

# Как просмотреть содержимое ключей и сертификатов SSL

## Как просмотреть содержимое ключей и сертификатов SSL

---

### Просмотр содержимого ключей и сертификатов

Мы можем подробно изучить содержимое всех созданных в OpenSSL файлов, а также при необходимости конвертировать их в другие форматы.

В следующих командах используются тестовые файлы со следующими именами:

- **rootCA.key** - ключ CA
- **rootCA.crt** - сертификат CA
- **mydomain.com.key** - ключ сервера
- **mydomain.com.csr** - запрос за подпись сертификата сайта
- **mydomain.com.crt** - сертификат сайта

Обратите внимание на расширения файлов — они могут отличаться от тех, которые используются в других инструкциях. Например, вместо **.key** и **.crt** может использоваться расширение **.pem**. Расширение файла не имеет особого значения кроме как служить подсказкой пользователю, что именно находится в этом файле. Это же самое касается и имён файлов — вы можете выбирать любые имена.

Все эти файлы являются текстовыми:

```
1 cat rootCA.key
```

Там мы увидим примерно следующее:

```
1 -----BEGIN PRIVATE KEY-----
2 MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQDJBKkr6XzzAcXD
3 eyDQdvB0SWE2F13nqlX/c2RgqMgScXtgidEzOu9ms3Krju5UKLokkQJrZFPMtiIL
4 MuPJFdYjVyfkfnqlZiouBVgJ60s8NQBBi8KnyyAoJCIFdASoW4Kv5C5LT8pX9eRa
5 /huJaRJL5XsFUGnTOLvW2ZLN52iAux9CoZlmH6ZF4nuQpblwN0MHULAhze52VNFT
6 .....
7 .....
8 .....
9 .....
10 .....
```

11 -----END PRIVATE KEY-----

12



```
mial@HackWare:~/test
Файл Правка Вид Поиск Терминал Справка
[mial@HackWare ~]$ cd test/
[mial@HackWare test]$ cat rootCA.key
-----BEGIN PRIVATE KEY-----
MIIEJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQDJBKkr6XzzAcXD
eyDQdvB0SWE2F13nqlX/c2RgqMgScXtgidEzOu9ms3Krju5UKLokkQJrZFPMTiIL
MuPJFdyjVyfkfnqlZiouBVgJ60s8NQBBi8KnyyAoJCIFdASoW4Kv5C5LT8pX9eRa
/huJaRjL5XsFUGnTOLvW2ZLN52iAux9CoZlmH6ZF4nuQpblwN0MHULAhze52VNFT
THP8JoAywgXrpXxwbs+5jeW2+RKAKnPsfcI40GJHdHiNiMwCfcEvUjwz7lEW2/r4
edICYVRDaf/9DmbQRHbFctvD3CGoVxWxt07Z6RizPSE+Hdo1scr9uWurmHX14n6
T8jLjhG/Pd9xFYz6E5unXmUIEBRQH0T7qt9HECpIWniezeyYzbLyrfMYH0wQ/r8x
jy2uGPF7jy8CGvA3DVHVNmqUXrhJB3bkbycAmJkRMbpnojEa5c0h1tcV8r8mMTxa
0YxN0iVTAuIMAR5qHlQ537vXGvA1PlslyTSeZivGNKB7Z8MUHjXXplqrWMGgwoAC
i0+L/qMYTCxCrWtr8FOUv9sEhIJGgiWqyx9qLlY9jgcbR1s51+6iL9uRJwQJdyFD
4UGkyjhyJ+h5mkhkhANVcfqq8gvNCPAXp3XIDFcXZADcFA7YdaQcb3GZNQpJb4
jg0cpK23eUSzkpfWeerfsu0BLcxQ6QIDAQABAOICAB+GcZmHZqcdsgw0NTM8U/2I
vIwk+dkpJ4+GY0mdbrMlxPmg44Qrs/xJTESHh+z+7iJp830wIMxMDA7BjmOs3wZv
1rvI8icaXKcXA3WaxxPMJBwTJaX2/zYMZKGCTnCw2VAIcxIwk0U0NPnoPB1SsyHf
/WkZ8V7E2QCa4qEUF5mIvyBOA6a2c/cly8c1lm143TQ6fUTv7OP17VYo+U+kIhrT
vuXz8A4Z8s3g6RPHABXfKqSWHG2YkLEWHO/lckk5u6BWP0bNqz3whe2N/Ve00A7S
gukG5yzmb9H7H/gzf9XMwoQ6YzDdk5gYkCwTtnMRTQp5l98zp2/4R5VQCeeSgkm6
fSKQBDWgQCsDv8CXFltSGPjQcK/cI8iR+frriBazYc27YwlqbiMzkvvccPPqr4lg
8yUUPwj6qjqLoT/TfPt+Y8f4WcDRwdIZMBKkHJd4c6P9Rt4hw6PKZXfBXr453NVG
9cBeMF+NXSPjtj3oo4RH2gb7+dprBwidHX/NCvYXn688fjOqxckjs0K1/Gu42L6s
jpBm0c8AHiSRGIPun4RAonghgghMm4TD+iK/0+bNEc0osUk0KY50kuXQesZB+BP3
-----END PRIVATE KEY-----
```

