Demo: Investigating Immersive Attacks with RealityCheck

Muhammad Shoaib
University of Virginia

Wajih Ul Hassan
University of Virginia

ABSTRACT

RealityCheck, recently published at USENIX Security 2025, is the first provenance-based auditing framework that enables comprehensive root-cause and impact analysis of complex attacks against Augmented/Virtual-Reality (AR/VR) head-mounted devices. This demonstration paper describes the live, hands-on instantiation of RealityCheck, highlighting how security analysts can transparently capture multi-layer logs from commodity headsets, automatically transform these heterogeneous traces into concise multi-layer provenance graphs, and perform real-time exploratory queries to isolate attack causality. We demonstrate RealityCheck reconstructing an end-to-end provenance graph for the Object-in-the-middle attack, published at USENIX Security 2024, on a Meta Quest 2, achieving millisecond-level query latency with negligible runtime overhead.

1 INTRODUCTION

Immersive AR/VR platforms expose a far richer attack surface than traditional desktop or mobile environments; adversaries can escalate from file or network manipulation [8] to perceptual deception and spatial hijacking [3, 5, 7]. Root-cause analysis therefore requires visibility that extends well above the operating-system call layer. RealityCheck [6] meets this need by enriching the W3C PROV model with an ontology that captures AR/VR entities and interactions, by applying a domain-specific natural language pipeline to consolidate unstructured logs from five software layers, and by partitioning execution along OpenXR session states to contain dependence explosion. This present paper concentrates on the practical experience that an end user obtains when analysing attacks with RealityCheck.

2 GOALS

The paper has two goals. First, it offers an end-to-end walk-through that begins with raw log acquisition and ends with interactive provenance queries, relying solely on data collected from commodity hardware (subsection 4.1). Second, it shows how RealityCheck distills the provenance graph into a concise set of semantically meaningful vertices so that an analyst can identify root cause without manual log analysis (subsection 4.2).

3 SYSTEM OVERVIEW

Figure 1 summarises the RealityCheck frontend pipeline. From an end-user's perspective two components are most prominent. The first is a provenance dashboard implemented with `pyvis` [1]; it presents the user with a provenance graph for the requested log data. The second is a lightweight query layer that provides four functions: `find_ancestors`, `find_successors`, `backward_query`, and `forward_query`. These functions enable ad hoc exploration of the provenance graph without specialised knowledge of graph theory or database languages. Low-level details such as spaCy model training or byte-code instrumentation remain unexplored in this work because the focus is on the analyst's workflow rather than internal implementation choices of RealityCheck.

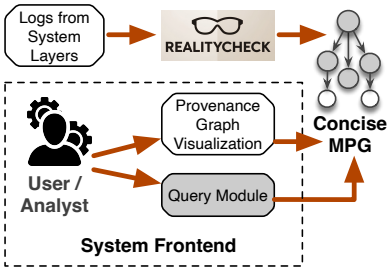

Figure 1: RealityCheck's frontend pipeline.

4 SYSTEM DEMONSTRATION

Attack Scenario. An adversary first compromises a developer-mode Windows PC that routinely connects to a consumer head-mounted display (HMD). Leveraging the *SideQuest* sideloading utility and *adb*, the attacker installs a trojanized Unity application onto the headset without user interaction. Once launched, the payload places an invisible collider over legitimate VR UI to capture a single user click. That click triggers incremental *Guardian* boundary updates which imperceptibly drift the safe zone, coercing the user to walk in a specific real-world direction, the so-called *Human-Joystick*. Concurrently, the app records controller events to a key-log file and later beacons the file location to Oculus telemetry, while the compromised PC maintains a command-and-control channel.

The resulting provenance graph, reconstructed automatically by RealityCheck, is depicted in Figure 2; the remainder of this section walks through that graph step by step.

4.1 Component 1: Provenance Dashboard

We first collected platform, application, and endpoint logs for the attack and imported them into RealityCheck. Filtering on the process identifier that corresponds to the Unity payload produced the concise multi-layer graph shown in Fig 2. The remainder of this subsection explains the sequence of vertices in that graph.

Sideload and Initial Access. At **18:10:00.000** the attacker begins by opening *SideQuest* on a compromised Windows workstation. SideQuest invokes the Android Debug Bridge (*adb*), which in turn transfers the trojanised application package *abc.apk* to the headset over a USB connection. Network telemetry collected from the same workstation shows a concurrent session to the remote host *52.89.187.149*, indicating that the attacker retains an active command channel while the sideload proceeds.

MobiHoc '25, October 27–30, 2025, Houston, TX, USA

ACM ISBN 979-8-4007-1353-8/25/10
https://doi.org/10.1145/3704413.3765306

Listing 1: Query functions provided by RealityCheck.

```
def find_ancestors(G, timestamp, entity_name):
    node = find_node(G, timestamp, entity_name)
    return nx.ancestors(G, node)
def find_successors(G, timestamp, entity_name):
    node = find_node(G, timestamp, entity_name)
    return nx.descendants(G, node)
def backward_query(G, timestamp, entity_name):
    node = find_node(G, timestamp, entity_name)
    sub_nodes = find_ancestors(G, timestamp, entity_name)
    sub_nodes.add(node)
    return G.subgraph(sub_nodes)
def forward_query(G, timestamp, entity_name):
    node = find_node(G, timestamp, entity_name)
    sub_nodes = find_successors(G, timestamp, entity_name)
    sub_nodes.add(node)
    return G.subgraph(sub_nodes)
```

Installation and Launch on the Headset. System logs record the sequence *Installing APK* followed by *Verification succeeded*, confirming that the operating system accepted the payload. Oculus *vrshell* then foregrounds the newly installed activity at **18:10:01.215**, and *ActivityManager* registers its Unity process. All subsequent events stem from *com.unity.OITM* program, signalling that the Object-in-the-Middle (OITM) [4] logic is now running on the headset.

