STATEAM LLC · GOWORDTODAY PRODUCTIONS

White Paper

VerseCut 3.0

A local-only video editor with offline licensing

Why organisations that produce video regularly should stop running a consumer subscription editor on a working computer, and what a local-first alternative looks like in practice.

Field	Detail
Document	VerseCut 3.0 White Paper
Version	1.1 — updated for VerseCut 3.0
Date	September 2026
Product release	VerseCut 3.0.0 (Windows 10/11; macOS 12 Monterey or later, Apple silicon and Intel)
Publisher	Stateam LLC, Princeton, New Jersey
Prepared by	Ben John, gowordtoday productions
Audience	Buyers, technical evaluators, security reviewers and partners
Contact	capture@stateam.net

Abstract

For any organisation that produces video on a schedule — a church or ministry, a school, a small studio, a clinic, an agency — the editing application is not a consumer purchase. It is production infrastructure installed on a working computer that also holds unreleased footage and sensitive material. The mainstream consumer suites have moved the other way: mandatory accounts, background updater services, updates that install without being asked, and advertising inside the working interface. Each is a real operational cost and, on a machine that matters, a real security consideration. This paper makes the case for a different shape of tool and describes one: VerseCut 3.0, a local-only editor with no account, no telemetry and no automatic updates, whose licence is a signed file verified on the customer's own computer with no server and no connection. It covers the architecture, the security posture including its limits, the licensing design and its commercial reasoning, the standards position, market pricing, and a checklist for verifying the claims independently.

Contents

1. The problem: the working computer as a marketing surface

An editing application occupies an unusual position in an organisation's estate. It is installed with broad access to the file system, it runs for hours at a time, it is frequently left open on a machine in a shared office or a control room, and it touches the most sensitive material the organisation produces: footage that has not been released, recordings of people who have not yet consented to publication, and in some settings material that is confidential by law. It is, in short, infrastructure. It is nonetheless bought and installed like a consumer product, and over the past several years the consumer product category has changed in ways that make that mismatch expensive.

Four changes are now close to standard across the mainstream consumer editing suites. The application requires an account, so the organisation's access to its own archive depends on a third-party sign-in. The application installs a background service or scheduled task that keeps running when the editor is closed, in order to check for updates and, in some products, to prepare content or upload diagnostics. The application updates itself without asking, on the vendor's schedule rather than the customer's. And the application places promotional material inside the working interface: upgrade prompts, feature advertisements, seasonal offers and cross-sells to other products from the same vendor.

None of these is a scandal. Each is a defensible commercial decision in a market where the software is cheap and the vendor is trying to retain and upsell an individual consumer. The argument of this paper is narrower and, we think, harder to dispute: taken together they impose costs on an organisation that a consumer does not pay, and those costs are not disclosed at the point of purchase.

1.1 Four operational consequences

Unannounced updates change behaviour mid-project. A production that runs weekly has a fixed routine: the same project template, the same export preset, the same export destination, the same keyboard shortcuts. An update that arrives on a Thursday can move a control, change a default, alter an encoder setting, or in the worst case fail to open a project created the previous week. The organisation discovers this not during an evaluation window but during the hours it has allocated to finishing. The cost is not the update; it is that the timing of the change belongs to someone else.

Background services and auto-updaters widen the attack surface. A background updater is, by design, a privileged process that runs continuously, reaches out to the internet, downloads code and executes it. That is a reasonable description of what an attacker wants on a machine. The risk is not hypothetical or unique to any one vendor: update channels are a well-understood supply-chain target precisely because they are trusted paths for code to arrive on a machine without a person deciding. On a general-purpose home computer that trade-off is often worth it. On a machine that holds unreleased footage, source recordings of identifiable people, or material that an organisation has undertaken to keep confidential, the calculation is different, and the person who has to defend it is usually not the person who chose the editor.

An account requirement puts a third party between an organisation and its own archive. If the editor cannot be used without signing in, then a lapsed payment, a lost password, a disputed charge, a change in the vendor's regional availability, a corporate acquisition or an outage at the vendor can all place the organisation's own project files temporarily out of reach. The files are still on the disk; what has been lost is the ability to open them. For a small organisation with no IT department and one person who knows the password, that is a single point of failure that nobody deliberately designed.

Advertising in a working tool costs attention, and sometimes more than attention. Staff time spent dismissing prompts is small per event and significant per year, and the interruption lands during the concentrated work that an editor is for. In some settings the problem is not attention but appropriateness. An editing station in a ministry office, a school classroom or a clinic is frequently visible to volunteers, students, patients or members of the public. Promotional content that the organisation did not select and cannot suppress is being displayed on the organisation's screen, on its premises, in its name. Most vendors do not allow it to be turned off; at least one states plainly in its own support documentation that its in-product promotions cannot be completely disabled.

1.2 Why VerseCut exists

VerseCut was not conceived as a market analysis. It began in a weekly Bible-teaching production at gowordtoday productions, where the routine was fixed and demanding: a long teaching video, dozens of on-screen scripture cards each landing on an exact frame, two logos, captions for publication, and a deadline every week. The decision to build a replacement editor followed a specific experience — a paid corporate subscription delivering unsolicited advertising and unrequested updates onto a work computer. The organisation judged that combination a security risk: software that could place content on the screen and code on the disk, on its own schedule, on a machine that held unreleased material.

That judgement is the origin of the product, but the point of this paper is that the experience was not unusual. The pattern is widespread across the category rather than the behaviour of one vendor, which is why the answer could not be to switch to a different consumer subscription. It had to be a different architecture.

> *The question is not whether a particular vendor behaved badly. It is whether an organisation should accept a tool that can change itself, advertise to its staff and gate access to its archive — when none of those properties is necessary to edit video.*

1.3 Who carries the cost

The organisations affected most are the ones with the least capacity to absorb it. A broadcaster has a managed image, a change-control process and staff whose job is to hold vendors to it. A two-person ministry media team has a laptop, a deadline and whatever the software does on the day. The same is true of a school media club, a single-clinician practice producing patient education material, a small agency, or a charity with one communications officer. These organisations produce video regularly enough to depend on the tool, and are small enough that the tool's behaviour is simply imposed on them. They are the intended readers of this paper.

2. Design principles

VerseCut is built on five principles. They are stated here as commitments, and each one is expressed somewhere a reviewer can check it: in the process model, in the source, or in the format of the files the product writes. Section 9 lists the specific checks.

2.1 Local only

There is no account, no sign-in and no cloud library. There is no telemetry, analytics, crash reporting or usage measurement of any kind. There are no silent updates. Media is never uploaded: import, preview, transcription, rendering and export all happen on the machine the software is installed on. The installer needs the internet once, to fetch Python, the speech-to-text packages and the speech model (the video engine is included in the download); after that the application works with no connection at all, including licence activation.

2.2 The tool is inspectable

VerseCut is a small Python engine plus a web interface that is rendered by a browser already present on the machine. It ships as readable source rather than an opaque binary, so a security team can read what it does instead of inferring it from network captures. That is a deliberate trade — it is discussed honestly in section 4.6 — but for a customer whose objection to the incumbent is that they cannot tell what it is doing, being able to read the code is the direct answer.

2.3 No background services

Nothing runs when the editor is closed. VerseCut installs no service, no daemon, no scheduled task, no login item and no updater. The engine is started by the launcher, and it stops by itself: a watchdog checks every fifteen seconds and exits the process once the editor window has been closed for two minutes and no render or transcription is still running. A machine with VerseCut installed and not in use is, from the operating system's point of view, a machine with nothing extra running on it.

2.4 Deterministic output

A render is not a black box. The engine converts the project into a single FFmpeg command with an explicit filtergraph, written to a script file and handed to FFmpeg as a separate program. The graph can be printed and read. Transitions are not an abstract effect applied at render time; they are expanded into concrete fades and overlaps in the clip list before the graph is built, using the same rules the preview uses, so what is seen in the preview is what is described in the graph. The same project, exported twice with the same settings on the same machine, produces the same output.

2.5 Plain formats the customer owns

Nothing VerseCut writes requires VerseCut to read. Project files are JSON. Captions are SRT, WebVTT or ITT. Exports are H.264 MP4 or MP3. Timed overlays are imported from FCPXML, CSV or timecodes in file names. Projects live in the customer's own Documents folder and exports in the customer's own Movies or Videos folder. If the company ceased trading tomorrow, every file a customer has produced would remain openable with standard tools, and the projects would remain readable text.

Table 1. Each principle, the mechanism that implements it, and how to verify it

Principle	Mechanism	How an evaluator checks it
Local only	No account code paths, no analytics client, no update client; engine bound to the loopback interface	Disconnect the network after setup and complete a full edit and export; monitor outbound connections
Inspectable	Python engine plus browser-rendered interface, shipped as readable source	Read app/server.py and app/engine/*.py; no decompilation required
No background services	Watchdog exits the process two minutes after the window closes; no service, task or login item is installed	Close the editor, wait two minutes, inspect the process list and the service and task registries
Deterministic output	Project compiled to one FFmpeg command with an explicit filtergraph in a script file	Export the same project twice and compare the files; read the generated graph
Customer-owned formats	JSON projects, SRT/VTT/ITT captions, MP4/MP3 exports, files in the customer's own folders	Open a .wcproj in a text editor; open an export in any player

Table 1: the five principles are properties of the build, not statements of intent, and each can be confirmed without the vendor's cooperation.

