MASKLINE · SECURITY

Security whitepaper

How Maskline keeps the text people work on from leaving their device: the design, the data it holds, how that is enforced and tested, and where it stops.

Edition November 2026 · content-5f678a25f1231e96

Contents

01 Summary
02 How it is built
03 What data exists, and where
04 Files and the masked copy
05 How it is enforced
06 Encryption
07 Accounts and access control
08 The browser extension
09 For organizations
10 Service providers
11 How we build and release
12 Limits
13 Reporting a security problem

01 Summary

Maskline finds private details in text (names, numbers, addresses, account and record numbers, secrets) and swaps them for placeholders such as [PERSON_1] before the text is pasted anywhere else, such as an AI tool, an email or a ticket. When text comes back with those placeholders, it puts the real details back in.

All of that happens on the person’s own device, in the browser. The text, the details found in it and the map from placeholders back to real values are never sent to us or to anyone else. We built the product so that we could not read a customer’s text even if we wanted to, and we test that on every change.

- Masking, review and restoring run in the browser tab or the browser extension, with no server involved.

- A deny-by-default security policy lets the app contact only itself, the account service and the one-time model download. Every change is tested by trying to send data out eight different ways.
- Saved work on the device is encrypted with AES-256-GCM, optionally under a passphrase only the person knows.
- A masked file is made on the device, cleaned of what its pages do not show, then read back and searched for every masked detail before it is saved.
- The account service holds account details, plan and content-free counts. It never receives text.

02 How it is built

Part	Where it runs	What it does
Studio	A web app in the browser, installable, works offline	Review and mask any text or file, restore AI answers, keep encrypted history
Browser extension	Chrome and Edge, on supported AI sites only	Checks a message on the page before it is sent and offers to mask it
Masking engine	Inside Studio and the extension, in a background worker	Pattern rules plus a small name-recognition model that runs on the device
Account service	Our backend (Supabase: Postgres, Auth, server functions)	Sign-in, plan, organization policies, content-free audit counts, billing records
Website	Static pages with one script of its own	Information only; it collects nothing

The name model is a file of about 66 MB downloaded once and kept in the browser. It is served from the app's own address, and only if that copy cannot be reached does the app fetch it from Hugging Face (huggingface.co, *.huggingface.co, *.hf.co), at the same pinned version. The download sends no text. After that, Studio works with no network at all.

03 What data exists, and where

On the device:

- Your text: What you paste, type or open.
- Masking engine: Finds details and swaps in placeholders, in your browser.
- Placeholder map: Pairs each placeholder with its real detail.
- Encrypted history: Sessions, words and settings, encrypted with AES-256-GCM.

What leaves the device, and exactly what goes with it:

- Wherever you paste it: Only the masked text you copy or send yourself, with placeholders like [PERSON_1].
- Maskline account service: Your email, plan, Free plan counts and error reports. It never receives your text.
- Stripe, if you pay: Your email, plan and seat count. Card details go to Stripe directly, never to us.
- Name model download: One download of the name model, from us or Hugging Face. No text is sent.

Data	Held by	Who can read it	How long
Text, findings and placeholder maps	The device only	The person	Until closed, or in history for the period they choose (30 days by default)
Words library and settings	The device, encrypted	The person	Until changed or erased
Documents opened as pages (a Word file or a PDF, with its formatting)	The device, encrypted, with its session	The person	As long as its session is kept in history
AI answers pasted or opened in Restore, and the restored copy	Memory only	The person	Until the tab closes

Data	Held by	Who can read it	How long
Pictures read by text recognition, and the words read in them	Memory only	The person	Until the tab closes
File names	Not kept anywhere	Nobody	Never stored
Email, name and sign-up details	Account service	The person; support staff when needed	Until the account is deleted
Plan, seats and Stripe ids	Account service and Stripe	The person and their organization’s admins	While the account exists
Random id per session or file, and a text’s length, for the Free plan’s 3 a day	Account service	Nobody but the server	7 days; nothing is kept for paid plans
Audit events (counts per kind of detail, site name, action)	Account service	The organization’s admins; nobody can change or delete them	365 days
Which getting-started emails were sent, and an unsubscribe choice	Account service	Nobody but the server	Until the account is deleted
Organization policies and enrollment tokens	Account service	The organization’s admins and its enrolled devices	Until the organization removes them

04 Files and the masked copy

Studio opens a Word document or a PDF as its own pages and saves the masked copy as a file. A file says more than its pages show, so a masked file is built to hold only what the person saw masked, and it is checked before it is saved.

- Opening. Files are read in the browser tab and nothing in them is run: no script, no macro, no form. A PDF's pages are drawn in the tab, and the fonts a PDF may name are served by the app itself, so opening a file asks nothing of any other host. Scanned pages and images are read by text recognition whose program and language data are the app's own files too.
- Storage. A document is kept with its session, in the same encrypted store as history, and is deleted with it. File names are not kept, because a file name can hold a person's name.
- A masked Word file keeps its layout, with the placeholders written into every part of it: body, headers, footers, notes and text boxes. What the pages do not show is taken out: author and other properties, comments, tracked changes, embedded files, macros and links to outside files.
- A masked PDF keeps its pages. The letters of each masked detail are removed from the page's content, not covered, and the placeholder is written into the content where they stood, so a tool that reads the file reads each placeholder in its place. A page whose text cannot be removed, such as a scan, is saved as a picture with its items painted over, and the person is told which pages. Properties, bookmarks, comments, form fields, links, attached files and scripts are left out.
- A picture keeps its dots and nothing else: what a camera or a program wrote into the picture's file (place, time, device, comments) is taken out.
- The check before saving. Every finished file (Word, PDF, Excel, CSV, Markdown, web page and plain text) is read back and searched for every masked value, in every place of the file that can hold words. A PDF is read back by two different PDF engines. If a value is found, nothing is saved and the person is told where it was. No setting turns this check off.
- Pictures. What a picture shows is not text, so masking does not read it. The person is told how many pictures a document has and can have them read by text recognition on the device, leave out the ones in which a detail was read, or leave out all of them. After saving, the pictures that were left in without being checked are named. Finding nothing is never described as safe.
- Restoring. A file the AI gives back is opened the same way, and the real details are written into it on the device: into a Word file's own layout, or onto a PDF's own pages. Neither the answer nor the restored copy is stored.

05 How it is enforced

Promises in a privacy policy are not enough, so the apps are built to make sending text impossible and the release process proves it.

- Content Security Policy. Studio is served with a deny-by-default policy: scripts, styles and fonts come only from the app itself, and network connections go only to the app, the account service and the model download hosts. Forms, plugins and framing are refused. Even a bug or an injected script has nowhere to send text.
- The extension's own pages may run only the extension's files. Its fonts and model runtime ship inside it; its only outside requests are the name model download and the account service. For a signed-in person that is renewing the sign-in, counting each masked text (a random id and the date) and fetching their organization's sealed library; for a browser an organization enrolled, that organization's policy and sealed library, and the counts described under "For organizations". No text, finding or placeholder is in any of them.
- The website runs one script of its own and its policy allows no other: no inline code, nothing from another host. The script reads the signed-in hint the app leaves (a cookie that holds no account details) to show the account menu, and sends nothing.
- The app and the admin console share one sign-in: the session the account service issued is kept in a cookie on the brand domain that both read, so signing in or out of one does the same in the other. It holds tokens and the account's id and email, never text, and the pages it is sent with are plain files that do not read it.
- Every response is marked no-transform, so the host cannot rewrite pages or add scripts, and HTTPS is enforced with HSTS.

Checks that run on every proposed change, in a real browser:

Check	What it proves
Lock-down	Acting as hostile code inside the built Studio, it tries fetch, XHR, beacon, image, WebSocket, remote script, inline script and form submission to an outside host. All eight must be blocked, and the app must contact no host outside its list.
Offline	Loads Studio once, cuts the network, then runs a full mask and restore. What works offline cannot depend on sending text.
Page editor and files	In the built Studio, opens made-up Word and PDF documents of 30 pages, saves the masked copy as each kind of file and searches every saved file for every detail that was masked. Each mark on a PDF page must sit within 4 px of its text, and the masked PDF is read back with two PDF engines. The files are then given back as an AI would give them, and restored.

Check	What it proves
Website	Serves the website with its real headers and opens every page at phone and desktop sizes: no script but its own, no inline styles, no outside hosts, and the account menu works for a signed-in visitor.
Extension	Runs the built extension against copies of the AI sites: its layer stays sealed from the page, the hand-off to Studio and back works, and an organization’s policy is applied.
Live	After each release, compares the published site and app with what was built, so a changed page or an added script is caught at once.

06 Encryption

- At rest on the device: history, placeholder maps, the words library and settings are encrypted with AES-256-GCM before they are stored. Only the time of the last change is readable, so expired history can be deleted.
- Device key (default): the browser holds the key and will not hand it to the page. This keeps stored data unreadable as plain files, but relies on the computer’s own login and disk encryption, because the key lives in the same browser profile.
- Passphrase (optional): the key is locked with a passphrase only the person knows (PBKDF2-SHA-256, 600,000 rounds). Without it the data cannot be read, even with full access to the device. There is no reset, by design.
- In transit: every connection uses HTTPS. There is little to protect, because text is never transmitted.