Если вам эти строки кажутся знакомыми на кодировку **Base64**, то вы совершенно правы — это она и есть. (Смотрите также [«Как быстро узнать и преобразовать кодировку»](#)). Этот формат, называемый форматом PEM, расшифровывается как Privacy Enhanced Mail.

PEM — это текстовое представление реального двоичного ключа или сертификата в формате DER. Представляет собой двоичного формата DER в кодировке base64 и с дополнительными строками «-----BEGIN PRIVATE KEY-----», «-----BEGIN CERTIFICATE-----» и другими в начале файла и строками «-----END PRIVATE KEY-----», «-----END CERTIFICATE-----» в конце файла.

Мы можем хранить двоичную версию файла только с кодировкой DER, но наиболее распространенным способом является версия PEM.

Вы можете увидеть структуру приватного следующей командой:

```
1 openssl rsa -text -in rootCA.key
```

Опция `-in ИМЯ_ФАЙЛА` указывает имя файла ввода для чтения ключа или стандартный ввод, если эта опция не указана. Если ключ зашифрован, будет запрошен пароль.

Опция `-text` печатает различные компоненты открытого или закрытого ключа в виде простого текста в дополнение к закодированной версии.

Пример вывода:

```
1 RSA Private-Key: (4096 bit, 2 primes)
2 modulus:
3 00:c9:04:a9:2b:e9:7c:f3:01:c5:c3:7b:20:d0:76:
4 [...]
```

```

5   publicExponent: 65537 (0x10001)
6   privateExponent:
7       1f:86:71:99:87:66:a7:1d:b2:0c:34:35:33:3c:53:
8       [...]
9   prime1:
10      00:f0:af:82:a6:f1:40:85:ee:c0:77:cc:41:ce:11:
11      [...]
12   prime2:
13      00:d5:cf:03:c6:2a:01:79:9a:e3:1d:ec:1b:52:40:
14      [...]
15   exponent1:
16      00:d7:7e:ed:65:f7:9f:a3:cb:2e:bc:94:3f:5e:f8:
17      [...]
18   exponent2:
19      00:ae:a1:5e:db:c4:03:60:67:79:89:3f:07:31:ae:
20      [...]
21   coefficient:
22      00:e4:7d:de:4e:00:a0:8d:c4:5a:14:93:b6:7f:c9:
23      [...]
24   writing RSA key
25   -----BEGIN RSA PRIVATE KEY-----
26   [...]
27   -----END RSA PRIVATE KEY-----

```

Любой формат ключа на самом деле является контейнером для набора длинных чисел. Все остальные данные можно считать «шумом».

Закрытый ключ содержит: модуль (modulus), частный показатель (privateExponent), открытый показатель (publicExponent), простое число 1 (prime1), простое число 2 (prime2), показатель степени 1 (exponent1), показатель степени 2 (exponent2) и коэффициент (coefficient).

Открытый ключ содержит только модуль (modulus) и открытый показатель (publicExponent).

