HTB — Soulmate

I scanned the box, found SSH and a web server, and discovered a CrushFTP instance on a subdomain. I abused an auth bypass in that CrushFTP build to create an account, uploaded a web shell, and gained a low-privilege web foothold as www-data. From there I found a hard-coded SSH password in an Erlang startup script, SSHed in as ben, and discovered an Erlang SSH service on port 2222 running as root. Dropping into the Erlang shell and using its command API let me run OS commands as root and read the root flag.

Recon

I started simple: port scan, hosts file entry, directory fuzzing, and subdomain enumeration. The main site looked like a dating app, but the ftp subdomain stood out — it was running CrushFTP and the version string matched a known auth bypass.

Web findings and foothold

The CrushFTP instance allowed me to register or create an account because of the version-specific issue. With that account I poked around the exposed directories until I found an upload area for the web production site. I uploaded a web shell and triggered it to get a shell as the web user (www-data).

Local enumeration and user takeover

From the web shell I ran the usual enumeration checks and a quick privilege-escalation audit. That flagged an Erlang process started by a system script. The startup script contained a plaintext credential for the local user ben. Using that password I SSHed in as ben, grabbed the user flag, and kept looking.

Erlang service and privilege escalation

There was an Erlang SSH service listening on port 2222 that accepted ben's credentials and dropped me into an Erlang shell running with root privileges. Erlang exposes an API that can execute OS commands; I used that capability to run commands as root and read the root flag in /root.

Takeaways

- Don't stop at the main hostname — subdomains can host completely different services with different risks.
- Version info in web assets is valuable. If a component's build/version lines up with a published vulnerability, it's worth investigating.
- Plaintext or hardcoded credentials in scripts and startup files are "one-click" compromises if you find them.
- Less-common runtimes (Erlang, Elixir, etc.) can expose powerful execution primitives that are easy to miss during basic audits.

— MalwareMusashi