SECURITY RESEARCH WHITEPAPER

EU Age Verification App: A Systemic Architectural Failure

Beyond the Surface: From Client-Side Logic Bypass to Biometric Trust Chain Collapse

Author:	Zampier Zago — Independent Security Researcher
Target:	eu.europa.ec.eudi.wallet.av (Pilot Build 2026.04-2)
Date:	April 19, 2026
TLP:	TLP:WHITE — May be freely shared
Disclosure:	Coordinated (ISO/IEC 29147:2018)
PoC Tooling:	Not distributed with this publication

ABSTRACT

This paper is my independent technical analysis of the EU Age Verification app (EUDI Wallet Pilot, build 2026.04-2). I am extending the initial audit conducted by Paul Moore (@Paul_Reviews) in April 2026. Moore already proved that you can kill the app's authentication layer in under two minutes just by manipulating plaintext XML files. But my research uncovered a much deeper, more systemic failure: the entire biometric verification pipeline—including liveness detection, face matching, and credential binding—executes without any hardware root of trust, TEE isolation, or cryptographic attestation of the sensor chain.

I have mapped out four escalating vulnerability tiers that prove the app's security guarantees are architecturally unenforceable. Surface-level patches will not fix this. My findings impact not just this pilot app, but the entire EUDI Wallet framework and the technical standards tied to the Digital Services Act.

SECTION 1: Executive Summary

The EU Age Verification app was rolled out as a pilot implementation of Article 28 of the Digital Services Act (DSA). The European Commission publicly called it a privacy-preserving, on-device identity solution. My technical analysis—conducted between April 15 and 19, 2026—shows that this characterization is pure fantasy.

The failure here is foundational. It is architectural. The app leaves the entire chain of trust—PIN verification, biometric matching, liveness detection, and credential binding—to client-side software running in Android **User-space**. It bypasses all hardware-backed security primitives sitting on modern Android devices (API Level 29+). This is a textbook case of **Security Theater**: a system that looks like it has strong identity verification but offers absolutely no cryptographically enforceable guarantees.

CRITICAL FINDING

Not a single component of the app's authentication and identity verification stack is protected by a hardware root of trust. The **Android Keystore**, **Trusted Execution Environment (TEE)**, and **Secure Element (SE)** are completely ignored. An attacker with standard developer tools (**ADB**) gains full control over every security decision the app makes.

Moore's research showed that the PIN layer can be reset in under two minutes by deleting plaintext XML entries in the SharedPreferences store. That alone proves the app uses a storage mechanism completely unfit for security-critical data. My analysis takes it further and exposes three more critical tiers: the complete lack of **Hardware Attestation** for camera input, an AI inference pipeline wide open to injection attacks in RAM, and a supply-chain disaster in the model download component that allows Man-in-the-Middle (MITM) replacement of core AI models.

You cannot secure this app with incremental patches. The DSA's mandate for robust age verification requires starting over with a fundamental architectural redesign grounded in hardware-bound credentials and remote attestation.

SECTION 2: Background & Scope

2.1 Relationship to the EUDI Wallet Framework

The app under analysis (package identifier eu.europa.ec.eudi.wallet.av) is hosted in the eu-digital-identity-wallet/av-app-android-wallet-ui GitHub repository. It is a fork of the EUDI Android Wallet reference implementation. It is written in Kotlin (99.6% of the codebase), targets API Level 29 (Android 10), and integrates the EUDIW Wallet Core SDK v0.20.0. The application implements OpenID4VP v1, DCQL, and OpenID4VCI v1 for credential exchange. It is split into distinct modules: *authentication-logic*, *business-logic*, *network-logic*, and *storage-logic*.

Because this is a white-label platform meant to be deployed by individual EU member states, any unresolved vulnerability in this shared core is going to propagate across all national rollouts.

