheimer, and M. Nebeling, "What makes xr dark? examining emerging dark patterns in augmented and virtual reality through expert co-design," *ACM Transactions on Computer-Human Interaction*, 2024.

[37] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, 1952.

[38] C. Lacey and C. Caudwell, "Cuteness as a 'dark pattern'in home robots," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019.

[39] H. Lee, J. Lee, D. Kim, S. Jana, I. Shin, and S. Son, "AdCube: WebVR Ad Fraud and Practical Confinement of Third-Party Ads," in *USENIX Security*, 2021.

[40] M. Luo, O. Starov, N. Honarmand, and N. Nikiforakis, "Hindsight: Understanding the evolution of UI vulnerabilities in Mobile browsers," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[41] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan, "Dark patterns at scale: Findings from a crawl of 11K shopping websites," *ACM on human-computer interaction*, 2019.

[42] A. Mathur, M. Kshirsagar, and J. Mayer, "What makes a dark pattern... dark? design attributes, normative considerations, and measurement methods," in *CHI conference on human factors in computing systems*, 2021.

[43] R. Mohamed, A. Arunasalam, H. Farrukh, J. Tong, A. Bianchi, and Z. B. Celik, "ATTention Please! An Investigation of the App Tracking Transparency Permission," in *USENIX Security*, 2024.

[44] R. Mohamed, H. Farrukh, Y. Lu, H. Wang, and Z. B. Celik, "istelan: Disclosing sensitive user information by mobile magnetometer from finger touches," *Privacy Enhancing Technologies (PETS)*, 2023.

[45] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2012.

[46] "Moonrider game," https://moonrider.xyz/, [Online; accessed 28-Jan-2025].

[47] C. Mukherjee, R. M. Aburas, A. Arunasalam, H. Farrukh, and Z. B. Celik, "Shadowed Realities: An Investigation of UI Attacks in WebXR - Research Artifacts," https://doi.org/10.6084/m9.figshare.28271207, 2025, [Online; accessed 28-Jan-2025].

[48] "Above paradowski game," https://aboveparadowski.com/, [Online; accessed 28-Jan-2025].

[49] A. Possemato, A. Lanzi, S. P. H. Chung, W. Lee, and Y. Fratantonio, "Clickshield: Are you hiding something? Towards Eradicating Clickjacking on Android," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[50] "Qualtrics," https://www.qualtrics.com/, [Online; accessed 28-Jan-2025].

[51] C. Ren, P. Liu, and S. Zhu, "Windowguard: Systematic protection of gui security in android." in *NDSS*, 2017.

[52] C. J. Scheirer, W. S. Ray, and N. Hare, "The analysis of ranked data derived from completely randomized factorial designs," *Biometrics*, 1976.

[53] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, 1965.

[54] "Soundboxing game," https://webvr.soundboxing.co/, [Online; accessed 28-Jan-2025].

[55] C.-A. Staicu and M. Pradel, "Leaky images: Targeted privacy attacks in the web," in *USENIX Security*, 2019.

[56] D. Susser, B. Roessler, and H. Nissenbaum, "Technology, autonomy, and manipulation," *Internet policy review*, 2019.

[57] "Three.js - javascript 3d library," https://threejs.org/, [Online; accessed 28-Jan-2025].

[58] "Virtual reality fitness," https://towermax.fitness/, [Online; accessed 28-Jan-2025].

[59] R. Van Auken and S. Kebschull, "Type i error convergence of three hypothesis tests for small rxc contingency tables," *RMS: Research in Mathematics & Statistics*, 2021.

[60] "Wackarmadiddle game," https://heyvr.io/arcade/games/wackarmadiddle, [Online; accessed 28-Jan-2025].

[61] X. Wang, L.-H. Lee, C. Bermejo Fernandez, and P. Hui, "The dark side of augmented reality: Exploring manipulative designs in AR," *International Journal of Human–Computer Interaction*, 2024.

[62] "Webxr device api," https://www.w3.org/TR/webxr/, [Online; accessed 28-Jan-2025].

[63] M. Zalewski, "Browser security handbook, part 2: Content security policy," https://code.google.com/archive/p/browsersec/wikis/Part2.wiki, 2011, [Online; accessed 28-Jan-2025].

[64] "Zesty: monetize spatial/webxr apps," https://docs.zesty.xyz/about/intro, [Online; accessed 28-Jan-2025].

