Demo: UI Based Attacks in WebXR

Chandrika Mukherjee
Purdue University
West Lafayette, IN, USA
cmukherj@purdue.edu

Reham Mohamed Aburas
American University of Sharjah
Sharjah, UAE
raburas@aus.edu

Arjun Arunasalam
Purdue University
West Lafayette, IN, USA
aarunasa@purdue.edu

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

Z. Berkay Celik
Purdue University
West Lafayette, IN, USA
zcelik@purdue.edu

Abstract

The WebXR API enables immersive AR/VR experiences directly through web browsers on head-mounted displays (HMDs). However, prior research shows that security-sensitive UI properties and the lack of an <iframe> like element that separates different origins can be exploited to manipulate user actions, particularly within the advertising ecosystem. In our prior work, we proposed five novel UI-based attacks in WebXR, targeting the ad ecosystem. This demo presents these attacks in a unified gaming application, embedding each into distinct interactive scenarios. Our work highlights the need to address design challenges and requirements for improving immersive web-based experiences. We provide our demo video at: *https://youtu.be/lTBQbxnNq34*.

CCS Concepts

• **Security and privacy → Usability in security and privacy**.

Keywords

WebXR, Security and Privacy, User Interface-Based Attacks

ACM Reference Format:
Chandrika Mukherjee, Reham Mohamed Aburas, Arjun Arunasalam, Habiba Farrukh, and Z. Berkay Celik. 2025. Demo: UI Based Attacks in WebXR. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3711875.3734381

1 Introduction

End-users can engage in immersive 3D AR/VR scenes directly from the web browser of their head-mounted displays (HMDs) without installing additional software. This enables the development of diverse WebXR [10] applications, from entertainment to education and training, allowing users to interact using novel input methods (such as controllers and gaze) within a 360° immersive environment. In addition, various UI properties, such as transparency, overlapping objects in the same space, and synthetic input enable the design of complex scenes and interactions. However, these properties can be exploited to trick users into actions that benefit an adversary.

MobiSys '25, Anaheim, CA, USA

ACM ISBN 979-8-4007-1453-5/2025/06
https://doi.org/10.1145/3711875.3734381

A line of prior work [2, 4] has shown that any entity within the ad ecosystem, such as the developer, ad service provider, or advertiser, can employ these manipulation strategies to profit from fraudulent ad clicks and impressions or to artificially boost visibility. These attacks compromise user autonomy in immersive environments, potentially leading to data theft and malware downloads, while also causing financial and reputational harm to involved stakeholders. For example, a developer may place an ad in the same space as a bait object so that clicks intended for the bait trigger the hidden ad instead [2], enabling the developer to profit fraudulently while the advertiser incurs a loss due to the lack of meaningful engagement with their ad content.

Building on prior work, our recent study [7] analyzed the security-sensitive UI properties that contribute to nine previously proposed attacks [2, 4]. Specifically, we identified 14 contributing UI properties including two new properties (3D capture, gaze-fusing override), and proposed five novel UI-based attacks by combining them. We developed a taxonomy of these 14 attacks, categorizing them into four groups: *Click Manipulation*, *Peripheral Exploitation*, *Functionality Disruption*, and *UI-based Privacy Leakage*, based on the primary objective of the adversary. We implemented a logging framework to capture granular user interactions in the 3D WebXR environment and conducted a between-subjects study with 100 participants to assess the impact of the attacks [6, 7].

In this work, we demonstrate the effectiveness of these five UI-based attacks [7] by integrating them into a publicly available WebXR gaming app [3]. We describe how each of these attacks can be integrated into different interaction contexts and highlight the potential malicious use of security-sensitive UI properties, e.g., transparency, synthetic input, first-click interception, and auxiliary browser screen. Our demonstration highlights the critical need for user-centered design approaches that maintain action awareness in immersive environments.

2 Design: UI Attacks in Gaming

We selected an active target-shooting game [3] for our demo as it encourages high user engagement and requires continuous 360° awareness, making it well-suited for illustrating the attacks. Built with A-Frame (v1.4.0) [1] and Three.js [9], the app was modified to suit our needs. A user starts a session where targets appear randomly in 360°, earning one point per hit, with the option to replay after the session ends. To accommodate different attack contexts, we created two immersive scenes: a controller-based one featuring Visual Overlapping, Sequential Rendering, and Malvertising attacks,

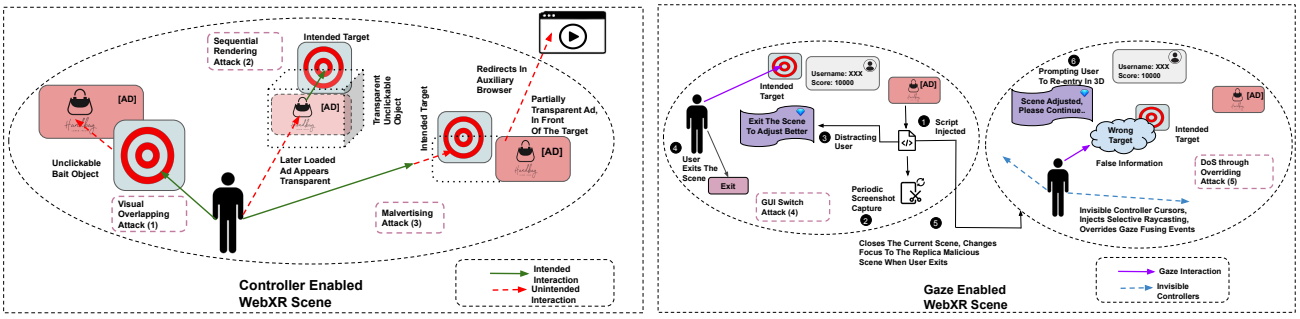

Figure 1: UI attack integration in WebXR gaming scene: (Left) visual overlapping, sequential rendering, and malvertising attacks and (Right) GUI switch, and DoS through overriding attacks.