3. Architecture

VerseCut is a desktop application assembled from two familiar parts: a local engine written in Python, and an interface written for the browser. Neither part is novel. The design decisions worth describing are the ones that keep the combination behaving like a desktop application rather than like a website.

3.1 The engine and the window

The engine is a Python ThreadingHTTPServer bound to 127.0.0.1 — the loopback interface — and to no other address. It selects the first free port from 8765 upward and records it so that launching VerseCut a second time opens another window on the copy already running rather than starting a second engine.

The interface is opened as a plain application window. On Windows the launcher starts Microsoft Edge or Google Chrome with the --app= flag and a dedicated user-data directory; on macOS it does the same with Chrome or Edge. An --app window has no tabs, no address bar, no bookmarks and no extensions from the user's normal browsing profile, because the profile is VerseCut's own. To the person using it, it is an application window. Where no suitable browser is present, the launcher falls back to the default browser.

3.2 Why a local HTTP server needs a security model

Any program on a machine can send a request to a loopback port, and so can a web page open in a browser on the same machine. A local HTTP server with no controls is therefore a way for other software to drive the editor: to enumerate the file system through the editor's own file-browsing endpoint, read media files, or start a render. VerseCut applies two independent controls to every request.

- **Host header check.** A request is rejected with 403 unless its Host header is 127.0.0.1 or localhost. This blocks DNS rebinding, where a hostile page persuades a browser to resolve an attacker-controlled name to the loopback address and then speak to a local service as though it were the site's own origin.
- **Per-launch token.** At start-up the engine generates a token with secrets.token_urlsafe(24) — 24 bytes from the operating system's cryptographic random source — and every request other than the initial page and its static assets must carry it, either in an X-VerseCut-Token header or as a query parameter. The token is substituted into the page as it is served, so the editor window holds it and nothing else does. It is new on every launch and is never written anywhere a different program would routinely read.

The practical effect is that a program or a web page on the same machine cannot drive the editor without first reading the token out of VerseCut's own window, which requires a level of access to the machine at which the editor is no longer the weakest thing on it. Responses are sent with Cache-Control: no-store, and static assets are resolved and confined to the application's own web directory so that a crafted path cannot walk out of it.

3.3 Media, jobs and progress

Media is served to the window over HTTP with full Range request support, so the browser can seek into the middle of a multi-gigabyte recording without transferring what comes before it, and scrubbing the timeline stays responsive. Files are streamed in one-megabyte chunks and a dropped connection — which is what a seek looks like from the server's side — is handled and ignored rather than logged as an error.

Long operations run as background jobs: rendering, transcription and scripture card import. Each job has an identifier, a status, a progress fraction and a message, and the interface polls it. Every job can be cancelled, which kills the underlying process rather than waiting for it. Finished jobs are held only long enough for the interface to collect the result and are then discarded. Media scanning is deliberately bounded: a status sweep asks about no more than 120 items at a time and answers from the registry and the on-disk cache without starting a probe, so no amount of polling by the interface can load the machine.

3.4 Speech to text

Transcription uses faster-whisper running on the CPU with int8 quantisation, with the Whisper model weights downloaded once at setup and stored locally. The engine renders the voice tracks to a 16 kHz mono WAV file in a temporary directory, transcribes it, reports progress as it goes, and deletes the temporary file. Nothing is uploaded, no API key is required and no per-minute charge applies. Captions can be exported as SRT for publication, WebVTT for the web, or ITT for Final Cut Pro.

3.5 The editing model

A new project opens with eight tracks, arranged for the kind of production VerseCut was built in. The arrangement is a default rather than a limit.

Track	Name	Purpose
T1	Titles	Lower thirds, centred titles, name tags and scripture-card styles, drawn natively at export resolution
V4	Logo (top)	Upper logo or bug, with one-click corner placement and drag-to-position in the picture
V3	Logo (bottom)	Second logo or persistent graphic
V2	Scripture cards	Timed overlays, placed automatically from an imported card set
V1	Main video	The principal footage; the compositing base
C1	Captions	Caption cues, burned in on export and exported as SRT alongside
A1	Voice / SFX	Additional voice and effects
A2	Music	Music bed, ducked under the voice automatically

Table 2: the eight default tracks. V1 to V4 composite from bottom to top; extra tracks can be added.