[65] H. Zhou, S. Wu, C. Qian, X. Luo, H. Cai, and C. Zhang, "Beyond the surface: Uncovering the unprotected components of android against overlay attack," in *NDSS*, 2024.

A Appendix

A.1 Interaction Events

WebXR supports gaze and controller based interactions. Raycaster component offers line-based intersections, while the cursor component enables hover and click states during raycasting events. We detail different event types, supported by these components in Table 5.

Table 5: Events supported in WebXR.

Event Name	Description	Emitted on
click	Clicking on an object (using gaze or controller).	Emitted on both the cursor and intersected object.
fusing	Starting to focus on an object (typically using gaze cursor).	Emitted on both the cursor and intersected object.
raycaster-intersection	Starting to focus on an object (using gaze or controller).	Emitted on the raycaster.
raycaster-intersected	Starting to focus on an object (using gaze or controller).	Emitted on the intersected object.
raycaster-intersection-cleared	Raycaster is no longer intersecting with one or more objects.	Emitted on the raycaster.
raycaster-intersected-cleared	Object is no longer in intersection with the raycaster.	Emitted on the intersected object.

A.2 Application Design

We used a WebXR shooting game [60] in which users can shoot or click on target objects to score points. Users can restart the game after each session or quit the application. We used a WebXR demo shopping environment [4] where users can view some men's and women's clothing, shoes, and wallets. Hovering/focusing on the products shows a zoom-in view of the corresponding product. Then, according to their choice, users can add one or more products to the cart and pay for them. Users can purchase multiple products multiple times in a single session. We used a WebXR demo travel application [5] where users can view and switch between 8 different places. In each place, users can view one or more targets, clicking on which switches the surroundings according to the number. Users can use the controller thumbsticks to move and rotate in each place and find the next switching target. We used a WebXR demo reading application [2] where users can read a comic book. By clicking on the book, the page flips. The right controller and the gaze cursor can flip the page forward, whereas the left controller flips the page backward. Users can zoom in or out of the book using the controller buttons. They can also change the book's position in 3D using the controller thumbsticks.

Table 6: User counts based on XR experience across groups.

Group	Beginner	Intermediate	Advanced
Click Manipulation	8	8	4
Peripheral Exploitation	14	4	2
Functionality Disruption	11	7	2
UI-based Privacy Leakage	9	5	6
Control Group	13	4	3

A.3 User Study Survey

After completing each of the three experiments, the participants were asked to complete a Qualtrics [50] questionnaire about their experience with the WebXR interface. The same set of survey questions was administered after each subsequent experiment. We avoided requesting specific opinions about advertisements to avoid bias. The survey questions were designed to be broad and focused on the user's interaction experience with the interface. We intentionally asked participants to complete the survey post-each experiment to ensure they could better recall their experience details. The following is a comprehensive list of survey questions.

- How would you describe the difficulty of the task? [open-text answer]
- Please rate the difficulty of the task on a scale of 1-5 (1-very easy, 5-very difficult). [likert scale]
- What was your experience with the user-friendliness of the task? Please describe any features or elements that were helpful or problematic. [open-text answer]
- Please rate the user-friendliness of the task on a scale of 1-5 (1-very unfriendly, 5-very friendly). [likert scale]
- How was your experience with using the gaze cursor for this task? [open-text answer]
- How was your experience with using the controllers for this task? [open-text answer]
- Would you like to share anything about your experience interacting with the application? [open-text answer]

A.4 Experience Level with AR/VR

The survey asks each user about their level of experience with AR/VR headsets. The users mark themselves as any of the three levels, 1 for beginner, 2 for intermediate, and 3 for advanced. Figure 6 illustrates the distribution of participants by age and AR/VR experience level. Tables 6, 7 show the distribution of AR/VR experience across different groups and apps. For each group and app type, most users are beginners, followed by intermediate, with the fewest being advanced. We found no relationship between user XR experience and UX discomfort or engagement.

Table 7: User counts based on XR experience across apps.