and a gaze-based one with GUI Switch and DoS through Overriding attacks. Figure 1 illustrates the integration of these attacks into the WebXR gaming scene.

Visual Overlapping. The attack designs the start button as an unclickable bait object, placing a clickable ad directly behind it. Although partially covered, the ad remains visible and may still attract genuine user interest. However, when the user clicks the start button, the click is registered on the ad behind instead (Figure 1 Left). To maintain the illusion of expected functionality, the attack verifies whether the raycasting line to the ad intersects the bounding box of the start button. If this condition is met, it programmatically emits a synthetic click event on the button. This prevents the user from noticing any discrepancy.

Sequential Rendering. The attack is triggered when the user clicks the restart button. A transparent, unclickable object is placed in front of it, and an image or video ad is dynamically loaded inside, making the ad invisible while the restart button remains visible. As a result, when the user clicks restart, the click is registered on the hidden ad (Figure 1 Left). To preserve functionality, a synthetic click is simultaneously emitted to the restart button.

Malvertising. The attack is integrated when the user clicks on target objects to earn points. While the attack can involve fully or partially transparent ads, the latter is used here to better illustrate its impact. A partially transparent ad, with only a small visible section showing product information, is placed near and in front of the targets. Since most of the ad is transparent and its visible portion is positioned off to the side, it appears spatially distant from the target, potentially reducing user suspicion. Consequently, clicking the ad's transparent area while aiming for a target registers as an ad click, triggering a redirection in the auxiliary browser screen or initiating a drive-by download (Figure 1 Left).

GUI Switch. This attack programmatically captures scene screenshots (`components.screenshot.getCanvas`) and extracts them as data URLs, potentially enabling the adversary to infer the scene context. Periodic screenshot captures degrade user experience, especially during rapid head movements. Leveraging this, the attack creates visual distractions that may prompt the user to exit immersive mode and rejoin in pursuit of a better gaming experience. Upon exit, the script replaces the browser URL with an adversary-controlled replica constructed using information from the captured scene to appear genuine, and re-entry may lead to unauthorized 3D data collection (Figure 1 Right).

DoS through Overriding. This attack is integrated into the second scene when the user enters the compromised replica via the GUI Switch. In the gaze-based environment, it introduces two transparent controllers without the user's knowledge. These controllers' raycasting events override gaze-fusing interactions, which are typically triggered when the user initiates focus on an object. This allows the adversary to present false information, disrupt intended functionality, and potentially damage the service provider's reputation (Figure 1 Right).

3 Demonstration

We will remotely serve our app on an HMD (Meta Quest) via Glitch [8] and MQDH [5]. Attendees can wear the headset and interact with the gaming app. Their post-interaction perceptions will provide insights for further investigation.

4 Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was partially supported by NSF through grants CNS-2144645 and IIS-2229876. The findings and recommendations in this work are those of the authors and do not necessarily represent the views of the NSF.

References

[1] A-Frame. https://aframe.io/. [Online; accessed 11-Apr-2025].
[2] Kaiming Cheng et al. 2024. When the User Is Inside the User Interface: An Empirical Study of UI Security Properties in Augmented Reality. In *USENIX Security Symposium*.
[3] A-Frame Gaming. https://heyvr.io/arcade/games/wackarmadiddle. [Online; accessed: 11-Apr-2025].
[4] Hyunjoo Lee, Jiyeon Lee, Daejun Kim, Suman Jana, Insik Shin, and Sooel Son. 2021. AdCube: WebVR Ad Fraud and Practical Confinement of Third-Party Ads. In *USENIX Security Symposium*.
[5] MQDH. https://developers.meta.com/horizon/documentation/unity/ts-odh/. [Online; accessed: 11-Apr-2025].
[6] Chandrika Mukherjee, Arjun Arunasalam, Habiba Farrukh, Reham Mohamed, and Z. Berkay Celik. 2025. Towards Secure User Interaction in WebXR. In *HumanSys Workshop*.
[7] Chandrika Mukherjee, Reham Mohamed, Arjun Arunasalam, Habiba Farrukh, and Z. Berkay Celik. 2025. Shadowed Realities: An Investigation of UI Attacks in WebXR. In *USENIX Security Symposium*.
[8] Glitch Platform. https://glitch.com/. [Online; accessed 11-Apr-2025].
[9] Three.js. https://threejs.org/. [Online; accessed 11-Apr-2025].
[10] WebXR. https://www.w3.org/TR/webxr/. [Online; accessed 11-Apr-2025].