07 Accounts and access control

- Sign-in is by email and password through Supabase Auth. Two-step sign-in with any authenticator app (TOTP) and single-use backup codes is available to every account.
- Once two-step sign-in is on, the database itself refuses the account’s requests until the code has been given in that session.

- Row-level security limits every query to the person’s own records and their own organization’s. Server-only operations run as narrow functions that check who is asking.
- Support staff use a separate console that needs two-step sign-in. It shows account details only (never text, which we do not have), cannot sign in as a customer, and records every action with a reason in a log nobody can change or delete.
- Payments run on Stripe’s own pages. Card details go to Stripe and never reach us.

08 The browser extension

Permission	Why
offscreen	A hidden page that hosts the masking engine
storage	Per-site choices, and the hand-off to Studio held in memory only and dropped once delivered
activeTab	Lets the toolbar panel see which site the current tab is on
scripting	Starts the extension on a site the person adds themselves
contextMenus	Adds Mask as to the right-click menu for text selected on a site the extension runs on. The browser shows no warning for it
cookies, and the app’s own address	Reads the sign-in the app keeps, for the app’s address only. The extension has no sign-in form and reads no other site’s cookies
Optional site access	Asked for one site at a time, only when the person chooses Protect this site

- It runs on the supported AI sites only, and checks messages on the computer.
- Its bubbles and prompts live in a sealed layer the site’s own code cannot read or restyle, and its files use per-session addresses so a site cannot probe for it.
- It never sends a message for the person and never changes what they wrote without asking.
- Open in Studio moves text, or a Word or PDF file being attached, from the AI tab to Studio inside the browser, never in an address or through a server. A file opened this way is not given to the AI site.

- A person’s words library reaches the extension from Studio inside the browser, never through a server. The extension keeps it sealed with a key the browser will not hand out, masks its words on AI sites with Studio closed, and gives them to a page only while the account they belong to is signed in. A word added on an AI site with Mask as is kept the same way and handed to Studio inside the browser.
- An organization can push an enrollment token with the browser’s own management tools. The extension then fetches the organization’s policy with that token alone, and keeps the last policy if the service cannot be reached.

09 For organizations

- Policies: admins publish numbered policy versions (blocked or allowed sites, what to do with each kind of detail, whether Send anyway is allowed). Enrolled browsers apply a new version within five minutes.
- Audit events record what happened, never what was written: an id, the time, the person or device, the app, the site’s host name, one of six fixed actions (one of them a send with nothing found, so every send is counted), counts per kind of detail, and the policy version. The service refuses any event with a field the format does not name, so text cannot be slipped in.
- Admins read their own organization’s events and nobody can change or delete them.

10 Service providers

Provider	Used for	What it receives
Supabase	Account service: database, sign-in, server functions	Account details, plan, policies, audit counts. Never text
Cloudflare	Hosting the website and Studio	Ordinary web requests for the app’s files. Never text
Stripe	Payments	Email, plan and seat count; card details directly
Hugging Face	The one-time name model download	A file request. Never text

Provider	Used for	What it receives
Resend	Getting-started emails and invitations, sent by the account service	The email address, a first name or the inviter's name, the organization's name and the email itself. Never text

11 How we build and release

- Every change goes through a pull request and the checks above. A change that fails any of them cannot ship.
- Only synthetic data is used in code, tests, examples and screenshots. No real customer data is ever used to build or test the product.
- Dependencies are pinned in a lockfile and installed exactly as recorded.
- Studio and the website are published automatically from the main branch, then checked live.

Still to come before general availability, and not claimed until done: a written threat model, a privacy test that records all network traffic across every journey, an accessibility audit, and an independent penetration test with a summary letter.

12 Limits

- Detection is not perfect. You see everything that was found and decide before the text goes anywhere.
- Anything you choose to keep, or that detection misses, goes with the text to the tool you use.
- Malware, a compromised browser or someone at your unlocked computer can read what is on screen. Use disk encryption and screen locks.
- Other browser extensions with permission to read pages can read what you type, here as anywhere. Managed browsers should limit extensions.
- What a picture shows is not text. Unless the person has it read, a picture in a document is not checked, and text recognition reads English print and can miss words, above all handwriting. The same holds for a scanned page: a word that is misread is not found.
- A PDF the AI gives back with a placeholder cut in two by a line break gets that detail back in the text, but not on the page.

The first model download tells the model host that the app was opened, as any file download does. It reveals nothing about the text.

13 Reporting a security problem

Write to us with "Security" in the subject. Tell us what you found and how to reproduce it. We aim to reply within two working days, keep you updated, and credit you if you wish. Please do not access other people’s data or disrupt the service while testing.

Contact: support@maskline.ai. Our security.txt is at https://maskline.ai/.well-known/security.txt.