SECURITYNUT

Burn

Technical Manual

One-time read notes, encrypted in the browser and deleted from the server the moment they are opened.

Item	Value
Document reference	SN-TM-004
Version	1.0
Status	Public
Issue date	11 September 2026
Author	MP
Subject	Burn application
Software version documented	v1.0.6
Classification	Public technical documentation

Contents

1. Summary 3
2. Scope and terminology 3
3. User flow 4
3.1 Creating a note 4
3.2 Reading a note 5
3.3 Two limits 5
3.4 Privacy page 6
4. Cryptographic design 6
4.1 What the server can and cannot do 6
4.2 What Burn does not protect against 7
5. Server behaviour 7
5.1 Routes 7
5.2 The atomic read 8
5.3 Abuse prevention 8
6. Data model 8
7. Interface and design 9
8. Service integration 9
9. Change history 10
10. Screenshot 10

1. Summary

Burn is a one-time read note service. A sender types a note, the browser encrypts it, and the sender receives a link. The reader opens the link, presses Decrypt, and the note is shown for twenty seconds before it is cleared and can never be shown again. The server stores only ciphertext, the decryption key travels in the URL fragment which browsers never transmit, and the stored row is deleted atomically at the moment of the first read.

Burn is the fifth public tool in the SecurityNut suite alongside the credential generator, Crypto, Notes and IP. It was deliberately built as its own site rather than folded into Crypto so that Crypto's promise of never touching the network remains intact. Burn is the one SecurityNut tool that must talk to a server, and this manual is careful about exactly what that server sees.

Two limits govern every note. The unread lifetime is one hour by default and can be extended to one day or one week, after which an hourly sweep removes it. The read lifetime is twenty seconds of screen time. Whichever is reached first wins, and the database never holds a note that has already been opened.

Design position. The twenty second rule is theatre about screen time, not a security control. A reader can screenshot the note, and that is accepted. What Burn guarantees is narrower and stronger: the operator cannot read the note, the database cannot leak it in the clear, and once one person has read it nobody else can.

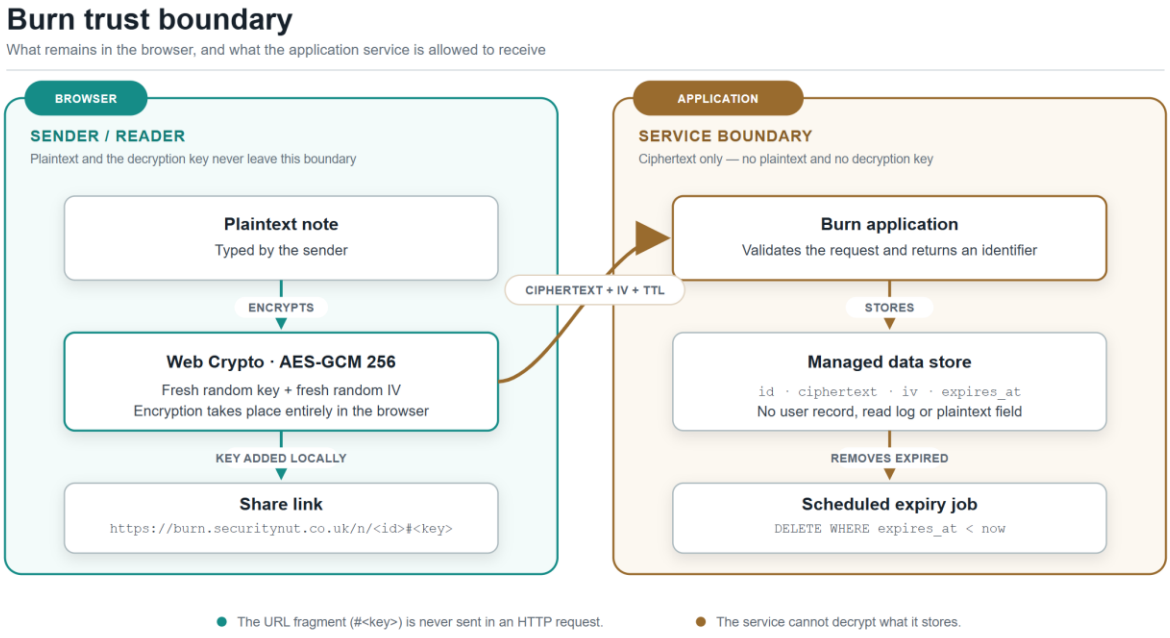

Figure 1. Trust boundary. Plaintext and key stay in the browser; the application service holds ciphertext only.