User-Triggered Payload Execution. The first user interaction is observed at **18:10:01.623**, when Unity records an *OnPointerDown* followed one millisecond later by an *OnClick*. These callbacks arise from an invisible collider the attacker placed over a legitimate virtual control. Immediately after the click, the payload issues two *updateGuardianConfig* calls, recorded at **18:10:01.689** and **18:10:01.811**. Each call adjusts the Guardian boundary by a few units, a manifestation of the Human-Joystick attack.

Local Key-Logging and Staging. Simultaneous with the boundary manipulation, the application opens a file named */sdcard/keypresses.txt* and begins logging controller-based keystrokes. Because RealityCheck correlates this file-write with the earlier Guardian updates, the analyst can see that perceptual manipulation and data collection form part of the same execution context.

Outbound Beaconing. At **18:10:01.726** the headset contacts the Oculus push-token registration service through the component labelled *VRPlatform*. The outbound request includes a reference to *keypresses.txt*, effectively notifying the attacker that the key-log is ready for retrieval without raising an obvious network anomaly.

Forensic Implications. RealityCheck reconstructs the full sequence, sideload, installation, boundary drift, key-logging, and outbound beacon, within a single, compact multi-layer provenance graph (MPG). Because the graph is time-stamped, an analyst can move from the final Guardian update back to the original *adb* transfer in under three milliseconds. This traversal confirms both the root cause of the incident and the dual objectives of the payload: covert manipulation of user movement and theft of credential data.

4.2 Component 2: Query Module

As mentioned in section 3, RealityCheck's provenance dashboard provides four kinds of provenance queries (Listing 1). For root-cause

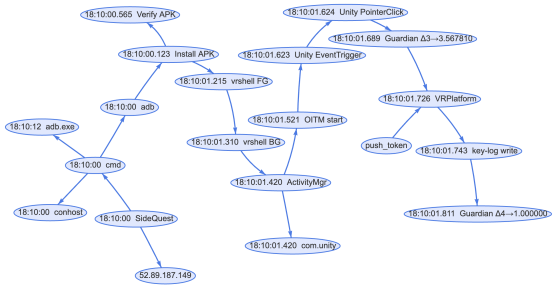

Figure 2: MPG for the Human Joystick-like OITM attack generated via RealityCheck using logs provided in the repository [2]. Vertex labels have been shortened for readability.

analysis we invoke `backward_query`, which walks the graph backwards in time until it reaches vertices with no predecessors. Calling `backward_query(G, t_{alert}, updateGuardianConfig)` where t_{alert} denotes the time-stamp of the final *updateGuardianConfig* invocation results in a sub-graph comprising thirteen vertices. The chronologically earliest vertex captures the outbound TCP flow from *adb* to *52.89.187.149*; because it has in-degree zero, it is identified as the provenance root and thus the ultimate point of control. Every other vertex, including the sideload event, the Unity process creation, the boundary shifts, and the push-token beacon, lies on at least one time-respecting path that connects this external host to the alert condition. The resulting sub-graph corroborates the narrative reconstruction presented above and provides a formally verifiable provenance chain linking the command-and-control infrastructure to the boundary-manipulation observed on the headset.

5 CONCLUSION

This work details the analyst or user facing frontend that was not thoroughly discussed in the original paper. Replaying the Object-in-the-Middle attack on a Meta Quest 2, we showed that RealityCheck captures heterogeneous logs from AR/VR devices, transforms them into a compact, ontology-enriched W3C PROV graph, and executes interactive root-cause and impact queries in milliseconds with negligible overhead. The exercise confirms that RealityCheck operationalises provenance-based forensics for immersive systems and provides a concrete foundation for future extensions to broader device classes and large-scale incident-response workflows.

REFERENCES

[1] Pyvis. https://pyvis.readthedocs.io/en/latest/.

[2] Realitycheck repository. https://github.com/DART-Laboratory/realitycheck.

[3] 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.

[4] 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 Symposium*, 2024.

[5] S. Luo, A. Nguyen, H. Farooq, K. Sun, and Z. Yan. Eavesdropping on controller acoustic emanation for keystroke inference attack in virtual reality. In *NDSS*, 2024.

[6] M. Shoaib, A. Suh, and W. U. Hassan. Principled and automated approach for investigating ar/vr attacks. In *USENIX Security Symposium*, 2025.

[7] C. Slocum, Y. Zhang, N. Abu-Ghazaleh, and J. Chen. Going through the motions: AR/VR keylogging from user head motions. In *USENIX Security*, 2023.

[8] R. Trimananda, H. Le, H. Cui, J. T. Ho, A. Shuba, and A. Markopoulou. OVRseen: Auditing network traffic and privacy policies in oculus VR. In *USENIX Security*, 2022.