0-DAY NO CVE HIGH PUBLIC

Element Web, fragment guest creds wipe a session

Element Web v1.12.29 loadSession prefers guest_user_id and guest_access_token from the URL hash over restore. Guests default on. clearStorage runs before whoami. A garbage token logs the victim out and wipes local crypto. No CVE yet.

NAME Element Web, fragment guest creds wipe a session	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR Element
AFFECTED Element Web v1.12.29 (2d90d6b). Guests enabled (default). app.element.io config.json does not set disable_guests. Gadget lab of loadSession.	PUBLISHED 12 Oct 2026
UPDATED 12 Oct 2026	TAGS 0-day, element-web, session, csrf, cwe-384, cwe-352, cwe-613, unauthenticated

TL;DR

Lab files and proof-of-concept **here**.

01 the comment said these keys looked unused. they still win

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/element-fragment-guest** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, gadget.js. Authorized lab only. It talks to loopback.

This is **Element Web v1.12.29** (`2d90d6b7b601ecb9ccdf81239a43b4396b4b0c9e`). No CVE yet. CWE-384 / CWE-352 / CWE-613. 7.1 High (`CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:H`). Remote session destruction. UI:R. Not token theft. Not OS RCE.

`Lifecycle.loadSession` prefers `guest_user_id` and `guest_access_token` from the URL hash over `restoreSessionFromStorage`. Guests are on by default. `onTokenLoginCompleted` strips SSO and OIDC params only. It never strips the guest keys. `doSetLoggedIn(..., true)` runs `clearStorage()` before any `/_matrix/client/v3/account/whoami`. A garbage token is enough.

Same pin: `vector:` Windows drive-letter, Univention ICS javascript: href.

02 Why Element Web

I had just finished Moodle leftovers and wanted a client that people actually live in. Element Web is a large TypeScript tree with a history of XSS and token-handling GHSAs. The security model is honest about Desktop: sandbox on, `nodeIntegration` off, XSS is Matrix ATO not OS RCE. Session restore is a different class. Logout CSRF that also wipes the E2EE store is High even without stealing the token.

I only wanted Critical or High. Local is in-scope. Medium stays on the floor unless it chains. Univention ICS is the Medium sibling. This one is the High.

03 Pinning v1.12.29

Latest GitHub release on the day I cloned was 22 Sep 2026, tag `v1.12.29`, commit `2d90d6b`. Monorepo: `apps/web`, `apps/desktop`, `packages/*`, `modules/*`. I did not take a nightly.

04 The skip list is half the hunt

On this tag the interesting GHSAs were already closed:

- GHSA-wqmv-r2qj-2j9p HTML export XSS (filename / MXID), patched 1.12.28
- GHSA-9r5h-8m2x-w7q6 bundled URL preview; `previewFromBundle` requires `http:` / `https:`
- GHSA-wrcp-5v3v-3j6v / CVE-2026-55850 homeserver homepage HTML; `sanitizedHtmlNode`
- GHSA-m6c8-98f4-75rr room predecessor spoof, patched 1.11.112
- GHSA-69q3-jg79-cg79 malicious Element Call media keys, patched 1.11.97
- GHSA-3jm3-x98c-r34x access token via authenticated media; SW origin check

SSO `loginToken` is query-only and stripped after use. That is the control I wanted guest keys to copy. They do not.

05 Surfaces that returned nothing

Five hunts: timeline/export XSS, widgets/tokens/Call, Electron IPC, SSO/crypto/session, URL-preview/CSP/modules.

Timeline `formatted_body` XSS did not fire on this pin. Widget `$matrix_access_token` stayed in the SW origin check. Electron XSS did not become OS RCE (`nodeIntegration: false` , sandbox). SSO token-in-URL is stripped. Those hunts returned NONE in this bar.

Session restore did not.

06 The guest-first branch

`apps/web/src/Lifecycle.ts` :

PLAIN TEXT Copy

```
if (enableGuest && guestHsUrl && urlParams?.guest?.guest_user_id && urlParams?.g
    logger.log("Using guest access credentials");
    await doSetLoggedIn(
        {
            userId: urlParams.guest.guest_user_id,
            accessToken: urlParams.guest.guest_access_token,
            homeserverUrl: guestHsUrl,
            identityServerUrl: guestIsUrl,
            guest: true,
        },
        true,
        false,
    );
    return true;
}
```

`doSetLoggedIn` with `clearStorageEnabled=true` calls `clearStorage()` before the Matrix client starts. There is no whoami on this path.