2.2 Prior Art: Moore's Initial Disclosure

Between April 15 and 16, 2026, security consultant Paul Moore published a thread on X documenting a critical vulnerability in the local storage architecture. Moore established the following facts:

- **PIN Decoupling Vulnerability:** The app stores the user's PIN as a Base64-encoded, AES-encrypted value (PinEnc) and its Initialization Vector (PinIV) in a plaintext XML file inside the SharedPreferences store. Nuke these entries, and the app goes back to a first-run state, letting you reset the PIN without invalidating the actual verified identity credentials.
- **Client-Side Rate Limiting:** The brute-force protection counter (FailedAttempts) is stored as a plain integer in the same XML file. You can reset it via ADB, making the lockout mechanism purely cosmetic.
- **Biometric Flag Bypass:** The boolean flag UseBiometricAuth sits in the same SharedPreferences file. Set it to *false* via ADB and you completely bypass biometric authentication requirements.

- **Unencrypted Biometric Storage:** Selfie images captured during liveness verification and photographs extracted from NFC passport chips are saved as unencrypted PNG files in the app's accessible directories.

Moore also built a browser extension capable of generating forged attestation responses by replicating the QR-code verification flow, proving that relying parties perform zero cryptographic validation of where the attestation actually comes from.

SECTION 3: Vulnerability Taxonomy — Four Escalating Tiers

Here are the four discrete vulnerability classes, ordered by increasing technical difficulty and decreasing dependency on physical access to the device. Any single one of these destroys the app's identity guarantees. Combined, they represent a total collapse of the system.

3.1 Level 1 — Environment Virtualization & Integrity Bypass

Vulnerability Class: Environment Isolation Bypass

The app leans on Google Play Integrity to spot rooted devices and block execution in non-certified environments. This creates a massive architectural conflict: any privacy-focused Android fork—like GrapheneOS, /e/OS, or CalyxOS—is blocked from accessing a European Union public identity service. Citizens are forced to use a US commercial service just to exercise their rights under the DSA.

Technical Analysis:

Android virtualization platforms (what the modding community calls 'dual-space' or 'island' apps) create isolated runtime containers. They present a certified, non-rooted environment to apps running inside them, while the host system still has full superuser access and filesystem visibility. The containerized app talks to a virtualized **Camera2 API** and **Keystore**. Play Integrity only checks the container's reported state, not the actual host system.

Because the app does not use **Remote Attestation** (where the server verifies a specific, unmodified version of the app is running on certified hardware), it has no technical way to know it is running inside a malicious, attacker-controlled container.

3.2 Level 2 — Sensor Hijacking via Absent Hardware Attestation

Vulnerability Class: Sensor Hijacking

The Computer Vision pipeline takes camera frames through the standard Android **Camera2 API** without checking if the source is an actual, certified physical sensor. Android's hardware-backed Keystore offers **Hardware Camera Attestation**. This lets an app request a cryptographically signed certificate from the device's **Secure Element** to prove that a frame actually came from the physical camera. The EU Age Verification App simply does not implement this.

Technical Analysis:

Without **Hardware Attestation** for the camera feed, any process capable of registering a virtual camera provider can feed arbitrary video frames straight into the app's biometric inference engine. The app takes these frames as gospel because it performs no cryptographic verification of where the frames came from. This bypasses any software-based liveness detection.

Worse, as Moore found, the app stores passport photos and selfies as unencrypted PNG files. An attacker has all the reference material needed to feed perfect, synthetic video directly into the virtualized API layer.

3.3 Level 3 — Semantic Injection & Liveness Automation

Vulnerability Class: AI Inference Pipeline Exploitation

The app runs its entire biometric inference stack—face detection, matching, and liveness—as standard Android processes in **User-space**. It does not delegate this to a **Trusted Execution Environment (TEE)** or **Secure Enclave**. This means the outputs (similarity scores, liveness values, decisions) are computed in and accessible from standard system RAM.

Technical Analysis:

Because processing happens in **User-space**, dynamic instrumentation frameworks like Frida can attach to the running process without even modifying the APK or breaking its digital signature. An attacker can just locate the memory address where the comparison result sits and overwrite it with a fake value—forcing a 100% confidence match against any enrolled identity. This is the exposure of the **Target Bit** in **User-space**.