Вы можете извлечь из файла ключей публичный ключ:

```
1 openssl rsa -in rootCA.key -pubout -out rootCA-public.key
```

По умолчанию выводится закрытый ключ: с опцией **-pubout** вместо него будет выведен открытый ключ. Эта опция устанавливается автоматически, если ввод является открытым ключом.

Следующая команда покажет информацию о публичном ключе:

```
1 openssl rsa -text -in rootCA-public.key -pubin
```

По умолчанию из входного файла считывается закрытый ключ. Используемая в предыдущей команде опция **-pubin** делает так, что вместо приватного ключа читается открытый ключ.

Можно также добавить опцию **-noout** — с ней будет выведена та же самая информация, но не будет показана кодированная версия ключа.

```
1 openssl rsa -text -in rootCA-public.key -pubin -noout
```

С такими же опциями, но уже используя команду **req**, можно изучить содержимое запроса на подпись сертификата:

```
1 openssl req -in mydomain.com.csr -noout -text
```

```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl req -in mydomain.com.csr -noout -text
Certificate Request:
  Data:
    Version: 1 (0x0)
    Subject: C = US, ST = CA, O = "MyOrg, Inc.", CN = mydomain.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (4096 bit)
      Modulus:
        00:b1:ca:41:79:41:28:60:d1:08:d4:6a:26:7f:ee:
        b3:5b:c0:76:79:2b:21:e1:30:0a:6c:42:a0:1e:3e:
        a3:5c:30:95:3d:8c:c0:4a:f5:62:95:61:ea:b3:b5:
        68:3a:23:62:07:62:53:8a:99:05:26:74:48:b7:a1:
        77:f9:7f:79:4c:37:19:4f:a6:d5:a0:5e:b4:ba:54:
        0f:fa:27:74:7c:f4:1b:b8:34:bc:25:3b:80:2b:08:
        ab:cd:4c:95:f7:d2:c0:f1:2b:4f:20:90:cd:d8:54:
        d4:6a:0e:69:f5:ce:78:59:1e:08:f7:62:be:a8:3a:
        73:53:5b:6c:86:02:77:02:ea:f1:8a:b6:dd:cf:c4:
        59:50:2b:33:75:48:cc:a2:a9:68:8c:be:9a:06:b9:
        ac:6f:c7:a6:36:85:4c:d5:8c:3a:8b:6c:4e:cb:67:
        27:47:81:67:5d:e6:e0:0e:ba:99:5f:e5:65:c9:d4:
        06:4f:af:5c:2a:2e:30:70:60:eb:ca:85:49:95:14:
        00:cf:3e:25:3e:c9:48:8b:3b:96:12:22:1f:d8:17:
        dd:5d:01:38:f7:e2:0d:4e:ea:1e:5e:90:0a:e5:85:
        4c:7c:d8:96:4f:bb:76:95:ef:83:b2:f5:27:e5:64:
        f8:8b:fe:61:b2:9b:e8:52:8d:a0:f7:7d:90:63:4f:
        20:93:91:de:ae:3b:c0:ec:6c:e4:6b:3f:11:e7:38:
        d2:6b:fc:f9:e5:bd:82:01:00:3c:dc:cf:9e:a7:da:
        d1:b7:1c:49:d0:68:91:ab:18:f4:7a:4e:a3:2e:81:
        9e:83:d5:61:6f:65:17:41:9b:6b:d2:34:5e:d7:a1:
        2b:30:2e:c0:a2:f5:b6:1d:61:01:f8:64:d5:60:15:
        11:fb:ad:a9:c2:fe:79:82:88:cd:43:3c:48:5a:aa:
        13:62:f2:21:ff:c9:96:23:c7:21:97:23:56:bf:f7:
        d2:7f:92:4b:c0:e8:87:1e:6d:d9:9e:c5:1f:2f:c8:
        dc:cd:f2:74:d9:fc:70:da:ff:e5:12:f1:a4:3f:a7:
        57:da:58:d7:a2:d3:7e:d2:6e:1e:d0:19:c4:40:16:
        cc:14:16:b5:42:2e:d3:98:da:1d:48:6f:32:ae:ed:
        66:fb:b9:08:e2:b4:ca:a3:6b:a4:1f:bb:dc:16:43:
        92:e7:77:f9:54:ba:dc:36:b4:49:f3:85:ba:2a:46:
        6d:96:0d:a4:70:b7:80:2c:85:a1:e8:4d:b6:be:71:
        a5:ec:2e:57:99:eb:f7:69:77:98:50:20:16:27:84:
        73:c9:c8:68:e0:89:a4:17:e7:eb:4a:4a:d7:28:a1:
        46:da:0d:8f:65:75:73:6a:e8:fa:44:b4:6b:61:4c:
        b0:af:f3
      Exponent: 65537 (0x10001)
    Attributes:
      a0:00
  Signature Algorithm: sha256WithRSAEncryption
    16:54:66:b3:b3:20:55:1d:f4:9f:2c:28:bb:3b:e3:d0:fd:2c:
    5d:99:ef:0b:52:c7:55:ad:fe:fd:6f:fd:1a:bc:7c:3e:fb:07:
    b3:98:77:15:e4:88:cd:d1:39:11:c6:17:48:af:98:f4:bb:90:
    98:50:09:0a:5d:9a:ed:67:94:62:ea:30:b5:f6:fe:3b:b2:12:
    0c:c2:e1:2a:a0:d3:94:7f:f8:9e:d4:e0:35:31:60:ad:a6:54:
```