App	Beginner	Intermediate	Advanced
Reading	14	7	4
Gaming	14	7	4
Shopping	13	7	5
Travel	14	7	4

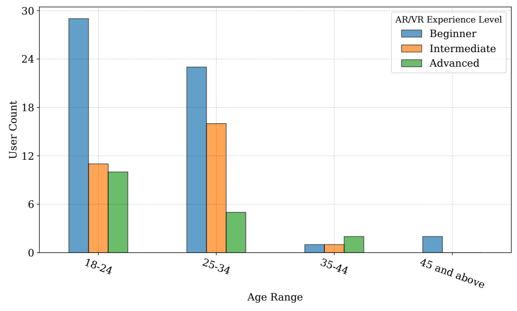


Figure 6: Distribution of participants based on age and AR/VR experience level.

A.5 Additional User Study Post-Hoc Tests

KW Analysis for SQ2. The KW test reveals a significant effect of "App" on $\mathtt{E_s}$ within each attack category (**p-value**$< \mathbf{0.001}$, $\eta^2 > \mathbf{0.47}$). Dunn's post-hoc test further shows that different apps are affected to varying degrees within each attack category. For instance, in *Click Manipulation*, there is a significant difference across all pairs of apps, with reading and shopping having higher $\mathtt{E_s}$ compared to travel and gaming. Similarly, in *Functionality Disruption*, all apps are significantly impacted by dark patterns, resulting in extremely low task involvement across all apps. However, the travel app, which has the lowest $\mathtt{E_s}$, shows a significant difference compared to the other apps.

The KW test reveals a significant effect of "Group" on $\mathtt{E_s}$ within each app type (**p-value** $< \mathbf{0.001}$, $\eta^2 > \mathbf{0.67}$). For example, Dunn's post-hoc test reveals that in the reading app, there is a significant difference between *Peripheral Exploitation* and the other attack categories, but not with the control group. Again, *Functionality Disruption* differs significantly from other groups. *Click Manipulation* is not significantly different from *UI-based Privacy Leakage* but is from the others.

KW Analysis for SQ3. KW test on *Functionality Disruption* data shows a significant impact of "App" on MA (**p-value** $= \mathbf{0.02}, \eta^2 = \mathbf{0.45}$). Dunn's post-hoc test in *Functionality Disruption* reveals that users unknowingly clicked more in the gaming app with the lowest presence (P). Similarly, analysis of the *Click Manipulation* with the KW test also indicates a significant impact of "App" on MA (**p-value** $= \mathbf{0.014}, \eta^2 = \mathbf{0.47}$). Dunn's post-hoc test in *Click Manipulation* shows significantly more unintended clicks in the travel app than in the reading app. In *Peripheral Exploitation* and *UI-based Privacy Leakage*, we did not find a significant difference across apps.

Table 8: Taxonomy of attacks: description, goals, and impacts. (Please refer to Table 1 for P1-P14 descriptions. For each attack, we identified a primary malicious entity; however, colluding with other entities could amplify the attack's impact. e.g., Malvertising and GUI Switch.)

