⚙ 卷 02：WiseFido_CA_部署与配置手册

版本：v1.0
发布日期：2025-10-04
编制单位：WiseFido Engineering & Infrastructure Division

🧭 2.1 文件目的

本手册详细说明 **WiseFido CA 系统的部署与配置过程**，
指导工程师在 Ubuntu Server 24.04 环境中，通过 Docker Compose 快速搭建 Vault CA 服务器。

部署目标：

- 在服务器上启动 Vault PKI 服务；
- 生成 Root / Intermediate CA；
- 开启 HTTPS（Vault 原生 TLS）；
- 准备 IoT 设备证书签发接口；
- 验证部署结果可通过浏览器与 CLI 正常访问。

📦 2.2 部署环境说明

项目	参数
操作系统	Ubuntu Server 24.04 LTS (64-bit)
公网 IP	23.170.40.60
域名	`ca.wisefido.work`
部署方式	Docker Compose
Vault 版本	1.13.x (HashiCorp 官方镜像)
Docker 版本	25.0+
Docker Compose	v2.27+
运行用户	root / sudo 权限账户
证书目录	`/opt/wisefido-ca/`
数据卷挂载	`/vault/config /vault/data /vault/logs`

🧩 2.3 Docker Compose 配置

文件路径：
`03_deploy/01_docker-compose.yml`

```
version: '3.8'

services:
  vault:
    image: vault:1.13.2
    container_name: wisefido-vault
    restart: always
    ports:
      - "8200:8200"
    cap_add:
      - IPC_LOCK
    volumes:
      - ./vault/config:/vault/config
      - ./vault/data:/vault/data
      - ./vault/logs:/vault/logs
    environment:
      - VAULT_LOCAL_CONFIG={"listener":[{"tcp":
{"address":"0.0.0.0:8200","tls_cert_file":"/vault/config/vault_cert.pem","tls_k
ey_file":"/vault/config/vault_key.pem"}}],"api_addr":"https://ca.wisefido.work:
8200","disable_mlock":true}
    command: vault server -config=/vault/config/vault.hcl
```

- 执行命令：

```
cd /opt/WiseFido_CA_Project/03_deploy
docker compose up -d
```

- 验证容器运行：

```
docker ps
# 应显示 wisefido-vault 运行中
```

🧩 2.4 Vault 初始化与 Root CA 生成

- 以下脚本位于：04_scripts/

📦 脚本 01：初始化环境

- 文件名：04_scripts/01_setup_init_vault.sh

```
#!/bin/bash
# 初始化 Vault 环境
echo "🔷 初始化 Vault 环境..."
mkdir -p /opt/wisefido-ca/{root,intermediate,issued,crl}
```

```
docker exec -it wisefido-vault vault operator init -key-shares=3 -key-
threshold=2 > /opt/wisefido-ca/vault_init_keys.txt
echo "✅ Vault 初始化完成，密钥已保存：/opt/wisefido-ca/vault_init_keys.txt"
```

说明：

- 生成 3 份解封密钥（需任意 2 份可解封）；
- Root Token 输出在该文件末尾；
- 该文件仅首次初始化生成，务必离线备份。

🛠️ 脚本 02：解封 Vault

文件名：04_scripts/02_setup_unseal_vault.sh

```
#!/bin/bash
# 读取密钥进行解封
read -p "输入第一个 Unseal Key: " key1
read -p "输入第二个 Unseal Key: " key2
docker exec -it wisefido-vault vault operator unseal $key1
docker exec -it wisefido-vault vault operator unseal $key2
echo "✅ Vault 已解封完成。"
```

验证：

```
docker exec -it wisefido-vault vault status
```

返回结果应包含：

```
Sealed: false
Initialized: true
```

🔐 脚本 03：启用 PKI 并生成 Root CA

文件名：04_scripts/03_setup_generate_root_ca.sh

```
#!/bin/bash
# 使用 Root Token 登录
read -p "请输入 Vault Root Token: " token
export VAULT_ADDR=https://ca.wisefido.work:8200
export VAULT_TOKEN=$token
```

```
# 启用 PKI 引擎（Root）
docker exec -it wisefido-vault vault secrets enable -path=pki pki
docker exec -it wisefido-vault vault secrets tune -max-lease-ttl=87600h pki

# 生成 Root CA 证书
docker exec -it wisefido-vault vault write -format=json
pki/root/generate/exported \
  common_name="WiseFido Root CA" \
  organization="WiseFido Inc." \
  country="US" \
  ttl=87600h \
  > /opt/wisefido-ca/root/root_ca_export.json

# 提取证书与密钥
jq -r .data.certificate /opt/wisefido-ca/root/root_ca_export.json >
/opt/wisefido-ca/root/root_ca.crt
jq -r .data.private_key /opt/wisefido-ca/root/root_ca_export.json >
/opt/wisefido-ca/root/root_ca.key
echo "✅ Root CA 生成完成：/opt/wisefido-ca/root/root_ca.crt"
```