При создании SSL сертификата мы создали две пары ключей (корневые и для домена), то есть это файлы **rootCA.key** и **mydomain.com.key**, но по своей технической сути они идентичны. Что касается сертификатов, которых у нас тоже два (**rootCA.crt** и **mydomain.com.crt**), то по своей природе они не являются одинаковыми: корневой сертификат является самоподписанным, а сертификат домена подписан приватным корневым ключом. Информацию о содержимом сертификатов можно посмотреть командой **x509** (остальные опции нам уже знакомы):

- 1 `openssl x509 -in rootCA.crt -noout -text`
- 2 `openssl x509 -in mydomain.com.crt -text -noout`





```
mial@HackWare:~  
Файл Правка Вид Поиск Терминал Справка  
1 s:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3  
i:O = Digital Signature Trust Co., CN = DST Root CA X3  
-----BEGIN CERTIFICATE-----  
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOfc2oLheynCDANBgkqhkiG9w0BAQsFADA/  
MSQwIgYDVQQKEExtEaWdpdGFsIFNpZ25hdHVyZSBUCnVzdCBDbY4xFzAVBgNVBAMT  
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTE2MDMxNzE2NDA0Nlow  
SjELMAkGA1UEBhMCVVMxIjAUBgNVBAoTUDUxldCdzIEVudY3J5cHQxIzAhBgNVBAMT  
GkxldCdzIEVudY3J5cHQxIzAhBgNVBAMTGDkxldCdzIEVudY3J5cHQxIzAhBgNV  
AQ8AMIBCBGKCAQEAAnMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF  
q6meNQH7Y7LEqXGihC6PjdeTm86dicbp5gWaf15Gan/PQeGdxyGkOlZHP/uaZ6WA8  
SMx+yk13EiSdRxta67nsHjcaHJyse6cf6s5K671B5TaYucv9bTyWan8jKkKQDIZ0  
Z8h/pZq4UmeUEZ9l6YKH9v6D1b2honzhT+Xhq+w3Brvaw2VFfn3EK6BlspkENnWA  
a6xK8xuQsXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvj  
/PIzark5McWvxI0NHQWM6r6hCm21AvA2H3DkwIDAQABo4IBftCCAXkwEgYDVR0T  
AQH/BAgwbGEB/wIBADAQBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEdzBxMDIG  
CCsGAQUBFzABhizodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv  
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k  
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUXKexpHsscfrb4UuQdf/EFWCFiRAw  
VAYDVR0gBE0wSzAIBgZnqQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC  
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNLbmNyeXB0Lm9yZyZaA8BgNVHR8ENTAz  
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWdNDUKwu  
Y3JsMB0GA1UdDgQWBBS0SmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF  
AAOCAQEATPXEfnjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo  
uM2VcGf196S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/  
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUGwu  
X4Po1QYz+3dszkDqMp4fklxBwXRsw10KXzPMTZ+sOPaveyxindmjkW8lGy+QsRlG  
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6  
KOqkqm57TH2H3eDJAKSnh6/DNFu0Qg==  
-----END CERTIFICATE-----  
---  
Server certificate  
subject=CN = w-e-b.site  
  
issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3  
---  
No client certificate CA names sent  
Peer signing digest: SHA256  
Peer signature type: RSA-PSS  
Server Temp Key: X25519, 253 bits  
---  
SSL handshake has read 3609 bytes and written 392 bytes  
Verification: OK  
---  
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384  
Server public key is 4096 bit  
Secure Renegotiation IS NOT supported  
Compression: NONE  
Expansion: NONE  
No ALPN negotiated  
Early data was not sent  
Verify return code: 0 (ok)
```