2. Scope and terminology

This manual describes the Burn service as deployed on 10 September 2026. It covers the user flow, cryptographic design, server behaviour, data model, operational controls and open items. It does not restate the general SecurityNut estate conventions, which are maintained separately.

In keeping with the other SecurityNut manuals, hosting infrastructure is described generically. The platform that runs Burn is referred to as the Service Provider. Its components are named by function: the application service, the managed data store, the scheduled expiry job, the administration console and the invisible

human verification challenge used when a note is created. Browser standards keep their names: Web Crypto, AES-GCM and the URL fragment.

Term	Meaning in this document
Note	A single message created by a sender for one reader
Ciphertext	The encrypted note as stored, unreadable without the key
Key	A random 256-bit AES-GCM key generated in the sender's browser and carried in the link fragment
IV	The 96-bit initialisation vector generated per note and stored alongside the ciphertext
Fragment	The part of a URL after #, which browsers never send to any server
TTL	Time to live, the unread lifetime chosen by the sender
Sweep	The hourly scheduled job that deletes expired unread notes
Read	The single Decrypt action that deletes the row and reveals the note

3. User flow

3.1 Creating a note

The sender opens burn.securitynut.co.uk and types a note into the central panel. On wide screens (1100 px and above) the four numbered how-it-works steps float as panels either side of the note, steps one and two stacked to the left and three and four to the right, so the page reads as a guided instrument rather than a form. On narrower screens the steps stack above the note.

The sender chooses an unread lifetime of one hour (the default), one day or one week, and presses Copy link. The invisible verification challenge runs silently. The browser generates a fresh key and IV, encrypts the note, sends the ciphertext, IV and TTL to the application service, and receives a note identifier back. The link is assembled in the browser with the key appended as the fragment, and a WarGames-style resolve sweep runs across the note text so the encryption is visible as an event rather than a blink.

The sweep now starts before the clipboard write. An earlier build wrote to the clipboard first, and the browser's clipboard permission prompt blocked the sweep on first use, which read as a failure. The order was reversed in v1.0.2.

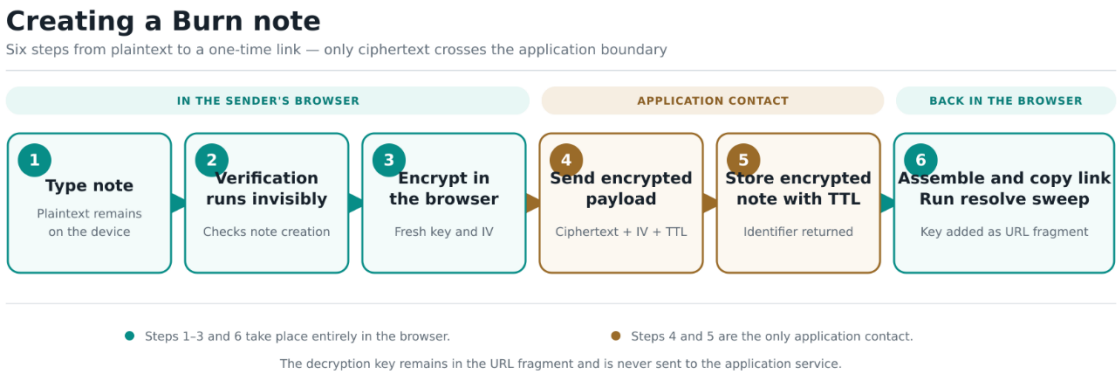

Figure 2. Note creation. Only steps four and five involve the server.

