Amulet Wallet Generator

Release Notes v1.63

22 August 2026

Contents

Amulet — Release Notes

v1.63 · 22 August 2026

SHA-256 `69de86d4c8a8d14abbdc5d93f70af6295ee8ceb6268ad6cdc172c9af17a44a92`

What this release is

v1.58 was audited end to end: the raw randomness, the seed pipeline, and every key derived from it. The cryptography held. **235 of 235 derivation assertions matched independent reference implementations exactly** — official BIP-39, BIP-84, BIP-86 and BIP-47 vectors among them — and the entropy passed a NIST SP 800-22 style battery at a rate statistically indistinguishable from `os.urandom`.

What did not hold was the **accounting**. In several places the interface described entropy the tool had not actually collected. v1.59 through v1.62 fix every one of those, plus the two sources whose entropy could not be trusted at all.

A theme runs through F-01, F-03 and F-04: in each case the code did the right thing and the interface **described it wrongly** — crediting a CSPRNG that was skipped, a card shuffle that never happened, an air gap that was never measured. Cryptographic tools are judged on their primitives. Users act on the labels. The labels were the defect.

Do I need to do anything about wallets I already made?

Almost certainly not. Read on only if one of these applies to you.

If you made a wallet…	Then
normally, with any version	**No action.** The 256-bit CSPRNG was always the dominant source. Every wallet made this way is sound.
in **Reproducible mode**, with **fewer than 99 dice rolls** (24-word) or **50** (12-word)	**Move your funds.** See F-01.

If you made a wallet…	Then
in **Reproducible mode** with the full roll count	No action. That wallet always had its full strength.
using **keystrokes** as the entropy source	No action for the wallet's security — the CSPRNG protected it. But the extra protection you thought you were adding may not have been there. See F-07.
using **cards** without reshuffling between rounds	No action. Even one shuffled deck carries up to 225.6 bits. See F-03.

Nothing in this release changes how seeds are derived. **Every wallet ever produced by this tool still reproduces byte-for-byte**, verified across versions. The `combined string` format is unchanged, including the now-unused `keys:` and `kt:` fields.

Findings

F-01 · Reproducible mode was scored as if the CSPRNG were present

Severity: High · Fixed in v1.59

Reproducible mode deliberately skips `crypto.getRandomValues` so the same draws always regenerate the same wallet. The entropy scorer was never told this. Every branch of it assumed a 256-bit CSPRNG contribution that did not exist.

Driving the real interface — 24-word wallet, Reproducible mode, 20 dice rolls, confirmation accepted:

```
reproducible : true
csprngHex    : ""                          ← no CSPRNG entropy at all
diceStr      : "12345612345612345612"      ← 20 rolls = 51.7 bits
scoreLabel   : "Very Good"                 ← 3 of 4
note         : "CSPRNG + some user entropy. Cryptographically strong…"
```

51.7 bits is brute-forceable. On 12-word wallets the mislabelling reached **"Excellent"** at 26 rolls (67.2 bits). Even the "Weak" branch asserted that "the CSPRNG still protects the seed" — false, at exactly the moment a user most needed an accurate statement.

Fixed: reproducible wallets are now scored on their own entropy alone, against a hard 128-bit floor, and every note states plainly that no CSPRNG entropy was used.

If this affected you: a reproducible wallet built from fewer than the full roll count has materially less security than its word count implies. Generate a new wallet and move your funds.

F-02 · The low-roll guard was a dismissible dialog

Severity: Medium · Fixed in v1.59

The dice commit button accepted a single roll, and reproducible mode's roll-count check was a `confirm()` the user could dismiss — softened deliberately in v1.54, per the code comment. Combined with F-01, the floor for a reproducible wallet was one die roll: about 2.6 bits.

Fixed: a hard gate. Reproducible mode requires 99 D6 rolls for a 24-word seed and 50 for a 12-word one, with no way past it. Outside reproducible mode nothing changed at the time — the CSPRNG covered any roll count — though v1.61 later introduced a floor there too.