Editing is overwrite-based: a clip lands exactly where it is dropped and takes the space it occupies. Splitting, ripple deletion to close gaps, drag-and-resize positioning within the picture, and snapping to safe margins are all present. Undo reaches back 200 steps and projects save automatically.

The distinctive capability is timed overlay import. A production that puts a scripture reference, a lyric or a speaker name on screen dozens of times per episode normally places each one by hand. VerseCut reads a set of card images together with their timing — from an FCPXML sheet, a CSV, or timecodes embedded in the file names — and places every card on its exact frame with a fade in and a fade out. Twenty-five cards, twenty-five exact frames, one operation.

3.6 Rendering and audio

At export the project is compiled into a single FFmpeg invocation. Transitions — cross-dissolve, dip to black, dip to white — are expanded before the graph is built into concrete fades and clip overlaps, with the audio crossfaded to match, using the same rules the preview applies. Dips are composited immediately above the track they belong to.

Audio is mixed with music ducking implemented as a genuine sidechain compressor keyed off the voice, not as a static level cut, so the music recovers between phrases. The final mix is loudness-normalised to −14 LUFS with a true peak ceiling of −1.5 dBTP, which is the target the major platforms normalise to, and an optional voice cleanup pass removes rumble and hiss.

Video is encoded to H.264 in MP4 at 3840×2160, 1920×1080 or 1280×720, or to MP3 for audio-only publication, using the H.264 encoder provided by the operating system (section 6.2). If a hardware encoder fails on a particular machine — which happens, and usually because of a driver — the engine reports it and retries the same graph on the CPU rather than failing the export.

4. Security posture

This section is written for the person who has to approve the installation. It states the position plainly, including the parts that are not advantages.

4.1 Attack surface

VerseCut's only listening socket is an HTTP server bound to 127.0.0.1. It is not bound to any routable address, so there is no port reachable from the local network or the internet, and no inbound exposure exists to be firewalled. Requests from the machine itself are filtered twice: by the Host header check and by the per-launch random token described in section 3.2. The server exists for the lifetime of an editing session and then the process exits.

VerseCut makes no outbound connections during normal operation. There is no update check, no licence check-in, no analytics endpoint, no crash reporter and no content delivery call. Telemetry in the bundled machine-learning stack is disabled explicitly at start-up. The only network activity in the product's life is the one-time setup, which fetches Python, the pinned Python packages and the speech model. The video engine is part of the download itself.

4.2 Supply chain

Three classes of third-party code enter the installation, all at setup time and never afterwards.

- **Python packages** are installed once from PyPI into VerseCut's private runtime, every one pinned to an exact version by a requirements file that ships with the release, and wheels only. faster-whisper is installed without its PyAV dependency, which bundles a GPL build of FFmpeg that VerseCut does not need. Nothing is refreshed afterwards. On macOS VerseCut brings its own Python 3.12, checked against a published SHA-256, inside VerseCut.app.
- **FFmpeg** ships inside the VerseCut download as an LGPL build with no GPL or non-free components, and runs as a separate program: on Windows an unmodified BtbN build of FFmpeg 9.0, on macOS FFmpeg 9.0.2 built by Stateam with network protocols switched off. The installer checks every file against a SHA-256 list before installing it, so the exact build on any machine is known. The complete corresponding source is published on the VerseCut website and supplied on request. Because FFmpeg is not linked into VerseCut, an organisation may substitute its own build of the same libraries.
- **The Whisper model weights** are downloaded once at setup and stored locally (on macOS outside the application, so reinstalls keep them). They are data, not executable code, and they are not updated afterwards.

4.3 No auto-update mechanism

VerseCut contains no code that can install a new version of itself. This is not a setting that can be toggled or a policy that could be reversed by the vendor; the capability is absent. Nothing on a working machine can change unless a person chooses to install it, which means a validated configuration stays validated until the organisation decides otherwise. The cost of this decision is borne by the customer: keeping up to date is a manual act, and an organisation that wants the current release must go and get it. We consider that trade the right way round for production machines.

4.4 The licence check cannot be used against the customer

Because activation is a signature check performed locally (section 5), there is no channel by which an installed copy could be remotely disabled, downgraded or audited. Stateam holds no database of customer machines and no session that could be revoked. A customer who has bought a licence and installed it owns a working copy whose continued operation does not depend on the vendor's infrastructure, the vendor's solvency or the vendor's goodwill. That property is unusual and it is deliberate.

4.5 Data at rest

Projects are written to the customer's Documents folder and exports to the customer's Movies or Videos folder. Media is read from wherever it already is and is not copied into a hidden library. Temporary files created during a render or a transcription live in a system temporary directory and are removed when the operation ends. Nothing is written outside the customer's own folders and the application folder, and no copy of any media leaves the machine.