The copied text is not just the link. It carries a short self-destruct line stating that the note will be shown for twenty seconds and that the link expires after the chosen lifetime, so a recipient pasted the link in a chat understands the rules before they click.

3.2 Reading a note

The reader opens the link. The page shows that a note is waiting, shows the ciphertext rather than the plaintext, and offers a single Decrypt button. Nothing has been fetched yet beyond the page itself, and nothing is fetched until the reader acts. Link preview bots that fetch a URL to render a card therefore see a page with a button, not the note, and they cannot consume the one read because they never press it.

When Decrypt is pressed the browser asks the application service for the note. The application deletes the row and returns its contents in a single atomic statement. The browser then decrypts locally using the key from the fragment and shows the plaintext with a countdown reading Gone in 20s, 19s and so on to zero, at which point the text is cleared from the page. A Copy button is available during the twenty seconds. The note cannot be reloaded: the row no longer exists, so a refresh shows the not-found state.

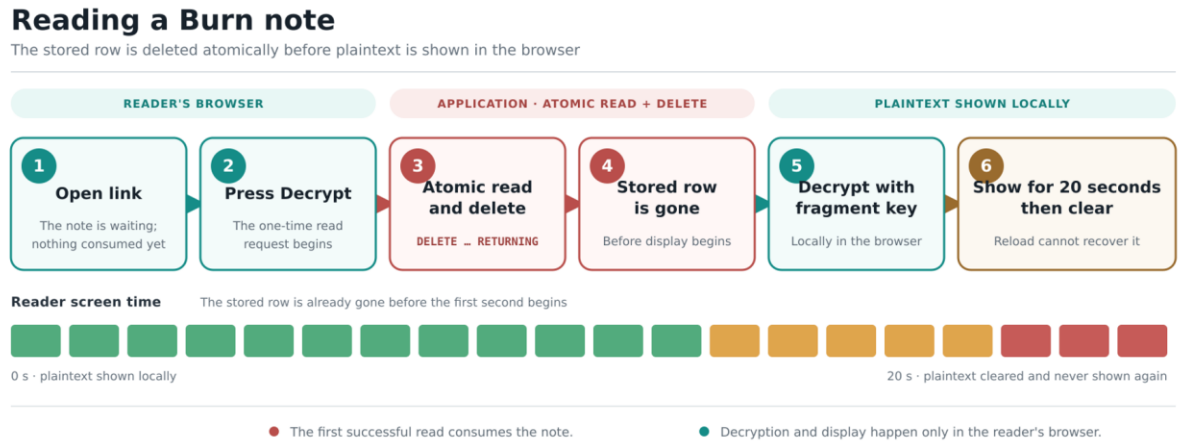

Figure 3. The read. The database row is gone before the first second of screen time begins.