Category	Attack Name	Malicious Entity	How Various UI Properties are Exploited	Goals	Negative Consequences
Click Manipulation	GCJ [39]	Ad Service Provider	The adversary exploits P_1 to inject a JS script that places a fake cursor near the authentic one. By leveraging P_2, the authentic cursor is made transparent, deceiving the user into perceiving the fake cursor as real. This manipulation, combined with the default lack of input source validation during event registration (P_5), allows the adversary to hijack user clicks intended for game objects to nearby ads.	Increase revenue generated from ad clicks.	Users remain unaware of the illegitimate ad clicks generated by their actions. Advertisers do not receive genuine engagement with the ads and lose money. The inflated ad clicks can disrupt user experience, potentially leading to developers or ad service providers losing clients, even if they earn more money from ad clicks.
	CCJ [39]	Ad Service Provider	The adversary employs P_1 to inject a JS script that inserts a fake controller cursor, which shares the same event listeners (P_6) as the authentic cursors. To keep the fake cursor in the user's blind spot (P_7), its z-axis is rotated by 180°. Since event registration does not validate the input source by default (P_5), clicks made with authentic cursors are also registered by the fake cursor, resulting in illegitimate ad clicks behind the user. **Note:** In our user study, to enhance the attack's effectiveness and prevent the fake cursor from becoming visible when users rotate their hands in the 3D environment, we rendered the fake cursor invisible (P_2).		
	Clickjacking: Leveraging Inconsistency between Rendering and Interaction Orders [16]	Publisher	The adversary leverages P_4 to load an interactive bait object matching the shape and size of an ad, occupying the same space. Due to sequential rendering (P_{10}), the bait object, loaded later, becomes visible instead of the underlying ad. Nonetheless, P_{11} ensures that clicks are still registered by the hidden ad beneath the bait object.		
	Visual Overlapping	Publisher	The adversary places the ad behind an unclickable bait object, effectively obscuring it from the user's initial view. By exploiting P_{12}, clicks intended for the bait object are instead captured by the hidden ad. Furthermore, using P_3, the adversary ensures that functionality remains intact.		
	Sequential Rendering	Publisher	The adversary creates a transparent (P_2), unclickable object containing a genuine ad, both occupying the same space (P_4). Due to P_{10} and P_2, the later-rendered ad appears transparent. When the user clicks on any bait object behind this invisible ad, the ad captures the click due to P_{12}, bypassing the transparent object. The adversary maintains the functionality using P_3 on the next intersected entity.		
Peripheral Exploitation	BST [39]	Ad Service Provider	The adversary uses P_1 to inject a JS script that hides the ad in the opposite direction of the user's current line of sight (P_7) while dynamically adjusting the ad's position to remain hidden as the user's gaze changes.	Increase revenue generated from ad impressions or clicks.	Users remain unaware of the illegitimate ad views and clicks generated by their actions. Advertisers suffer financial losses due to a lack of genuine engagement with the ads.
	AAD [39]	Ad Service Provider	The adversary uses P_1 to inject a JS script that monitors when a user enters or exits the 3D immersive mode (P_9), and renders a video within an iframe on the active webpage on a connected auxiliary desktop monitor (P_8) when the user is within the 3D mode. The video is removed when the user exits immersive mode.		
	Input Forgery: Leveraging Synthetic User Input [16]	Publisher	The adversary detects if the ad is outside user's view (P_7), then it generates synthetic input (P_3) to increase interaction. The ad click listener does not validate the input source (P_5), thus even synthetic click will be regarded as genuine.		
	Malvertising	Advertiser	The adversary creates a partially or fully transparent (P_2) ad overlaying other content. P_1 is used to determine if the user is in 3D mode (P_9). When the user unknowingly clicks on the transparent ad (P_{12}) while attempting to interact with objects behind it, the click redirects another webpage or a muted YouTube video on the auxiliary browser screen (P_8) of the HMD, or triggers the download of malware.	Boost an advertiser's website SEO ranking or increase YouTube video views. Gain first-party cookie access for tracking.	Users' browsing history and behavior may be compromised. Malware downloads can disrupt device functionality. Ad service providers and developers risk losing potential customers.
Functionality Disruption	Denial-of-Service: Leveraging Invisibility [16]	Competitive Ad Service Provider	The adversary uses P_1 to inject a JS script to locate a target object and overlays a fully transparent object (P_2) on the same space (P_4) restricting any interaction (P_{12}) from user.	Restrict user's intended actions.	A competing ad service provider can block user interactions with ads from other ad service provider, causing associated advertisers to lose revenue.
	DoS through Overriding	Competitive Ad Service Provider	Using P_1, the adversary introduces invisible (P_2) controller cursors in a gaze based WebXR environment, overriding gaze-fusing event (P_{14}) to display false information or suppressing any warning for following click event.		
	Object Erasure: Leveraging Invisible Meshes [16]	Competitive Ad Service Provider	The adversary uses P_1 to inject a JS script to locate a target object and then loads a transparent (P_2) image in the same space (P_4) as the competing ad to erase/alter it.		
UI-based Privacy Leakage	Intercepting User Inputs: Combining Invisible Objects and Synthetic User Input [16]	Any Third-Party Entity	Using P_1, the adversary overlays transparent (P_2) overlapping (P_4) meshes in front of the PIN pad. Upon receiving user interaction (P_{12}), the malicious transparent component generates synthetic input (P_3) to pass the user's correct input to the PIN pad. Due to P_5, the PIN pad is unable to distinguish between user-generated and synthetic events.	Extract user's private information (e.g., password).	Users lose their private and sensitive information.
	GUI Switch	Advertiser	The adversary leverages P_1 to inject a JS script that programmatically captures screenshots (P_{13}), causing slower rendering and deceiving the user into exiting the immersive mode. Upon exiting immersive mode (P_9), the adversary replaces the URL with a deceptive replica created by the attacker.	Gain first-party access, enabling data collection and user tracking in 3D.	