USENIX WOOT '26 Artifact Appendix: Defective by Design: Universal Recovery of All Widevine-Protected Content on Desktop Environments

Florian Roudot
Univ Rennes, CNRS, IRISA

Mohamed Sabt
Univ Rennes, CNRS, IRISA

A Artifact Appendix

A.1 Abstract

Nowadays, streaming services, such as *Netflix*, rely on Digital Rights Management (DRM) systems to deliver their protected content. These systems aim to prevent piracy. Specifically, non-subscribers are prevented from accessing the content altogether, while subscribers are prevented from acquiring decrypted copies of the media to avoid uncontrolled distribution. Among the currently deployed DRM systems, Google Widevine is the most widely used, especially on desktops, where it provides a fully software-based solution.

In this paper, we investigate Widevine's decryption interface and its integration in modern web browsers. We show that Widevine's boundary (*i.e.,* its output after media decryption) is inherently unprotected and can be intercepted with relative ease. Under an attacker merely observing this interface, we show that audio content can be trivially recovered because the decrypted samples are returned prior to decoding. We further identify that, under a commonly used Widevine configuration, the same "decrypt only" behavior also applies to video, enabling direct recovery of video frames. When this misconfiguration is absent, we show that Widevine still outputs decrypted and decoded frames that can be efficiently re-encoded with negligible quality degradation.

Based on our findings, we build an attack that "downloads" any content protected by Widevine into a playable format on both Linux and Windows. Finally, we assess the effectiveness of our attack by applying it to premium streaming platforms.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Executing the attack does not present any security risks to the user's device. The exploited vulnerability has been responsibly disclosed to Widevine, which did not consider it severe enough to warrant a patch. The protected content used for the attack is publicly accessible on Widevine demonstration website[1].

[1] https://integration.widevine.com/player

A.2.2 How to access

The artifact is accessible on https://doi.org/10.5281/zenodo.20024534.

A.2.3 Hardware dependencies

A standard x86-64 computer is sufficient to compile and run the artifact.

A.2.4 Software dependencies

The attack can be performed on Linux (verified on Ubuntu 24.04.3) and Windows (verified on Windows 11 Enterprise 25H2), either on physical machines or virtual machines. For each platform, the required software dependencies are listed below:

On Linux.
- g++[2], cmake[3] and make[4].
- libavutil-dev, libavcodec-dev, libavformat-dev, libswscale-dev from FFmpeg[5].
- Firefox[6] and VLC[7]

On Windows.
- Visual Studio[8] with the *Desktop development with C++* kit.
- Firefox[6].

A.3 Set-up

A.3.1 Installation

As described in the paper, we implemented two versions of the attack depending on the security level of the content keys. The content distributed on the Widevine demonstration website is protected with the lowest security level, allowing us to decrypt it directly without re-encoding. The second method,

[2] https://gcc.gnu.org/
[3] https://cmake.org/
[4] https://www.gnu.org/software/make/
[5] https://www.ffmpeg.org/
[6] https://www.mozilla.org/firefox/
[7] https://images.videolan.org/vlc/
[8] https://visualstudio.microsoft.com/

which involves re-encoding, can still be enforced by setting the preprocessor macro `REENCODE` in `Processor/hook/src/processor.cpp`. For faster execution, we set the AV1 encoder preset to 13 (on a scale from 1 to 13), but you can adjust this value in `Processor/hook/src/encoder.cpp`.

On Linux.

1. Install all the required dependencies. On Ubuntu, they are all available in the default APT repositories, except for Firefox-ESR, which requires configuring the Mozilla APT repository[9].
2. Compile the attack executable by running `cmake . -B build && make -C build/` in `Processor/`.

On Windows.

1. Download (https://visualstudio.microsoft.com/downloads/) and install Visual Studio. When asked which additional components to install, select *Desktop development with C++* and continue.
2. Download and install the latest version of Firefox.
3. Double click on `Processor/DefectiveByDesign.sln` to open the solution in Visual Studio then build it with *Build* > *Build Solution*.

A.4 Evaluation workflow

A.4.1 Major Claim

This artifact enables the reproduction described in our paper. When visiting a website that plays protected content using Widevine, an attacker can use the artifact to "download" a DRM-free version of the content, allowing for its redistribution and unlimited playback.

A.4.2 Experiment

(E1): [Decrypt Attack] [A few seconds to minutes]: Run the attack to retrieve decrypted content.
Preparation: Ensure that all instances of Firefox are closed.
Execution: Open a terminal in `Processor/` and run:
`./RUN.sh` on Linux.
`./RUN.ps1` on Windows.
On both systems, once Firefox has opened, proceed as follows. In the first tab, click *Load Temporary Add-on...* and select any file from `Web-Extension/` to load the extension. In the second tab, start video playback using the play button, then open the *Widevine Downloader* extension. The protected tracks should appear in the extension's pop-up. Select the ones you want to download and decrypt.
Results: Once the track is downloaded, the video should stop playing, indicating that Widevine is busy decrypting the downloaded file. Depending on the track resolution and the encoder preset, the process may take a few minutes to complete. The resulting decrypted files are typically stored in `~/Downloads/WidevineMedia`.
Troubleshooting: Logs are written to `Processor/`. When attempting the attack multiple times, ensure that all instances of Firefox are fully closed (verify using the system's process list) before re-running the launch script.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/woot2026/.

[9] https://support.mozilla.org/en-US/kb/install-firefox-linux#w_install-firefox-deb-package-for-debian-based-distributions-recommended