3.3 Two limits

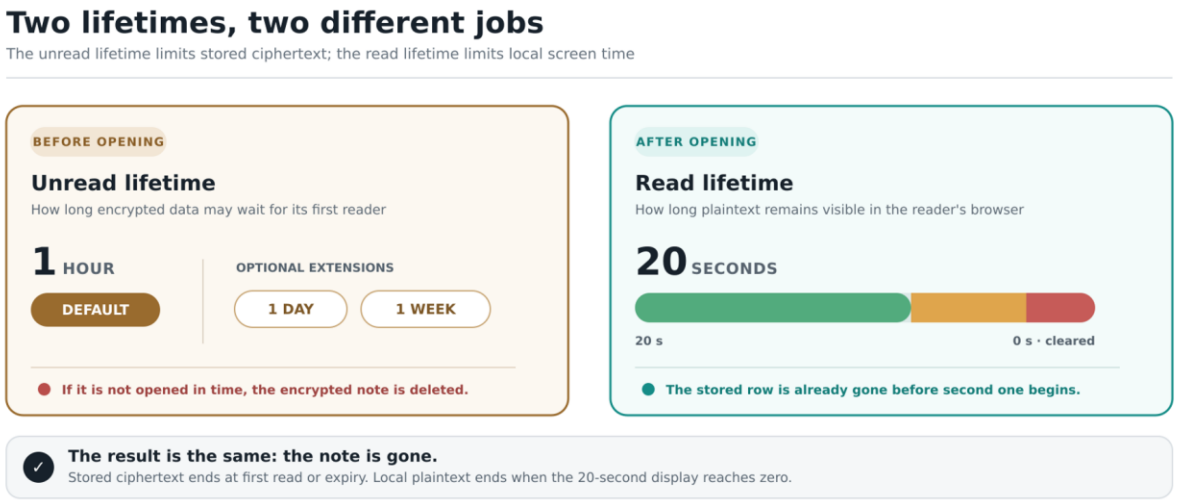

Figure 4. Unread lifetime versus read lifetime.

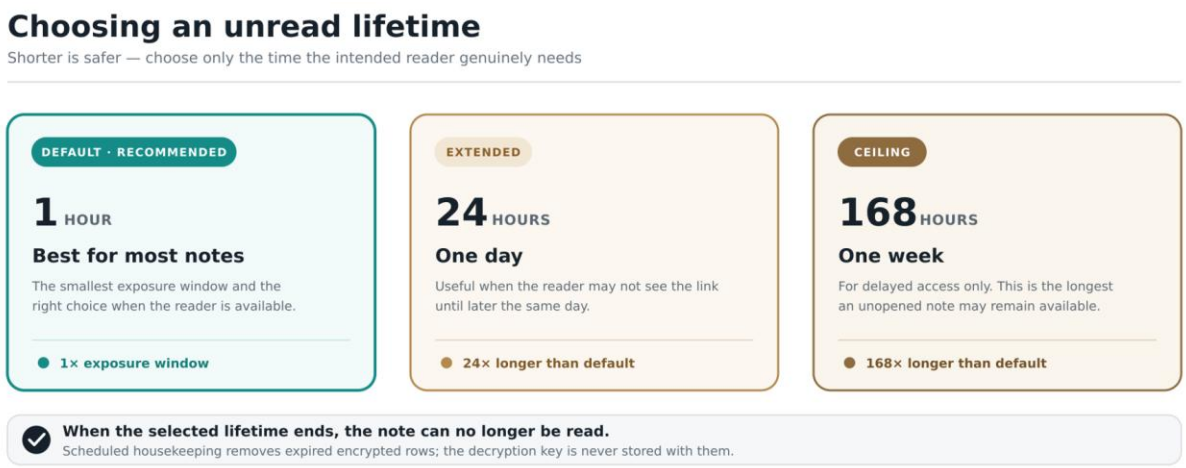

Figure 5. The three unread lifetime options. One hour is the default for security; one week is the ceiling.

3.4 Privacy page

The privacy page at /privacy is a live page in its own right and states the twenty second rule plainly, describes what is and is not stored, and explains that an automated verification service is used when a note is created. It is deliberately narrow in width so it reads as a document rather than a dashboard

4. Cryptographic design

Burn uses the browser's Web Crypto API and nothing else. There is no cryptographic code in the server-side application, which is the point: an implementation that cannot decrypt cannot be compelled or compromised into decrypting.

Element	Choice	Rationale
Cipher	AES-GCM, 256-bit key	Authenticated encryption; tampering with the ciphertext fails decryption rather than producing garbage
Key generation	crypto.getRandomValues, 32 bytes, per note	Fresh key per note; no key reuse, no key derivation, no passphrase to guess
IV	12 random bytes, per note	GCM standard length; stored with the ciphertext, safe to store because it is useless without the key
Key transport	URL fragment after #	Browsers do not send the fragment in requests, request logs or referrers
Encoding	Base64url for id and key	Copy-safe and link-safe; no symbol characters to break in messengers
Size limit	64 KB of ciphertext	Enough for any note; blocks abuse as a file drop

4.1 What the server can and cannot do

The application receives ciphertext, an IV and a TTL. It can count notes, time them and delete them. It cannot read them. If the managed data store were copied in full, the copy would contain ciphertext with no keys anywhere on the server side. Request logs may contain the note path but never the fragment. The one place the plaintext exists outside the two browsers is nowhere.

