menelausx / **Calibre-Web-Automated I.md** Secret

Last active 18 hours ago

Code Revisions 2

Calibre-Web-Automated IDOR

Calibre-Web-Automated I.md

I found a security issue in the Kobo authentication flow that allows any authenticated user to impersonate other users.

Public issus report

crocodilestick/Calibre-Web-Automated#1303

/kobo_auth/generate_auth_token/int:user_id only requires a logged-in user and never checks that user_id == current_user.id or that the caller is an admin before looking up or creating a Kobo token for that user at cps/kobo_auth.py (line 70). It then renders the victim's token back in the response at cps/kobo_auth.py (line 104). A normal user can just request another user's numeric ID and obtain that user's Kobo auth token.

```
@kobo_auth.route("/generate_auth_token/<int:user_id>")
@user_login_required
def generate_auth_token(user_id):
    warning = False
    host_list = request.host.rsplit(':')
    if len(host_list) == 1:
        host = ':'.join(host_list)
    else:
        host = ':'.join(host_list[0:-1])
    if host.startswith('127.') or host.lower() == 'localhost' or host.startsw
        warning = _('Please access Calibre-Web Automated from non localhost t

    # Generate auth token if none is existing for this user
    auth_token = ub.session.query(ub.RemoteAuthToken).filter(
        ub.RemoteAuthToken.user_id == user_id
    ).filter(ub.RemoteAuthToken.token_type==1).first()
```

```
    if not auth_token:
        auth_token = ub.RemoteAuthToken()
        auth_token.user_id = user_id
        auth_token.expiration = datetime.max
        auth_token.auth_token = (hexlify(urandom(16))).decode("utf-8")
        auth_token.token_type = 1

        ub.session.add(auth_token)
        ub.session_commit()

    books = calibre_db.session.query(db.Books).join(db.Data).all()

    for book in books:
        formats = [data.format for data in book.data]
        if 'KEPUB' not in formats and config.config_kepubifypath and 'EPUB' i
            helper.convert_book_format(book.id, config.config_calibre_dir, 'E

    return render_title_template(
        "generate_kobo_auth_url.html",
        title=_("Kobo Setup"),
        auth_token=auth_token.auth_token,
        warning=warning
    )
```

That token is the authenticator for the Kobo API. requires_kobo_auth looks up the user solely by RemoteAuthToken.auth_token and logs them in as that user at cps/kobo_auth.py (line 140). So once an attacker gets the victim's token, they can call /kobo/<auth_token>/... as the victim and access whatever that victim's Kobo session is allowed to access, such as sync and download flows in cps/kobo.py (line 154).

```
def requires_kobo_auth(f):
    @wraps(f)
    def inner(*args, **kwargs):
        auth_token = get_auth_token()
        if auth_token is not None:
            try:
                limiter.check()
            except RateLimitExceeded:
                return abort(429)
            except (ConnectionError, Exception) as e:
                log.error("Connection error to limiter backend: %s", e)
                return abort(429)
            user = (
                ub.session.query(ub.User)
                .join(ub.RemoteAuthToken)
                .filter(ub.RemoteAuthToken.auth_token == auth_token).filter(u
```

```
                .first()
            )
            if user is not None:
                login_user(user)
                [limiter.limiter.storage.clear(k.key) for k in limiter.curren
                return f(*args, **kwargs)
        log.debug("Received Kobo request without a recognizable auth token.")
        return abort(401)
    return inner
```

The application's user_id is an auto-incrementing integer starting from 1, making it very easy to predict other users, including administrators.

Reproduction:

- Enable Kobo sync
- Log in as any low-privileged user.
- Request /kobo_auth/generate_auth_token/<victim_id>.
- Use the returned token against /kobo//v1/initialization.
- The server authenticates the session as the victim.

Reproduced video

https://drive.google.com/file/d/14GC9DnEIl4DInBluiN5tYfrmvboEFEkT/view?usp=drive_link