

PLOTVA STUDIO PRO — Source Seal

Build v39.6-r9 · 28 October 2026

Document	PLOTVA-SOURCE-SEAL
Subject	Encryption of the application source in every shipped build
Applies to	the site copy, the free viewer, the sealed sale file and both Windows executables
Status	Implemented, tested, and in the current release

CONTENTS

1	What was done	1
2	How it works	2
3	What this protects against	2
4	What this does not protect against	.
5	Build and verification	3
6	Rebuilding after a code change	4
7	Fingerprints of the current release	4

1. What was done

Every inline script in the product is now stored **encrypted** rather than as readable JavaScript. The build takes each script in turn, compresses it (LZ77), encrypts the result (ChaCha20, fresh key and nonce per build), and records a SHA-256 of the compressed data. At run time a small bootloader decrypts each block, checks the hash, expands it, and hands it to the browser in exactly the position the original script occupied.

The effect: if you open any shipped file in a text editor you see the page furniture, the styles, and about one and a half megabytes of base64 ciphertext. You do not see the program. "View source" and "Save page as" produce the same ciphertext.

Build	Before	After	Change
<code>pages-site/app.html</code> (site copy)	2,904,304 B	1,793,966 B	–38 %, sealed
<code>PLOTVA-STUDIO-PRO.html</code> (sale file)	3,884,825 B	2,404,373 B	–38 %, sealed + password vault
<code>PLOTVA-STUDIO-PRO-DEMO.html</code> (viewer)	2,912,819 B	1,800,602 B	–38 %, sealed

Build	Before	After	Change
PLOTVA-STUDIO-PRO.exe	3,926,528 B	2,446,336 B	−38 %
PLOTVA-STUDIO-PRO-DEMO.exe	2,955,264 B	1,843,200 B	−38 %
PLOTVA-CLOUDFLARE-PAGES-UPLOAD.zip	1,120,784 B	1,120,784 B	site bundle, sealed app

The files got **smaller**, because compression runs before encryption and more than pays for the base64 expansion (2.44 MB of source becomes 0.99 MB of compressed blocks).

2. How it works

```
`` build time run time (browser) ----- script source bootloader runs
first | LZ77 compress base64 decode | SHA-256 of compressed bytes -----> decrypt
(ChaCha20, counter 1) | ChaCha20 encrypt SHA-256 check — mismatch —> refuse, log, block
stays closed | base64 LZ77 expand ▼ execute in place, synchronously
<script>>window.__PLOTVA_UNSEAL__(i,...)</script> ``
```

Each block keeps the `id` the original script had, so anything that looks a script up by name still finds it. Execution is synchronous and in document order, which is why nothing inside the program had to change: the navigation, the ribbon, the licence gate, the export engine and the drawn output are byte-for-byte the same code, simply not stored in the clear.

3. What this protects against

Exposure	Position now
Someone opens the file in a text editor and reads the program	They see ciphertext; the program is not readable
"Save page as" or a copy of the file circulates	What circulates is ciphertext
A search for a function name, a string, an API key or the licence logic	Returns nothing; the strings are inside encrypted blocks
The file is edited to remove a check ("crack it in Notepad")	The block's SHA-256 no longer matches and the application refuses to start
Casual inspection on a shared or borrowed machine	Only the bootloader and the page markup are visible

The integrity check matters as much as the secrecy: a modified build does not run at all, rather than running with a disabled licence check.

4. What this does not protect against

This section is deliberately blunt. Do not promise a customer more than this.

1. **The key is in the file.** It has to be — the browser has to decrypt the program without asking anyone. A person with the right skills can read the bootloader, recover the key, and decrypt the blocks. Encryption here removes the trivial path, not every path.
1. **A running browser holds the plain source.** Once the page has started, the expanded code exists in memory and in the DOM. Developer tools, a breakpoint, or a script that dumps `document.scripts` recovers it. **No HTML application can prevent this.**
1. **Screenshots and screen recordings are unaffected.** Nothing in a browser file can stop them; the earlier shortcut and right-click suppression remains a deterrent only.
1. **It is not a licence.** Access control is, and remains, the licence server: a copy without a valid machine-bound key stays in viewer mode no matter how the source is stored.
1. **It changes nothing for the owner.** The plain master (`_source/naksha-master-private.html`) is never shipped and remains the only readable copy. Keep it private; it is the key to everything.