4.6 What this does not protect against

An honest security section names its limits, and these are the ones that matter.

- VerseCut does not encrypt project files or exports. They are ordinary files with ordinary permissions. Confidentiality at rest is the operating system's job, through full-disk encryption and file permissions, and an organisation handling sensitive footage should be using both regardless of which editor it runs.
- VerseCut does not manage identity. It does not decide who can log in to the computer, it has no roles or permissions of its own, and it does not audit who did what. Anyone with an interactive session on that machine can use the editor and read the files that session can read.
- VerseCut's source being readable means that a person using that computer can modify their own copy, including the parts that enforce plan limits. That is the same trade as inspectability, taken in the other direction; section 5.6 explains why we accept it.
- VerseCut does not defend against a compromised machine. If an attacker already runs code as the user, the editor's token and files are available to them, as is everything else that user can reach. No application-level control changes that.
- VerseCut renders with FFmpeg and decodes with the operating system's own media components. A vulnerability in either is inherited, which is the usual reason to keep the platform patched and to be deliberate about the provenance of media files.

None of these is unusual for a desktop application. They are stated because a reviewer will find them anyway, and a paper that omitted them would deserve to be distrusted on everything else.

5. The licensing model and why it is built this way

Licensing is normally where a product with these principles compromises, because the ordinary way to sell software is to make the software ask a server for permission. VerseCut does not have a server to ask. This section describes what replaces it and the commercial reasoning that makes the replacement workable.

5.1 What a licence is

A VerseCut licence is a small text file — short enough to paste into an email — carrying a payload and a signature between BEGIN and END markers. The payload is a JSON object with the following fields.

Field	Meaning
id	The licence identifier, used for support and for reissue
name, email	The holder, so the licence is attributable to a person or organisation
plan	free, creator or studio
term	annual or perpetual
seats	How many people the licence covers, for Studio
issued	The date of issue
expires	For an annual licence: the renewal date
lockVersion	For a one-time licence: the release line it covers, for example 3.0
machines	Optional. Present only where a customer has asked for the licence to be locked to specific computers

Table 3: the licence payload. Nothing in it is collected from the customer's machine; every field is known at the point of sale.

5.2 Canonical serialisation and Ed25519

The payload is serialised canonically before signing: compact JSON with sorted keys and no whitespace, so that the same object always produces exactly the same bytes. Those bytes are signed with Ed25519, the signature scheme standardised as RFC 8032. The application carries only the public key; the private key never leaves Stateam.

Verification is implemented in a dependency-free pure-Python module inside the application. That choice is deliberate. If verification were delegated to a third-party cryptography package, the check could be defeated simply by removing or replacing that package — a failure mode that has embarrassed more than one product. With the implementation inside the application, there is no library to strip. Ed25519 verification is a few milliseconds once per launch, so the performance cost of a pure-Python implementation is irrelevant.

Because the whole check is a signature verification against a key already present on the machine, it requires no server, no account and no internet. Activation works on a computer that has never been online. Stateam operates no activation service, keeps no machine database, and therefore holds nothing about a customer's hardware that could be leaked, subpoenaed or sold. Where a customer prefers the opposite trade and wants a licence bound to named machines, VerseCut displays a Machine ID — a one-way hash containing no name, serial number or network address — which the customer can send in to have their own licence locked.

5.3 The commercial logic

An annual subscription is the core model. It is what makes the revenue recur, and recurring revenue is what funds the maintenance of a product that has no advertising, no data business and no upsell surface. A subscriber always receives the newest release for as long as the subscription runs, including new major versions.

Alongside it, VerseCut offers a one-time licence at a single stated price. It never expires. It covers one release line: a one-time Creator licence for the 3.0 line includes every 3.0.x update for good, while 3.1 and later lines require an upgrade purchase at the price set for that line. That boundary is

enforced in the licence itself through the locked release line, and the message a customer sees when they encounter it says plainly that the version they bought keeps working.

The one-time option exists because some of the customers this product is for cannot responsibly enter an open-ended commitment. A small charity budgets by the year, a school buys from a capital line, a volunteer-run media team may not have a card that can carry a recurring charge. Offering them a licence they buy once and keep is not a concession; it is the same promise the rest of the product makes, expressed commercially. It is also, as section 7 shows, the only way in this category to stop paying and keep working.