F-03 · Card entropy credited a reshuffle that was never verified

Severity: Medium · Fixed in v1.60

The scorer credited $\log_2(52 \cdot 51 \cdot 50 \cdot 49) \approx 22.4$ bits per round, which holds only if every round is dealt from a full, freshly shuffled deck. A user who shuffled once and dealt four cards at a time got substantially less, and nothing noticed.

Fixed: the assumption is now tested. Within any run of 13 rounds (52 cards), dealing off one un-reshuffled deck yields each card at most once, while genuine per-round reshuffling yields about 18 duplicates. Amulet takes the leanest such window — the count across a full 23-round session is uninformative, since 92 cards drawn from 52 must repeat under any scheme.

Calibrated over 100,000 simulated honest sessions and 4,000 lazy ones:

Dealing pattern	Leanest window duplicates	Trigger	Detection
Reshuffled every round · 11 rounds	≥ 5	4.5	0 false positives
Reshuffled every round · 23 rounds	≥ 8	6.1	0 false positives
One shuffle, dealt straight through	0	—	100%
Same deck order dealt twice	0	—	100%

When the reshuffle is not corroborated, entropy is re-credited on the single-deck model and the report says so. **The corrected figures are still strong** — one shuffled deck carries up to $\log_2(52!) = 225.6$ bits — so a missed reshuffle is reported as an observation, not a defect. It escalates to a flag only if the corrected number falls under 128 bits.

F-04 · `navigator.onLine` was reported to the user as "safe"

Severity: Medium · Fixed in v1.60

The status pill read **"Offline · safe"**, derived entirely from `navigator.onLine` — which reports network-interface state, not reachability. A VM with a virtual adapter, a captive portal, or a LAN with no route can all report misleadingly. The zero-network implementation is genuinely clean; the overstatement was the word *safe*, on a page whose entire threat model is not being observed.

Fixed: the pill now reads **"No network detected"**, its hint reads *"A hint, not proof — verify yourself"*, and its tooltip names `navigator.onLine` and states outright that it is not proof of an air gap. The check itself is unchanged by design: probing the network to prove you are off it would defeat the purpose.

F-05 · The plaintext report is seed-equivalent

Severity: Note · Fixed in v1.60

The exported `.txt` and `.pdf` print the full combined input string, including the raw CSPRNG hex. Anyone holding either file can recompute the seed **without ever seeing the mnemonic** — so redacting the word list would not make it safe. This is intentional, and it is what makes the verify lane work, but the interface did not say so.

Fixed: both exports now open with a banner stating the file is equivalent to the seed phrase and rebuilds the wallet two independent ways. The plain-text export tile is labelled seed-equivalent, and the on-screen verify panel carries a shoulder-surfing warning above the line it prints.

If you have old exports: treat every `.txt` and `.pdf` Amulet has ever produced exactly as you would treat the seed phrase itself. Offline only. The encrypted `.enc` file is the one to keep on a normal disk.

F-06 · No way to verify the HTML file itself

Severity: Medium · Fixed in v1.60

There was no published checksum, no signature, and no in-page guidance for verifying the download. Every guarantee in the audit sits downstream of trusting the bytes the server returned. For an offline wallet generator that is structurally the largest practical risk — larger than anything in the code.

Fixed: a *"Verify this file before you trust it"* panel now sits at the top of the first screen. Drop the `.html` onto it and it computes the SHA-256 of the real bytes on disk, with a field for a checksum obtained elsewhere. A page cannot hash itself — `fetch()` on `file://` is blocked and the serialised DOM is not the file — so the user supplies it. The panel also prints the `sha256sum`, `shasum`, `certutil` and `Get-FileHash` equivalents, because computing the digest outside the page is the better habit.

The panel is careful about what it claims: a match proves the file is unmodified, not that it is trustworthy. See `CHECKSUMS.txt`.

F-07 · Keystroke entropy could not be trusted

Severity: Medium · Source removed in v1.62

Two defects, one fixable and one not.