5. Build and verification

```
`` bash python3 _source/plotva_crypto.py # ChaCha20 known-answer test (RFC 8439) python3
_source/build-all.py # seal → vault → demo → exes → site → packs python3 _source/build-all.py
seal # one stage only ``
```

The seal step prints, for every build, how many scripts were sealed, the compression ratio and the new file size, and it round-trip checks each block before writing anything.

Verified on the delivered files (all executed, none assumed):

Check	Method	Result
ChaCha20 correct	RFC 8439 §2.4.2 vector, Python + JavaScript + the C code in the executable	pass
SHA-256 correct, including the padding edge	7 message lengths plus 123,456 random bytes against Node's <code>crypto</code>	pass
LZ77 round-trip	decompress every block and compare with the original script	identical
All blocks unseal in a real browser	Chromium on the shipped site copy	21 / 21 blocks, 0 failures, no page errors

Check	Method	Result
The program still runs after sealing	<code>_qa/full_sweep.py</code> on the sealed password build, against the local licence service	56 / 56 pass , 16 real downloads
Password gate unaffected	<code>_qa/sealed_v396.py</code>	locked → no app; wrong password refused; correct password boots
Entitlements unaffected	<code>_qa/daytrial_e2e.py</code>	1 day = design only; 3 days = + save; no downloads
Ribbon unaffected	<code>_qa/tiercheck.py</code>	9 / 9, no overlaps
No plain source in any shipped build	<code>_qa/verify_ten.py</code> string checks	clean for site copy, viewer and sale file
Executable still decrypts and verifies	delivered <code>.exe</code> run with <code>--selftest</code>	<code>bytes=2404373</code> , <code>sha256=23043213313b10ad...</code> , matches the sealed build

One defect was found and fixed while testing this work: the SHA-256 padding loop in the bootloader shifted a 32-bit word by 32 bits, which JavaScript treats as a shift by zero, so blocks longer than three bytes failed their integrity check. It was caught by the test vector and is corrected; the vector suite now covers seven lengths plus random binary data.

6. Rebuilding after a code change

1. Edit `_source/naksha-master-private.html` — the only file a human edits.
2. `python3 _source/build-all.py`
3. Re-run the suites: `_qa/verify_ten.py` , `QA_SEALED=file:///tmp/PLOTVA-local.html`
`_qa/full_sweep.py` .
4. Record the new hashes in `builds/REGISTER.csv` (the build prints them).
5. Re-upload the site zip and re-paste the Worker file if either changed.

Every build produces a new key, so **a rebuilt file never has the same hash as the previous one**. Rebuild once, then publish; do not mix files from two builds in one delivery.

7. Fingerprints of the current release

File	Bytes	SHA-256
<code>PLOTVA-STUDIO-PRO.html</code> — sale file (sealed source + password vault)	2,404,373	<code>23043213313b10ad2a68a493a5276957f3a645a5a9762d0a130f2c339f0001d8</code>

File	Bytes	SHA-256
<code>PLOTVA-STUDIO-PRO-DEMO.html</code> — free viewer	1,800,602	<code>c0c685781e75f6171c0253f886e0aba073b94d51e6ef181d4923c55366dae1fa</code>
<code>pages-site/app.html</code> — site copy	1,793,966	<code>bc004074bce1858f12120e37c3a926898ba395f5b5aa5cc60e9d36681ef6d122</code>
<code>PLOTVA-STUDIO-PRO.exe</code>	2,446,336	<code>9f922569a58e410759fe6abed64f8c27d79ce6d3d6dc62a369cf85fe46970247</code>
<code>PLOTVA-STUDIO-PRO-DEMO.exe</code>	1,843,200	<code>de2bd329fb028e4b4f9d2da6947d95de723d5a09eb13ca174c4bad4412739f45</code>
<code>PLOTVA-CLOUDFLARE-PAGES-UPLOAD.zip</code>	1,120,784	<code>cc64c9250b2bc337f1047335a620555432937c7cf6f8925eeda83de17ece3c73</code>

End of document. Section 4 is the part to read aloud to a customer who asks whether the program can be copied; section 5 is the part that proves the work was tested.