Table 4. Plans, prices and what each one unlocks

Plan	Price (US)	What it unlocks	Terms
Free	$0	Full editing on the whole timeline, 1080p export with no watermark, titles and fades, caption import and export, 3 saved projects	Permanent. No account, nothing to cancel
Creator trial	$0	Every Creator feature	14 days from first launch. No card, no account; steps down to Free when it ends
Creator	$59 a year, or $7.99 a month	Unlimited projects, 4K export, offline transcription, timed card import, transitions, the two logo tracks, commercial use, 2 computers	Always the newest release while it runs. 14-day grace period after the renewal date
Creator, one time	$89 once	The same Creator features	Never expires. Covers the 3.0 release line, including every 3.0.x update; later lines need an upgrade
Studio	$119 per seat a year	Everything in Creator, plus seats for a team, shared templates, priority support and invoicing	Three seats or more. Always the newest release while it runs
Studio, one time	$179 per seat once	The same Studio features	Never expires. Covers the 3.0 release line only
Nonprofit and ministry	30% off any plan	As the chosen plan	Verified 501(c)(3) or the equivalent in the customer's country

Table 4: prices are US list prices as at September 2026. Paddle acts as merchant of record and collects and remits VAT and sales tax.

5.4 The trial, the grace period and the step-down

The first launch starts a 14-day Creator trial with every Creator feature available. There is no account to create, no card to enter and nothing to cancel. The trial clock is kept in a local state file protected by a machine-derived message authentication code, and it records the latest date the installation has ever seen, so winding the computer clock backwards does not extend it.

An annual licence that passes its renewal date does not stop at midnight. For 14 days afterwards the software continues at the full plan, with a message stating the date it expired and how many days of grace remain. The reason is operational rather than generous: a payment that fails on a Tuesday should never be the thing that stops a broadcast on a Wednesday.

When the grace period ends, and when a trial ends, VerseCut does not lock. It steps down to the Free plan. Every project remains present and openable, editing continues on the whole timeline, and 1080p export with no watermark still works. Only the paid features stop. This is the most consequential decision in the whole licensing design, and it is the one we would defend hardest: a customer who stops paying has not stopped owning their work, and a tool that holds a deadline hostage to a lapsed card has made itself the risk that section 1 described.

5.5 Where enforcement lives

Plan limits are enforced in the engine, not in the interface. The engine evaluates the current entitlement on every request, caching it for a few seconds, and the gates sit in the operations themselves: an export above the plan's maximum height is refused by the export endpoint, transcription and card import are refused before the job starts, a project save beyond the Free plan's limit of three is refused by the save endpoint, and a project containing transitions or content on the two logo tracks is refused at export under a plan that does not include them. The interface reflects the plan; it does not decide it. Hiding a button in the interface is presentation, not enforcement, and putting the check anywhere else would make the entitlement a matter of what the window happened to display.

Refusals are returned as HTTP 402 with a message that names the feature, names the current plan, and says exactly where to enter a licence. A customer who hits a limit should never have to guess what happened.

5.6 What the scheme is for

No licensing scheme that runs entirely on the customer's own computer can make copying impossible, and VerseCut's is no exception: the source is readable, so a determined person on that machine can modify their copy. We are not attempting to prevent that, and a paper claiming otherwise would be arguing with arithmetic.

The aim is different. It is to make honest purchase easy: one price, stated plainly; a file that arrives by email and activates in seconds with no account to create; no reactivation when a machine is rebuilt; no penalty when a renewal is late; and no circumstance in which paying customers lose access to their own work. A scheme judged by that standard is succeeding when buying is simpler than not buying. A scheme judged by how hard it is to defeat would have led us back to an activation server, and the activation server is precisely what this product exists to avoid.

> *The signature check is not a lock on the customer's door. It is a receipt they hold, that we cannot take back.*

6. Standards, licensing and compliance

6.1 Open-source components

VerseCut is built on open-source software and says so, in the product and in the shipped notices file. The components and their terms are as follows.

Component	Licence	How VerseCut uses it
FFmpeg	LGPL (v3 on Windows, v2.1+ on macOS); no GPL or non-free components; shipped with VerseCut	Downloaded once by the installer and run as a separate program for decode, preview and export. Unmodified; complete corresponding source supplied on request
faster-whisper (SYSTRAN)	MIT	Offline speech to text
Whisper models (SYSTRAN conversions)	MIT	Speech model weights, downloaded once at setup
CTranslate2	MIT	Runs the speech model
Python 3.12	PSF licence	Runs the VerseCut engine
Lora, Poppins, Source Sans 3, JetBrains Mono	SIL Open Font License 1.1	Brand and interface typography

Table 5: third-party components. The full licence texts ship with the product.