📄 Root CA 输出文件说明

文件名	路径	说明
root_ca.crt	/opt/wisefido-ca/root/	自签根证书（公钥）
root_ca.key	/opt/wisefido-ca/root/	根私钥（必须离线存储）
root_ca_export.json	/opt/wisefido-ca/root/	原始 Vault 导出 JSON 结果

安全建议：

- root_ca.key 文件应立即复制到离线介质（U盘/安全存储）；
- 从服务器中删除明文私钥副本；
- Root CA 仅用于签发 Intermediate，不直接用于任何 TLS 通信。

➡ 提示： 下一段（节 2.5～2.9）将继续包括：

- Intermediate CA 生成与导入
- Vault HTTPS 配置
- 签发示例证书
- 验证与测试指令
- 常见故障处理与重建流程

2.5 Intermediate CA 生成与导入

Intermediate CA 负责签发 IoT 设备与服务器证书。
Root CA 仅离线使用，一次性签发 Intermediate 证书。

脚本 04：创建 Intermediate CA

文件名：`04_scripts/04_setup_create_intermediate_ca.sh`

```
#!/bin/bash
set -euo pipefail
PROJECT_ROOT="/opt/00_WiseFido_CA_Project"
INT_DIR="${PROJECT_ROOT}/05_opt/01_wisefido-ca/02_intermediate"

read -p "请输入 Vault Root Token: " token
export VAULT_ADDR="https://ca.wisefido.work:8200"
export VAULT_TOKEN="$token"
export VAULT_SKIP_VERIFY=true

# 启用 Intermediate PKI 引擎
docker exec -i wisefido-vault vault secrets enable -path=pki_int pki
docker exec -i wisefido-vault vault secrets tune -max-lease-ttl=43800h pki_int

# 生成 Intermediate CSR
docker exec -i wisefido-vault vault write -field=csr
pki_int/intermediate/generate/internal \
  common_name="WiseFido Intermediate CA" organization="WiseFido Inc."
country="US" ttl=43800h \
  > "${INT_DIR}/intermediate.csr"

echo "✅ Intermediate CSR 生成: ${INT_DIR}/intermediate.csr"
echo "🔷 请使用 Root CA 离线签署此 CSR..."
```

离线 Root 签署 Intermediate

在离线 Root 环境中（例如安全工作站）执行：

```
cd /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-ca
openssl x509 -req -in 02_intermediate/intermediate.csr \
  -CA 01_root/root_ca.crt -CAkey 01_root/root_ca.key -CAcreateserial \
  -out 02_intermediate/intermediate.crt -days 1825 \
  -extensions v3_ca -extfile <(printf "
[v3_ca]\nbasicConstraints=CA:TRUE,pathlen:0")
```

然后将生成的 intermediate.crt 导回服务器，执行：

```
docker cp /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/02_intermediate/intermediate.crt wisefido-vault:/vault/
docker exec -i wisefido-vault vault write pki_int/intermediate/set-signed
certificate=@/vault/intermediate.crt
```

配置证书 URL：

```
docker exec -i wisefido-vault vault write pki_int/config/urls \
  issuing_certificates="https://ca.wisefido.work:8200/v1/pki_int/ca" \
  crl_distribution_points="https://ca.wisefido.work:8200/v1/pki_int/crl"
```

✅ 完成后 Vault 已具备签发能力。
所有签发的证书都由 Intermediate CA 私钥签名，并最终由 Root CA 链接信任

⚙ 2.6 Vault HTTPS 正式证书配置

Vault 初次启动使用了临时自签证书。
在生成 Intermediate 后，我们可以签发正式的服务器证书供 Vault HTTPS 使用。

🧩 脚本 05：为 Vault 签发正式证书

文件名：04_scripts/05_setup_configure_https.sh

```
#!/bin/bash
set -euo pipefail
PROJECT_ROOT="/opt/00_WiseFido_CA_Project"
CONF_DIR="${PROJECT_ROOT}/02_config"

read -p "请输入 Vault Root Token: " token
export VAULT_ADDR="https://ca.wisefido.work:8200"
export VAULT_TOKEN="$token"
export VAULT_SKIP_VERIFY=true

# 创建角色（允许签发服务器证书）
docker exec -i wisefido-vault vault write pki_int/roles/vault-server-role \
  allowed_domains="wisefido.work" allow_subdomains=true max_ttl="8760h"

# 签发服务器证书
docker exec -i wisefido-vault vault write -format=json pki_int/issue/vault-
server-role \
  common_name="ca.wisefido.work" ttl="8760h" >
"${CONF_DIR}/vault_server_cert.json"

jq -r .data.certificate "${CONF_DIR}/vault_server_cert.json" >
"${CONF_DIR}/vault_cert.pem"
```

```
jq -r .data.private_key "${CONF_DIR}/vault_server_cert.json" >
"${CONF_DIR}/vault_key.pem"

echo "✅ 新 HTTPS 证书生成完成，路径：${CONF_DIR}/vault_cert.pem"
echo "🔄 重启 Vault 容器以加载新证书..."
cd "${PROJECT_ROOT}/03_deploy"
docker compose restart vault
```

- 验证 Vault UI：在浏览器中访问https://ca.wisefido.work:8200
- 应能正确显示 HTTPS 并由 WiseFido Intermediate CA 签发。

🧩 2.7 签发测试证书（服务器与 IoT 设备）
示例：签发服务器证书

```
docker exec -i wisefido-vault vault write -format=json pki_int/issue/vault-
server-role \
  common_name="api.wisefido.work" ttl="4380h" >
/opt/00_WiseFido_CA_Project/05_opt/01_wisefido-ca/03_issued/server_api.json

jq -r .data.certificate /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/server_api.json > /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/server_api.crt
jq -r .data.private_key /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/server_api.json > /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/server_api.key
```

示例：签发 IoT 设备证书

```
docker exec -i wisefido-vault vault write -format=json pki_int/issue/device-
role \
  common_name="iot-device-001.wisefido.work" ttl="8760h" >
/opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/01_devices/device_001.json

jq -r .data.certificate /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/01_devices/device_001.json >
/opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/01_devices/device_001.crt
jq -r .data.private_key /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/01_devices/device_001.json >
/opt/00_WiseFido_CA_Project/05_opt/01_wisefido-
ca/03_issued/01_devices/device_001.key
```

IoT 设备可将 device_001.crt 与 Root CA 链写入安全芯片，完成出厂注册。

🩹 2.8 验证与测试

1️⃣ 验证证书链

```
openssl verify -CAfile /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-ca/01_root/root_ca.crt \
  /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-ca/02_intermediate/intermediate.crt
```

2️⃣ 验证 Vault HTTPS

```
curl -v --cacert /opt/00_WiseFido_CA_Project/05_opt/01_wisefido-ca/01_root/root_ca.crt https://ca.wisefido.work:8200/v1/sys/health
```

返回示例：

```
HTTP/2 200
{
  "initialized": true,
  "sealed": false,
  "standby": false
}
```

3️⃣ 检查审计日志

```
docker exec -i wisefido-vault cat /vault/logs/audit.log | jq .
```

应包含签发操作记录。

📠 2.9 常见问题与恢复流程

问题场景	可能原因	解决步骤
Vault 容器启动失败	TLS 文件缺失或权限错误	确认 `/opt/00_WiseFido_CA_Project/02_config/vault_cert.pem` 与 `vault_key.pem` 存在且权限为 644/600
HTTPS 报错 "certificate verify failed"	临时自签证书仍在使用	执行脚本 05 重新签发正式证书并 `docker compose restart`
`vault operator init` 已运行过	重复初始化	删除数据卷重新部署：`docker compose down -v`

问题场景	可能原因	解决步骤
Intermediate 导入报错	CSR 或证书路径错误	确认 `.csr`、`.crt` 文件均存在且 Vault 已解封
浏览器无法访问 8200	防火墙或安全组未放行	开放 TCP 8200 端口
审计日志空白	未启用审计	运行：`vault audit enable file file_path=/vault/logs/audit.log`

✅ 部署完成标志

项目	验证命令	正常输出
Vault 状态	`docker exec -it wisefido-vault vault status`	`Sealed: false`
Root CA	`ls 05_opt/01_wisefido-ca/01_root/root_ca.crt`	存在
Intermediate CA	`ls 05_opt/01_wisefido-ca/02_intermediate/intermediate.crt`	存在
HTTPS 访问	浏览器打开 `https://ca.wisefido.work:8200`	正常响应
审计日志	`docker exec -it wisefido-vault cat /vault/logs/audit.log`	有签发记录

编制人： WiseFido 系统架构组
审核人： Chief Security Officer
批准人： WiseFido Engineering Director
发布日期： 2025-10-04