4.2 What Burn does not protect against

- A reader who screenshots or copies the note within the twenty seconds. Accepted; the Copy button exists for exactly this.
- A sender who pastes the full link somewhere an attacker can read it before the intended reader does. The first person to press Decrypt consumes the note; the intended reader then sees not found, which is itself a useful signal.
- Malware in either browser. Out of scope for any browser-based tool.
- A note that is never opened. It is deleted by the sweep at the end of its unread lifetime, which is why the default is one hour.

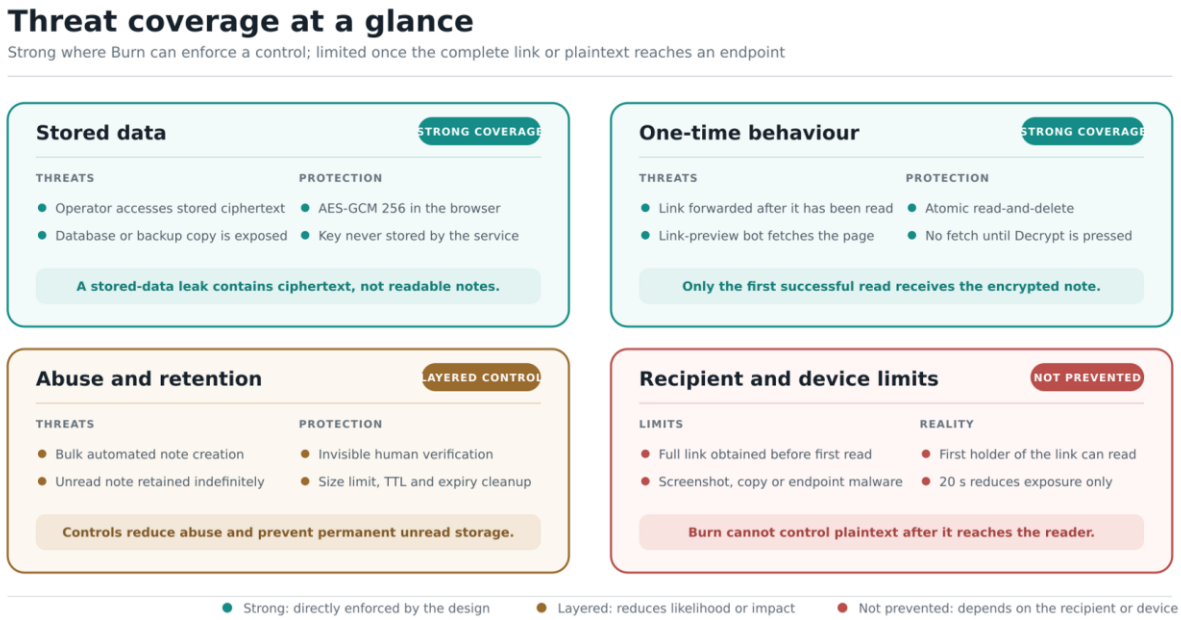

Figure 6. Threat coverage. Client-side encryption and the fragment carry most of the weight; the atomic delete handles the after-read cases.

5. Server behaviour

5.1 Routes

Route	Method	Behaviour
/	GET	Serves the create page with the four-step strip, the note panel and the footer
/privacy	GET	Serves the privacy page
/api/create	POST	Verifies the challenge token, validates size and TTL, generates an id, inserts the row, returns the id
/n/<id>	GET	Serves the read page. Does not touch the database
/api/read/<id>	On demand	Single statement DELETE ... RETURNING; returns ciphertext and IV once, or not found
expiry maintenance	Scheduled	Hourly: DELETE FROM notes WHERE expires_at < now

The routes below describe the public interface and its intended behaviour. Internal hosting and deployment details are deliberately omitted.

