

Lesson 03 — DLP While Using AI

Data Loss Prevention, rewritten for a world where the biggest egress channel is the prompt box on your screen.

Audience: Everyone

AI OpSec

Foundations

Notes: The lesson is a habit, not a product. Four classes, one test, one redaction reflex.

What you will learn

- **Data classification for AI prompts** — a simple four-class model you can run in your head
 - **The redaction habit** — how to mask values without destroying the question
 - **The "email test"** — a one-line heuristic that catches most bad prompts before they leave
-

What is DLP?

Classic Data Loss Prevention is a set of controls that watch the ways data can leave your organization — email, uploads, USB, chat — and block or flag anything sensitive.

AI adds a new egress channel that most classic DLP tools do not see: **the prompt box**. Every time you paste into a chat window, an IDE assistant, or a browser extension, you are performing an egress event.

New definition

DLP while using AI = preventing sensitive data from entering a prompt in the first place.

A four-class model

Class	Examples
Never	Secrets, credentials, API keys, PII of real customers
Enterprise tier only	Source code, internal docs, infra diagrams, ticket contents with internal names
With redaction	Stack traces, SQL, log snippets, configs
Free	Public docs, open-source, generic questions

The "email test"

Would I email this to an unknown vendor?

If the answer is no, do not paste it into a prompt.

- The AI provider **is** an unknown vendor the first time you use it
 - A prompt is a message to a third party, not a private thought
 - The test works because your brain already knows how to answer it for email
-

Redaction that still lets the AI help — stack trace

Raw (do not send)

```
Traceback (most recent call last):  
  File "/srv/acme/orders/api.py", line 142, in checkout  
    user = db.get_user("jane.smith@acme.corp")  
ConnectionError: could not connect to  
db-prod-3.internal.acme.corp:5432
```

Redacted (safe)

```
Traceback (most recent call last):  
  File "<app>/orders/api.py", line 142, in checkout  
    user = db.get_user("<user@example.com>")  
ConnectionError: could not connect to  
<db-host>:5432
```

Keep the shape, mask the values. The model can still diagnose the error.

Redaction, part 2 — SQL query

Raw (do not send)

```
SELECT order_id, total, status
FROM orders_prod_2026
WHERE customer_id = 'cust_84312'
      AND created_at >= '2026-01-01'
      AND created_at <  '2026-02-01';
```

Redacted (safe)

```
SELECT order_id, total, status
FROM orders_table
WHERE customer_id = '<customer_id>'
      AND created_at >= '2026-01-01'
      AND created_at <  '2026-02-01';
```

Dates alone are low-risk. Real table names and real customer IDs are not.

Side channels you forget

- **File uploads** — dragging a PDF or CSV is the same as pasting it
- **Screenshots** — a screenshot of a ticket is a ticket, and the model reads it
- **Clipboard history** — the last thing you copied may ride along with the next paste
- **IDE autocomplete** — every keystroke in a completion-enabled editor is a prompt

If it can reach the model, it is a prompt. Treat it like one.

Zero-retention is not invisibility

Zero-retention is a contract, not a magic shield

It changes what the provider promises to keep on disk by default. It does not change physics, law, or operational reality.

- **Provider admins** can still see live traffic during abuse investigations
 - **Legal orders** can compel preservation, overriding the default
 - **Breaches** happen to every vendor, eventually
-

Example — redacting an auth bug

Raw report

```
User alice@acme.corp reports that  
POST /api/v2/sessions/refresh returns  
401 even though her cookie  
SESSION=abc123xyz789qrs456tuv  
is still inside its TTL.  
cc @platform-core @identity-team
```

Safe to paste

```
A user reports that  
POST /sessions/refresh returns 401  
even though the session cookie  
appears to still be inside its TTL.  
What are common causes?
```

Describe the bug. Never paste a live cookie "to illustrate the shape."

Rules to remember

- Classify every prompt before you send it: never / enterprise-only / with-redaction / free.
- Run the email test: if you wouldn't email it to a vendor, don't paste it.
- Redact values, keep shapes — replace real IDs/secrets/hostnames with `<placeholder>`, keep the structure intact.
- Screenshots and file uploads are prompts. Treat them the same way.
- Enterprise tier protects you from training, not from logs, breaches, or subpoenas.