6.2 H.264 encoding, the GPL and patents

VerseCut bundles no GPL-licensed encoder. It encodes H.264 video and AAC audio only with the encoders that are part of the operating system and that the platform vendor licenses: Media Foundation (and Intel Quick Sync, NVIDIA NVENC or AMD, whichever the machine provides) on Windows, and Apple VideoToolbox and AudioToolbox on macOS. The engine enforces this: x264 and FFmpeg's own AAC encoder cannot be selected in a shipped build. MP3 uses LAME, whose patents have expired.

This decision lets the FFmpeg build VerseCut ships be a plain LGPL one with no GPL components in it, run as a separate program, and it means the encoding is performed by components Microsoft, Apple and the GPU vendors provide for the purpose. It is the customary route for a small vendor, and it has the useful side effect of making long exports faster on hardware that has an encoder. It is a design choice, not a legal opinion; Stateam's counsel advises on patent questions.

So that a particular FFmpeg build can always be identified, each release lists the SHA-256 of every video-engine file in setup/ffmpeg/SHA256SUMS (setup/ffmpeg-macos/SHA256SUMS on macOS), and the installer refuses a file that does not match.

6.3 Distribution and code signing

Direct download from the product's own website is the primary channel. It keeps the great majority of the revenue with the publisher and, more to the point, it is the channel that permits an offline licence file rather than a store account.

VerseCut 3.0 is distributed as ZIP downloads with script installers that are not yet code-signed, so Windows SmartScreen and macOS Gatekeeper ask the user to confirm the first run; the installation guide shows how. A signed Windows installer and a Developer ID-signed, Apple-notarised macOS installer are the next distribution step. On macOS the installer builds one self-contained VerseCut.app in Applications. The Microsoft Store would be a secondary channel once a single

signed offline installer exists: non-game applications may use their own payment mechanism, which keeps the licensing model intact. Google Play cannot carry a desktop editor at all — it distributes Android applications only — so it is not a channel for this product in any form. The Mac App Store is a later question at best: its sandboxing and update rules, and its requirement that purchases run through Apple, are difficult to reconcile with an offline licence file.

6.4 Privacy position

Because VerseCut collects nothing, its privacy position is short. There is no analytics, no telemetry, no crash reporting and no cloud storage. No personal data is processed by the application. The only information that leaves a customer's control is what they give to the payment provider at checkout, and the payment provider is the merchant of record for that transaction. An organisation subject to a data-protection regime is therefore assessing a piece of software that processes its material entirely on its own equipment — a materially easier assessment than the alternative, and one that does not change when the vendor revises a privacy policy.

7. Market position

VerseCut is not the cheapest editor available and does not claim to be. It is priced within the consumer range while removing the three things this paper has argued are costs rather than features.

Table 6. Category pricing and terms, as observed in September 2026

Product	Price (US list)	Account required	Online check	In-product promotion
VerseCut Creator	$59 a year, or $89 once	No	Never; the licence is verified on the computer	None
Wondershare Filmora	$49.99–$59.99 a year, or $79.99 perpetual for one platform	Yes, a Wondershare ID	Activation online	Vendor support states its prompts cannot be completely disabled
CapCut Pro	About $179.99 a year	Yes, a CapCut or TikTok account	Sign-in, and paid features are checked online	Upsells throughout the free version
Adobe Premiere Pro	$22.99 a month on an annual plan	Yes, an Adobe ID	Licence check every 30 days	Not applicable

Table 6: prices and terms were observed in September 2026 from each vendor's own published pricing and support pages, and should be re-checked before they are relied on. Other product names are the trademarks of their owners and appear here only for factual comparison.

Three observations follow from the table. VerseCut Creator at $59 a year sits at the low end of the category, at roughly a quarter of the annual cost of a Premiere Pro plan and a third of CapCut Pro, while being the only entry in the group with no account, no online check and no promotional content. The $89 one-time licence is the only way in this group to stop paying and keep working — Filmora's perpetual option is close in price but is limited to one platform and still requires a vendor account to activate, while CapCut and Premiere Pro have no one-time option at all. And the comparison cuts

both ways: VerseCut does not offer cloud collaboration or generative effects, and an organisation that needs those should buy a product that has them.

The intended customer is therefore specific rather than general. It is the organisation that produces video every week, whose requirement is that the tool be predictable, that the footage stay on the premises and that the cost be known in advance. For that organisation the comparison is not feature-count against feature-count; it is whether the tool it installs will behave the same way next month as it does today.

8. Where VerseCut goes next

The near-term roadmap is ordinary product work: template packs for titles and overlays, batch export, and seat administration for Studio customers. The more interesting direction is that the architecture is transferable, and the constraint it was built for is not unique to ministry media.