5.2 The atomic read

The read is one statement, not a SELECT followed by a DELETE. Two readers pressing Decrypt at the same instant cannot both receive the note, because the database returns the row to whichever DELETE wins and returns nothing to the other. There is no window in which the note has been read but still exists.

```
DELETE FROM notes WHERE id = ?1 AND expires_at > ?2 RETURNING ciphertext, iv
```

The expiry condition is included in the delete so a note that has passed its lifetime but not yet been swept behaves as already gone. The sweep is a tidy-up, not the enforcement mechanism.

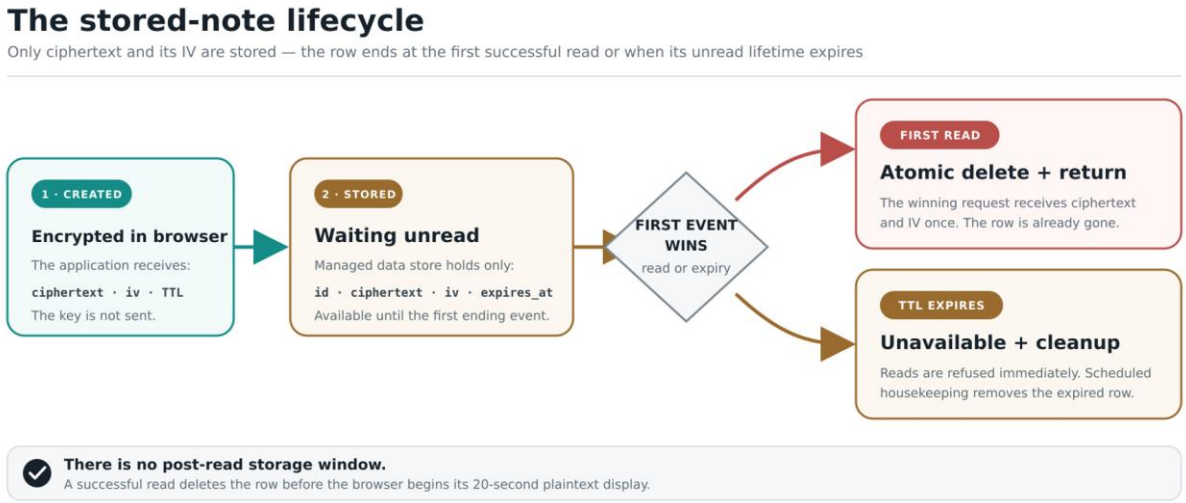

Figure 7. The server holds ciphertext from creation until the first read or the end of the unread lifetime, whichever is sooner, and nothing afterwards.

5.3 Abuse prevention

Note creation is protected by an invisible human verification challenge. The challenge runs in the sender's browser without a visible widget, and the resulting token is verified by the application before any note is stored. The privacy page explains this processing. Reads are not challenged: the reader has already been given a link, and the atomic delete means a guessed identifier gains nothing without the key.

Upstream web application firewall rules and shared rate limits apply before requests reach the application, in line with the other SecurityNut tools.

6. Data model

Column	Representation	Purpose
id	Opaque identifier	Random URL-safe identifier, the only thing in the path
ciphertext	Encoded encrypted data	Base64url AES-GCM output, capped at 64 KB
iv	Encoded 12-byte value	Base64url 12-byte IV
created_at	Creation timestamp	Unix time of creation, for the sweep and for operational stats
expires_at	Expiry timestamp	created_at plus the chosen TTL

The managed data store contains only the fields required to identify, return and expire an encrypted note. There is no user table, no read log and no plaintext field of any kind. Hosting-specific connection names and deployment details are intentionally excluded from this public manual.