The liveness mechanism is equally broken. It asks for sequential behavioral challenges (blink, smile, head rotation) from a small, finite set. A background process can easily monitor the app's state, see which challenge prompt is being called, and instantly queue the matching synthetic video response to the virtual camera feed. Execution on a smartphone's **Neural Processing Unit (NPU)** introduces no delay, completely defeating timing-based defenses.

3.4 Level 4 — Model Supply Chain Compromise (ModelDownloader.kt)

Vulnerability Class: Insecure Model Distribution

This is the most severe hole. The **ModelDownloader.kt** component is responsible for grabbing the face embedding extraction model (ONNX format) from a remote server when the app initializes. It does this over unencrypted HTTP. The retrieved file is never checked against a cryptographic hash (SHA-256) and its digital signature is never validated before it is loaded into the inference engine.

Technical Analysis:

Any actor with network path access—an attacker on the same Wi-Fi, an ISP, or a malicious network operator—can intercept the HTTP request. They can swap the legitimate model with a rigged one designed to output a constant, predetermined vector. No matter what face you show the camera, the rigged model will always output a positive result. This completely neutralizes biometric verification at the model layer.

There is no check to see if the model was swapped after installation. It fails silently, providing zero biometric assurance while the app continues to operate normally.

SECTION 4: Compound Attack Surface Assessment

Fixing any single one of these tiers won't restore security. The remaining tiers are still independently exploitable.

Analysis of the credential issuance flow also shows a massive issuer-side vulnerability: the EU member state authority signing the attestation has no way to verify that the NFC passport scan actually occurred in a secure, tamper-free environment on the user's device. Pre-authenticated 'Over 18' credentials could easily be manufactured and transferred across devices, creating a black market for digital identities.

SECTION 5: Architectural Remediation Roadmap

The vulnerabilities I have documented are not fixable with quick patches. They require a complete rewrite focused on hardware isolation. If this system is ever going to satisfy the security level required by DSA Article 28, these remediation steps are non-negotiable:

- **R-1: Hardware-Bound Credential Storage:** All security-critical state (PIN keys, biometric templates, session tokens) must be generated within and confined to the **Android Keystore** in StrongBox mode. SharedPreferences XML files are absolutely forbidden for this data.
- **R-2: TEE-Isolated Biometric Inference:** Face matching and liveness detection must run within the device's **TEE**. Raw biometric embeddings and similarity scores must never be exposed in **User-space** RAM.
- **R-3: Camera Hardware Attestation:** The app must use **Camera Hardware Attestation** to cryptographically verify that input frames originate from a certified physical sensor. Virtual camera streams must be rejected at the API layer.
- **R-4: Model Integrity Verification:** All AI models must be verified against hardcoded SHA-256 checksums and digital signatures. Downloads must exclusively use TLS 1.3 with certificate pinning.
- **R-5: Remote Attestation for All Sessions:** Every session must be anchored by a **Remote Attestation** flow where the server receives a signed assertion proving the app is unmodified and running on a certified hardware platform.
- **R-6: Biometric Data Lifecycle Enforcement:** All biometric data must be confined to volatile memory (RAM) strictly for the duration of the verification session. Persistent storage of this data is a direct violation of GDPR Article 9.

SECTION 6: Disclosure Ethics & Coordinated Reporting

I have conducted and disclosed this research in accordance with **Coordinated Vulnerability Disclosure (CVD)** principles (ISO/IEC 29147:2018). I am not distributing any Proof-of-Concept code or functional attack tools with this paper.

Because the European Commission has already described the affected version as a pilot, and because Moore's research already generated massive public discussion on the security of this system, I consider this public architectural teardown appropriate and in the public interest.

*This document is classified **TLP:WHITE** and may be freely shared. The author asserts that no functional exploit code or attack tooling is distributed with this publication.*