VerseCut solves a general problem — editing and processing video on a machine that must not send that video anywhere — that also arises wherever footage cannot leave the building.

- **Clinical and imaging workflows.** Procedural video, ultrasound and endoscopy recordings, patient-education material and teaching clips are routinely trimmed, captioned and annotated. The work is small, the confidentiality requirement is absolute, and a tool that both edits locally and reads and writes DICOM would fit a workflow currently served by general-purpose editors that were never designed for it.
- **Secure field and forward-base video.** Footage captured where connectivity is intermittent, monitored or forbidden must be cut and reviewed on the machine that holds it. An editor with no account, no telemetry, no background service and no update channel is the shape of tool such an environment can actually accredit.
- **Laboratory imaging.** Microscopy and instrument video, time-lapse and experimental capture are prepared for publication and teaching on machines that are frequently isolated by policy, and often by design.
- **Regulated corporate environments.** Legal, financial and industrial organisations produce internal video that cannot be processed on a third party's infrastructure, and are the readers most likely to want the architecture documented and the source readable before anything is installed.

None of this is shipping capability, and this paper does not present it as such. VerseCut 3.0 is a video editor for creators and small production teams. What the current product establishes is that the hard parts are solved: local-only processing with no network dependency, offline licensing with no activation server, a documented and auditable security model, deterministic output, and files in formats the customer owns. Those are the components each of the directions above requires, and they exist today in a product people use to hit a weekly deadline. That is the proof; the rest is direction of travel.

9. Summary and a checklist for evaluators

The consumer subscription editor has become an operational and security liability for organisations that produce video regularly. Mandatory accounts, background updater services, updates that arrive unasked, and advertising inside a working tool are each defensible in the consumer market they were designed for, and each imposes a cost that market never has to account for: disruption at the vendor's timing, privileged code running permanently on a machine that holds sensitive material, a

third party standing between the organisation and its own archive, and promotional content displayed on the organisation's premises in its name.

A local-only editor removes those liabilities without giving up capability. VerseCut 3.0 edits multi-track video, transcribes speech on the machine's own processor, places timed overlays automatically, mixes and normalises audio to platform standards and exports to 4K — with no account, no telemetry, no background service and no mechanism by which it could update itself. Its licence is a signed file verified on the customer's own computer, so activation needs no server and no connection, and no installed copy can be disabled remotely. Its limits are stated in section 4.6 rather than hidden.

The claims in this paper are deliberately the kind that can be checked. An evaluator does not have to take any of them on trust.

What to verify during a trial

1. **Install, then disconnect the network.** Complete setup, disconnect the machine from the network entirely, and then perform a full session: import media, edit, transcribe, and export. Everything should work. Setup is the only step that needs a connection.
2. **Confirm there are no outbound connections while editing.** Reconnect the machine and watch it with a network monitor or a host firewall in prompting mode through a complete editing session and export. There should be no outbound traffic from VerseCut at all.
3. **Confirm nothing is left running afterwards.** Close the editor window, wait two minutes, and inspect the process list, the service list, scheduled tasks and login items. The engine should have exited by itself and installed nothing that persists.
4. **Confirm an export repeats exactly.** Export the same project twice with identical settings on the same machine and compare the two files byte for byte. They should be identical.
5. **Confirm a licence activates offline.** With the machine disconnected, paste a licence into the Licence panel. It should activate immediately, because the signature is verified locally against a key already in the application.
6. **Read the source of the parts you care about.** The local server and its security model are in app/server.py; licence verification is in app/engine/licensing.py and app/engine/ed25519.py; the render pipeline is in app/engine/render.py. None of it requires decompilation.
7. **Confirm your files stay yours.** Open a saved project in a text editor — it is JSON. Open an exported file in any player. Let a trial licence lapse deliberately and confirm that the projects still open and still edit on the Free plan.

If any of those checks fails, the argument of this paper fails with it. That is the intended standard.

Appendix A. Contact

Item	Detail
Publisher	Stateam LLC, Princeton, New Jersey, United States — gowordtoday productions
Product	VerseCut 3.0 — an offline video editor for creators
Contact	Ben John — capture@stateam.net

Item	Detail
Evaluation	A 14-day Creator trial starts on first launch. No account, no card, nothing to cancel
Platforms	Windows 10 and 11 (64-bit); macOS 12 Monterey or later (Apple silicon and Intel)
Source and audit	The engine ships as readable source. The FFmpeg source and the SHA-256 of every engine file are published with each release
Nonprofit and ministry	30% off any plan for a verified 501(c)(3) or the equivalent