7. Interface and design

Burn wears the SecurityNut vault instrument panel: graphite background, brass accents, Chakra Petch for headings, IBM Plex Sans for body and JetBrains Mono with tabular numerals for the countdown. Fonts are served by Bunny Fonts. The brass shield favicon and the shared SecurityNut navigation carry Burn as the fifth entry, and Burn appears in the nav of the root, Crypto, Notes and IP.

- Wide screens: five floating panels with no outer card, steps one and two left, note centre, steps three and four right, boxes stretched to the note's height with large brass numerals and a mono footnote pinned to the bottom of each step.
- Countdown label reads Gone in 20s with a lowercase s, by decision.
- Footer line: Encrypted in your browser, deleted from our server the moment it is opened. Read once, then gone.
- html{scrollbar-gutter:stable} was removed in v1.0.3 because it shifted Burn left of the other four SecurityNut sites; Burn is a single page and does not need it.

8. Service integration

Burn is monitored by Pulse, the estate status board, which probes the site every five minutes and expects a successful response carrying the page title as a marker. The title check helps distinguish a healthy page from an incomplete application response. Anonymous visit counts are recorded through the suite's shared analytics service. Burn also appears on the public Digital Projects showcase at martinpaul.co.uk

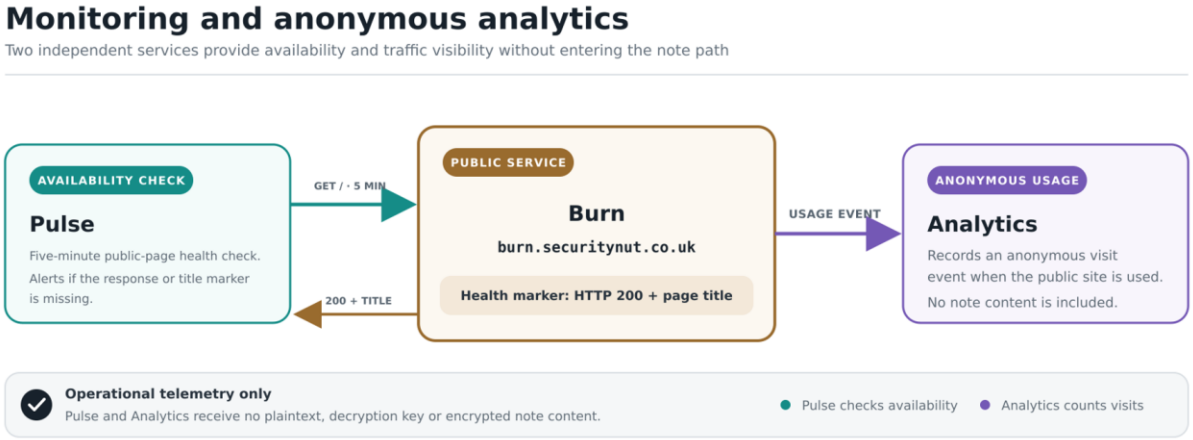

Figure 8. Burn's place in the shared monitoring and analytics services.

9. Change history

Version	Date	Change
1.0.0	10 Sept 2026	First deployment. Create and read flows, atomic delete, hourly sweep, visible verification widget, privacy page, four-step strip
1.0.1	10 Sept 2026	Verification switched to invisible mode; privacy page updated to explain automated verification
1.0.2	10 Sept 2026	Resolve sweep starts before the clipboard write; countdown label lowercase s; footer matched to the root site

Version	Date	Change
1.0.3	10 Sept 2026	scrollbar-gutter:stable removed to align Burn with the other SecurityNut pages
1.0.4	10 Sept 2026	Wide-screen layout introduced; privacy page narrowed; Contact section removed; verification disclosure confirmed
1.0.5	10 Sept 2026	Steps one and two left, three and four right, stretched to note height, brass numerals, Chakra Petch titles
1.0.6	10 Sept 2026	Panels float without an outer card; hairline under each step title; mono footnotes pinned to panel foot; fuller step three wording

10. Screenshot

Pictured below, the landing page of burn.securitynut.co.uk