Fixable — the timing channel could be a literal constant. Inter-key timing packed `Math.floor(dt*1000) & 0x7` and credited a flat 2 bits per interval. On any browser that coarsens `performance.now()` to 1 ms — Firefox with `resistFingerprinting`, Tor Browser, i.e. precisely this audience — `dt` is a whole number of milliseconds, `dt*1000` is a multiple of 1000, and `1000 & 7 == 0`. The field was the constant zero for every keystroke. Measured payload for 65 characters: forty hex digits of `0000000000…`, credited **128 bits**.

Not fixable — the content estimators cannot see what a user already knew. Shannon, bigram, Rényi-2 and LZ all read the observed distribution of the text itself. Measured on this build:

Typed input	Credited	Flags
honest keyboard mash, 67 chars	251 bits	0
Bitcoin genesis headline, 69 chars	**334 bits**	0
a famous quotation, 68 chars	**311 bits**	0
the user's own email address ×3, 59 chars	**240 bits**	0

Memorable text **outscores genuine mashing**. Language detection would catch the quotation and the email address, but never a memorised password — and that is the case that costs someone their coins.

Removed. Dice and cards work because a physical randomiser makes the entropy and the human merely reads it off. The chaos grid works because involuntary motor noise does. Keystrokes asked the user to supply unpredictability from memory, which is the one place it is not.

If you used it: your wallet was still protected by the 256-bit CSPRNG, and reproducible mode never accepted keystrokes. No action is required. But the defence-in-depth you believed you were adding may not have been there.

Findings (continued)

F-08 · The chaos grid could strand the player, and said "target reached" when it had not been

Severity: Medium (usability, not key security) · Fixed in v1.63

Two separate bugs that met in the middle and produced a dead end.

The board ran out of tiles. Revealing a symbol twice shatters every remaining tile carrying it, so a 25-tile board burns down in the low twenties of clicks. Continuing past that depended on happening to flip one of the three restore tiles. Driving the real board 300 times:

	v1.62	v1.63
sessions that reached the 128-bit floor	65.0%	**100%**
sessions that ran out of clickable tiles	**39.0%**	**0%**
clicks when the board died	median 22 (110 bits)	—
clicks to reach the floor	median 26	median 26

A player in that 39% was left below the floor with nothing left to click and no way forward but *Start over*.

The shimmer lied. The "entropy target reached — keep going if you like" caption fired at `GRID_ENOUGH_BITS = 80`, while committing required 128. So the grid announced success at 110 bits while the Commit button stayed disabled — which is exactly the state that was reported:

```
22 clicks · ~110.0 bits · cells 91.2 · timing 63.0 · pos 42.5
110.0 / 128 bits — about 4 more clicks to go
                                    …with no tiles left to click
```

Fixed:

- When fewer than four tiles remain, a fresh board is dealt automatically. The click log and every credited bit carry over, and the page says so. Effort is unchanged at a median of 26 clicks — the refill removes the dead end without making anyone click more.
- The shimmer now fires on exactly the same condition as the Commit button, so the two cannot disagree. Its caption reads *"enough entropy — you can commit now, or keep flipping"*.
- The progress bar tracks credited bits toward 128 rather than raw clicks. It previously read 73% at 110 of 128 bits, and could reach 100% while Commit was still disabled.

Verified over 40 full sessions checking the invariant after **every single click**: zero disagreements between shimmer and button, zero dead ends, bar never above 100%. Quality gating is untouched — 15 clicks is still short of the floor, and metronome play is still refused at any length.

Other changes

v1.61 · A 128-bit floor on all user-supplied entropy

User entropy matters in exactly one scenario: the CSPRNG is backdoored or broken. In that scenario the attacker knows the `csprng:` field and brute-forces the rest, so the seed's surviving security is **exactly** the bits the user's draws contributed. A source delivering 26 bits is not a little defence in depth — it is theatre that costs real effort and buys nothing.

No source can now be committed below **128 bits**: the size of a 12-word BIP-39 seed, and the same number that gates reproducible mode.

Source	Was	Now	At the floor
Dice	1 roll (2.6 bits)	50 rolls	129.2 bits
Cards	11 / 23 rounds	≥ 6 rounds	135.8 bits
Chaos grid	10 clicks (50 bits)	~30 clicks	128.0 bits

The gate reads **credited** bits, not raw counts, so patterned input costs more: 60 identical dice rolls is 155 nominal bits and is refused outright.