`apps/web/src/vector/app.tsx` defaults guests on:

PLAIN TEXT Copy

```
enableGuest={!config.disable_guests}
```

`apps/web/element.io/app/config.json` does not set `disable_guests`.

`url_utils.ts` already comments that these fragment params look unused. They still parse. Query-string guest params are ignored because the keys live in the hash.

07 The lab

Gadget of the pin functions. Not a live Element SPA. `how=gadget`. Loopback `:19600`. Compose project `element-fragment-guest`.

Seeded `@victim:example.org` plus a crypto pickle, then opened `#/home?guest_user_id=@x:example.org&guest_access_token=not-a-real-token`. `used_guest=yes`, `prior_session_kept=no`, `crypto_empty=yes`, `whoami_called=no`. Query-string-only guest params were ignored. `disable_guests` kept the victim. A patched loadSession that skips guest when a non-guest session exists also kept it.

PLAIN TEXT Copy

```
SUCCESS element-fragment-guest how=gadget guest-wipe=yes prior-kept=no garbage-to
```

PLAIN TEXT Copy

```
python3 element-fragment-guest-Abraxas-Labs.py
```

or `cd lab && ./run.sh`. Target only `http://127.0.0.1:19600`. Stack tears down after the run.

08 Wrong turns, in the order I made them

Treating query-string `?guest_user_id=` as the vector. Keys live in the hash. Query-only is the negative.

Calling this token theft. The real access token is deleted, not sent to the attacker. Logout and crypto wipe. If the guest token is valid on the default HS, the victim is parked on that guest. That is fixation, not exfil.

Expecting `disable_guests` on app.element.io config. It is unset. Default is on.

Expecting `onTokenLoginCompleted` to strip guest keys. It strips SSO and OIDC. Guest keys stay in the hash, so a reload does it again.

Booting a full Element SPA. The guest-first branch is the leftover. A gadget of `loadSession` is the honest lab.

A reverse shell. Theatre. The oracle is the wiped pickle plus `whoami_called=no`.

09 What an attacker can do

Get a signed-in user to open a hash URL with guest fragment params. The client logs them out, wipes local crypto, and may park them on an attacker guest.

Class: Session destruction. **Reach:** Remote.

PRIVILEGE	METHOD / SINK
Parse guest creds from the hash	`url_utils.ts` fragment k `guest_user_id` / `guest_access_token`

PRIVILEGE	METHOD / SINK
Prefer guest over a stored non-guest session	`Lifecycle.loadSession` g branch
Wipe localStorage, IndexedDB account, sessionStorage, crypto	`doSetLoggedIn(...,` `clearStorageEnabled=true` `clearStorage()`
Skip token validation	Guest path never calls `/_matrix/client/v3/accou`
Persist attacker guest creds	`persistCredentials` after
Fixate onto a real guest on the default HS	Same persist if the guest to
Keep doing this on reload	Guest keys stay in the hash

Closed: steal the real access token; OS RCE / LPE; query-string guest params; `disable_guests: true` keeps the non-guest session.

10 The fix

Ignore guest fragment params when a non-guest session already exists. Validate the guest token with `/_matrix/client/v3/account/whoami` before `clearStorage`. Strip `guest_user_id` and `guest_access_token` the same way SSO tokens are stripped. Default `disable_guests: true` for hosted app.element.io.

I am not going to print a permalink you can send to someone else's Element tab. The guest-first branch is the useful part. If you own the client, run the proof of concept against loopback.

11 References

- Proof of concept: abraxas/element-fragment-guest · element-fragment-guest-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py · gadget.js
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-384 · CWE-352 · CWE-613
- Tree: element-hq/element-web v1.12.29 (`2d90d6b`) · `Lifecycle.ts` · `url_utils.ts` · `app.tsx`
- Same pin: element-vector-drive-letter · element-ics-javascript

A
X
Abraxas Labs

Independent security research by abraxas.

Analysis of 0-days, n-days, malware, and the

leftovers that still matter.

GitHub **@abraxas_null** **abraxas.null@proton.me** **abraxaslabs.tech**

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs