OpenSSL

Pentesting Cheat Sheet

SSL/TLS & Certificate Operations

Author: 0xNetrunner
Date: December 5th, 2025
Version: 1.0
Category: SSL/TLS Security

Created with LaTeX ♥ Rose Pine Dawn Theme

Contents

1 Introduction

OpenSSL is a robust, full-featured toolkit for SSL/TLS protocols and general-purpose cryptography. In penetration testing, it's essential for certificate manipulation, SSL/TLS reconnaissance, and cryptographic operations.

Info

OpenSSL is pre-installed on most Linux distributions. It's invaluable for extracting credentials from certificates, testing SSL configurations, and manipulating cryptographic data during engagements.

1.1 Installation

Kali Linux / Debian:

```
sudo apt update && sudo apt install openssl
```

Check Version:

```
openssl version -a
```

2 PKCS12 / PFX Operations

PKCS#12 (.pfx, .p12) files bundle certificates and private keys together. These are commonly found during pentests and often contain authentication credentials.

Warning

PFX files from Windows environments often contain domain authentication certificates. Extracting these can provide direct access to systems via tools like Evil-WinRM or SSH.

2.1 Extract Private Key from PFX

```
# Extract private key (will prompt for password)
openssl pkcs12 -in cert.pfx -nocerts -out key.pem

# Extract private key without encryption (-nodes = no DES)
openssl pkcs12 -in cert.pfx -nocerts -out key.pem -nodes

# Example from HTB machines:
openssl pkcs12 -in legacyy_dev_auth.pfx -nocerts -out key.pem -nodes
```

2.2 Extract Certificate from PFX

```
# Extract certificate only (no private keys)
openssl pkcs12 -in cert.pfx -nokeys -out cert.pem

# Extract certificate chain
openssl pkcs12 -in cert.pfx -nokeys -chain -out fullchain.pem
```

2.3 Extract Both Key and Certificate

```
# Extract everything to single file
openssl pkcs12 -in cert.pfx -out all.pem -nodes

# Extract CA certificates
openssl pkcs12 -in cert.pfx -cacerts -out ca.pem -nokeys
```

Tip

Common PFX Passwords to Try:
Empty password (just press Enter), `password`, `changeit`, `123456`, `mimikatz`

2.4 Create PFX from Key and Certificate

```
# Create PFX from separate files
openssl pkcs12 -export -out certificate.pfx \
    -inkey private.key -in certificate.crt

# Include CA chain
openssl pkcs12 -export -out certificate.pfx \
    -inkey private.key -in certificate.crt \
    -certfile ca-chain.crt
```

3 Certificate Operations

3.1 View Certificate Details

```
# View certificate in human-readable format
openssl x509 -in cert.pem -text -noout

# View specific fields
openssl x509 -in cert.pem -subject -noout
openssl x509 -in cert.pem -issuer -noout
openssl x509 -in cert.pem -dates -noout
openssl x509 -in cert.pem -serial -noout

# View Subject Alternative Names (SANs)
openssl x509 -in cert.pem -text -noout | grep -A1 "Subject Alternative
    Name"
```

3.2 Convert Certificate Formats

```
# PEM to DER
openssl x509 -in cert.pem -outform DER -out cert.der

# DER to PEM
openssl x509 -in cert.der -inform DER -out cert.pem

# PEM to PKCS7
openssl crl2pkcs7 -nocrl -certfile cert.pem -out cert.p7b

# PKCS7 to PEM
openssl pkcs7 -in cert.p7b -print_certs -out cert.pem
```

3.3 Verify Certificate

```
# Verify certificate against CA
openssl verify -CAfile ca.pem cert.pem

# Verify certificate chain
openssl verify -CAfile ca.pem -untrusted intermediate.pem cert.pem

# Check if key matches certificate
openssl x509 -noout -modulus -in cert.pem | openssl md5
openssl rsa -noout -modulus -in key.pem | openssl md5
# If MD5 hashes match, key and cert are paired
```

4 Key Operations

4.1 Generate Keys

```
# Generate RSA private key (2048-bit)
openssl genrsa -out private.key 2048

# Generate RSA key with passphrase
openssl genrsa -aes256 -out private.key 4096

# Generate EC private key
openssl ecparam -genkey -name secp384r1 -out ec_private.key

# Generate ED25519 key
openssl genpkey -algorithm ED25519 -out ed25519.key
```

4.2 View Key Details

```
# View RSA private key
openssl rsa -in private.key -text -noout

# View public key from private key
openssl rsa -in private.key -pubout -out public.key

# Check RSA key validity
openssl rsa -in private.key -check
```

4.3 Remove Passphrase from Key

```
# Remove encryption from private key
openssl rsa -in encrypted.key -out decrypted.key

# For EC keys
openssl ec -in encrypted_ec.key -out decrypted_ec.key
```

Warning

Removing passphrases creates unprotected keys. Handle with care and delete when no longer needed.

5 SSL/TLS Testing

5.1 Connect to SSL/TLS Services

```
# Basic SSL connection
openssl s_client -connect host:443

# Show full certificate chain
openssl s_client -connect host:443 -showcerts

# Specify SNI (Server Name Indication)
openssl s_client -connect host:443 -servername hostname

# Connect with specific TLS version
openssl s_client -connect host:443 -tls1_2
openssl s_client -connect host:443 -tls1_3
```

5.2 Certificate Reconnaissance

```
# Extract certificate from server
echo | openssl s_client -connect host:443 2>/dev/null | \
    openssl x509 -text -noout

# Get expiration date
echo | openssl s_client -connect host:443 2>/dev/null | \
    openssl x509 -noout -dates

# Extract SANs (find additional hostnames)
echo | openssl s_client -connect host:443 2>/dev/null | \
    openssl x509 -noout -text | grep -A1 "Subject Alternative"

# Save server certificate
echo | openssl s_client -connect host:443 2>/dev/null | \
    sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > server.crt
```

Tip

Subdomain Discovery: SANs often reveal internal hostnames, development servers, and other subdomains not publicly listed.

5.3 Test SSL/TLS Configuration

```
# Test specific cipher
openssl s_client -connect host:443 -cipher 'ECDHE-RSA-AES256-SHA'

# List supported ciphers
openssl ciphers -v 'ALL:COMPLEMENTOFALL'

# Test for SSLv3 (POODLE vulnerability)
openssl s_client -connect host:443 -ssl3

# Test for weak ciphers
openssl s_client -connect host:443 -cipher 'NULL,EXPORT,LOW,DES'

# Check certificate chain
openssl s_client -connect host:443 -CApath /etc/ssl/certs
```

5.4 Test STARTTLS Services

```
# SMTP
openssl s_client -connect mail.host:25 -starttls smtp

# IMAP
openssl s_client -connect mail.host:143 -starttls imap

# POP3
openssl s_client -connect mail.host:110 -starttls pop3

# FTP
openssl s_client -connect ftp.host:21 -starttls ftp

# LDAP
openssl s_client -connect ldap.host:389 -starttls ldap

# XMPP
openssl s_client -connect xmpp.host:5222 -starttls xmpp
```

6 Encryption & Decryption

6.1 Symmetric Encryption

```
# Encrypt file with AES-256-CBC
openssl enc -aes-256-cbc -salt -in plaintext.txt -out encrypted.enc

# Decrypt file
openssl enc -aes-256-cbc -d -in encrypted.enc -out plaintext.txt

# Encrypt with base64 encoding
openssl enc -aes-256-cbc -a -salt -in file.txt -out file.enc

# Specify password on command line (not recommended)
openssl enc -aes-256-cbc -salt -in file.txt -out file.enc -k "password"

# Use PBKDF2 key derivation (more secure)
openssl enc -aes-256-cbc -salt -pbkdf2 -in file.txt -out file.enc
```

6.2 Asymmetric Encryption

```
# Encrypt with public key
openssl rsautl -encrypt -pubin -inkey public.key \
    -in plaintext.txt -out encrypted.bin

# Decrypt with private key
openssl rsautl -decrypt -inkey private.key \
    -in encrypted.bin -out plaintext.txt

# Using pkeyutl (recommended for newer OpenSSL)
openssl pkeyutl -encrypt -pubin -inkey public.key \
    -in plaintext.txt -out encrypted.bin
```

7 Hashing Operations

7.1 # Generate Hashes

```
# Common hash algorithms
openssl dgst -md5 file.txt
openssl dgst -sha1 file.txt
openssl dgst -sha256 file.txt
openssl dgst -sha512 file.txt

# Hash a string
echo -n "password" | openssl dgst -sha256

# Output hash only (no filename)
openssl dgst -sha256 file.txt | awk '{print $2}'

# Binary output
openssl dgst -sha256 -binary file.txt > hash.bin
```

7.2 HMAC (Keyed Hashing)

```
# HMAC-SHA256
openssl dgst -sha256 -hmac "secret_key" file.txt

# HMAC from string
echo -n "message" | openssl dgst -sha256 -hmac "key"
```

8 </> Base64 Operations

```
# Encode to base64
openssl base64 -in file.bin -out file.b64
echo -n "text" | openssl base64

# Decode from base64
openssl base64 -d -in file.b64 -out file.bin
echo "dGV4dA==" | openssl base64 -d

# Encode without line breaks
openssl base64 -A -in file.bin
```

9 Password & Random Generation

9.1 Generate Password Hashes

```
# Generate Unix crypt hash
openssl passwd -1 "password"            # MD5
openssl passwd -5 "password"            # SHA-256
openssl passwd -6 "password"            # SHA-512

# With custom salt
openssl passwd -6 -salt "customsalt" "password"

# Apache htpasswd format
openssl passwd -apr1 "password"
```

Tip

Privilege Escalation: Use `openssl passwd` to generate hashes for `/etc/passwd` or `/etc/shadow` injection attacks.

9.2 Generate Random Data

```
# Generate random bytes (hex)
openssl rand -hex 32

# Generate random bytes (base64)
openssl rand -base64 32

# Generate random file
openssl rand -out random.bin 256
```

10 CSR Operations

```
# Generate CSR with new key
openssl req -new -newkey rsa:2048 -nodes \
    -keyout private.key -out request.csr

# Generate CSR from existing key
openssl req -new -key private.key -out request.csr

# View CSR details
openssl req -in request.csr -text -noout

# Verify CSR
openssl req -in request.csr -verify
```

11 Self-Signed Certificates

```
# Generate self-signed certificate (1 year)
openssl req -x509 -newkey rsa:4096 -keyout key.pem \
    -out cert.pem -days 365 -nodes

# One-liner with subject
openssl req -x509 -newkey rsa:2048 -keyout key.pem \
    -out cert.pem -days 365 -nodes \
    -subj "/CN=localhost/O=Test/C=US"

# From existing key
openssl req -x509 -key private.key -out cert.pem -days 365
```

Info

Self-signed certs are useful for testing HTTPS interception, creating rogue access points, or setting up C2 infrastructure.

12 Common Options Reference

Option	Description
`-in <file>`	Input file
`-out <file>`	Output file
`-text`	Output in human-readable text
`-noout`	Prevent output of encoded version
`-nodes`	No DES encryption (unencrypted key)
`-nocerts`	Don't output certificates
`-nokeys`	Don't output private keys
`-passin pass:<pwd>`	Input password
`-passout pass:<pwd>`	Output password
`-inform DER/PEM`	Input format
`-outform DER/PEM`	Output format
`-CAfile <file>`	CA certificate file
`-verify`	Verify signature/certificate

13 HTB Common Scenarios

13.1 Windows Certificate Auth (Evil-WinRM)

```
# Extract from PFX found on target
openssl pkcs12 -in user_auth.pfx -nocerts -out user.key -nodes
openssl pkcs12 -in user_auth.pfx -nokeys -out user.crt

# Connect with Evil-WinRM
evil-winrm -i target.htb -c user.crt -k user.key -S
```

13.2 SSH Key from Certificate

```
# Extract key for SSH authentication
openssl pkcs12 -in ssh_cert.pfx -nocerts -out id_rsa -nodes
chmod 600 id_rsa
ssh -i id_rsa user@target
```

13.3 ADCS Reconnaissance

```
# Examine certificate for ADCS information
openssl x509 -in cert.pem -text -noout | grep -A5 "Issuer"
openssl x509 -in cert.pem -text -noout | grep -A10 "Extensions"

# Check for User Principal Name (UPN)
openssl x509 -in cert.pem -text -noout | grep -i "principal"
```

14 Quick Reference Card

Essential Commands

```
openssl pkcs12 -in file.pfx -nocerts -out key.pem -nodes
openssl pkcs12 -in file.pfx -nokeys -out cert.pem
openssl x509 -in cert.pem -text -noout
openssl s_client -connect host:443
openssl dgst -sha256 file
openssl passwd -6 "password"
openssl rand -hex 32
```

Danger

Only use OpenSSL for authorized security testing. Extracting credentials from certificates or testing SSL configurations without permission is illegal.

15 Resources

- **Official Docs:** https://www.openssl.org/docs/
- **GitHub:** https://github.com/openssl/openssl
- **Man Pages:** `man openssl`, `man openssl-x509`
- **HackTheBox:** https://www.hackthebox.com/