Partial input no longer produces a “continue with CSPRNG only?” dialog — those either added entropy too weak to matter or silently discarded the user’s work. Generation now stops and states the shortfall. **Entering nothing at all remains perfectly valid**, and CSPRNG-only is still the strongest default.

v1.62 · Chaos grid scoring repaired

Three bugs, all of which rejected honest players:

- **The `distinctCells` penalty was a false positive.** A tile can only reappear in the log *after* a restore tile legitimately regenerates the board, where `log₂(liveBefore)` has already priced the choice correctly. Worse, past 32 clicks the threshold `ceil(0.8n)` exceeded the 25 cells that exist, so the 0.5× penalty fired for **every possible player** — punishing exactly the extended play needed to reach the floor. Deleted.
- **Cadence and aim flags were counted twice** — once as a graded penalty, again as a hard block. An ordinary 600 ms ±120 ms clicker was rejected 58% of the time. Now advisory; the penalty already prices them. Metronome-exact timing still blocks, as it is not humanly achievable.
- **Aim thresholds were absolute pixels** (4 and 10) against tiles that are ~70 px on desktop and ~55 px on a phone. Users were being penalised for having a small screen. Now scaled by tile size.

Play profile, 30 clicks	v1.61	v1.62
relaxed clicker, 600 ms	rejected 58% of the time	accepted
fast steady, 180 ms	rejected	accepted
accurate aim, 2.5 px	rejected	accepted
phone, 55 px tiles	rejected	accepted
metronome + dead centre	rejected	rejected

What was verified, and what was not

Verified by test:

- 235/235 derivation assertions against `bitcoinjs-lib`, `bip39` and `bip32`, across 3 mnemonics × 3 passphrases (including Unicode with an emoji, to exercise NFKD), all four address types, SLIP-132 serialisation, and published BIP-39 / BIP-84 / BIP-86 / BIP-47 vectors.
- Wordlist byte-identical to the official BIP-39 English list.
- Entropy indistinguishable from `os.urandom` under the same battery (1.159% vs 0.873% of p-values below α=0.01, Fisher exact p = 0.13).
- Zero network requests at runtime, with all non-`file://` requests intercepted.
- Encrypted export: AES-256-GCM, PBKDF2-HMAC-SHA256 at 600,000 iterations, 16-byte random salt, 12-byte IV, header bound as AAD.
- Cross-version chain identity: v1.61 and v1.62 produce byte-identical entropy for the same inputs, including legacy keystroke strings.

Assumed, not verified — you should know these:

- **Card entropy** assumes a genuine shuffle. v1.60 detects the *absence* of a reshuffle between rounds but cannot judge shuffle quality within one.
- **Chaos grid entropy** assumes human cell choice is close to uniform. Humans are imperfect randomisers, and a lag-1 adjacency test on a session this short lacks the power to check it — a 2× adjacency bias is caught only 41% of the time. The global 128-bit cap against a raw estimate of ~270 bits at 30 clicks builds in a >2× haircut, but the discount is assumed from the literature, not measured from your session. This is a genuine step down from `log₂(6)` per die roll, which is arithmetic you can check yourself.
- **Grid timing and position entropy** assumes the machine is not observing you. On a compromised machine, click timing is recoverable and that credit evaporates.

If you want entropy you can fully verify yourself: use dice. 50 rolls, $50 \times \log_2(6) = 129.2$ bits, and you can do the arithmetic on paper.

Version history

Version	Change
v1.63	Chaos grid: automatic board refill (F-08). Shimmer and progress bar now track the real commit condition.
v1.62	Keystroke source removed (F-07). Chaos grid scoring repaired. Dead CSS removed.
v1.61	128-bit floor on all user-supplied entropy. Weak-input dialogs replaced with hard stops.
v1.60	Card reshuffle verification (F-03). Network wording corrected (F-04). Seed-equivalence warnings (F-05). File integrity panel (F-06).
v1.59	Reproducible-mode entropy scoring corrected (F-01). Roll-count guard made a hard gate (F-02).
v1.58	Audited baseline.