К примеру, сертификат сайта w-e-b.site я сохранил в файл **w-e-b.site.crt**, для просмотра информации о нём:

```
1 openssl x509 -in w-e-b.site.crt -text -noout
```

Пример вывода:

```
mial@HackWare:~/test
Файл Правка Вид Поиск Терминал Справка
[mial@HackWare test]$ openssl x509 -in w-e-b.site.crt -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      03:ad:15:e1:34:f5:a4:d7:6f:ef:36:9d:86:a4:c1:ed:5b:27
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
    Validity
      Not Before: May 12 03:42:46 2020 GMT
      Not After : Aug 10 03:42:46 2020 GMT
    Subject: CN = w-e-b.site
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (4096 bit)
      Modulus:
        00:ed:f5:b5:f9:2f:0e:1f:b9:23:21:f1:4b:a3:66:
        26:c2:9c:84:d4:be:aa:34:6b:b6:f7:1b:8e:a9:7e:
        20:0b:76:36:cf:14:73:65:55:be:9f:9d:3d:1b:60:
        c3:64:ea:74:64:ab:97:8b:3d:7b:a1:37:ae:74:c8:
        0e:e7:ce:7c:e7:0e:c2:a7:7e:a4:c7:35:0f:c2:68:
        0e:45:15:b1:7f:69:6a:6a:0b:a9:06:28:86:84:c1:
        82:bf:24:dc:d7:cf:0e:40:a5:68:a8:85:5e:80:d0:
        91:2e:9f:b4:cb:5b:98:9f:17:95:24:c3:dc:6a:9d:
        4d:ec:1b:36:51:0c:44:26:72:78:68:2f:03:65:dd:
        8f:94:fa:9a:eb:a8:3d:ee:77:93:b7:02:7c:e8:47:
        dd:af:6c:fa:80:fc:e9:83:6d:22:af:01:eb:2c:75:
        df:ec:72:4e:0b:80:13:a9:65:fd:3e:b7:1c:ec:0a:
        d8:41:b4:a9:4b:04:7e:eb:cb:00:1c:eb:c3:63:0c:
        0f:f0:91:f5:b6:20:9c:9d:2f:89:76:b5:49:eb:50:
        31:e8:bc:f6:6f:cf:84:92:5e:df:8b:ee:47:4d:51:
        81:db:24:50:6d:da:9c:6d:72:8e:2d:c2:8e:87:c3:
        90:88:3c:74:92:70:35:49:95:34:4a:c3:b0:25:45:
        e4:63:21:bf:b3:97:3d:52:4b:42:cf:e0:b9:9a:b2:
        80:4e:dc:4a:84:37:68:67:b8:ea:30:97:1e:1a:e5:
        7a:da:92:f3:c3:bc:84:f4:a1:c8:4f:65:47:2e:3f:
        cd:d8:63:b0:d9:90:2d:df:39:4f:ba:a9:ff:f2:f4:
        80:b0:35:16:04:c5:05:18:6c:98:df:d8:cf:ed:a7:
        74:85:0a:26:86:4c:50:54:18:1a:03:04:5b:ac:b0:
        e1:14:e9:b2:df:ff:65:3b:68:29:d7:45:d1:7b:54:
        e5:33:aa:4a:e5:64:9c:59:eb:01:1c:a2:35:81:71:
        31:ed:33:9d:f2:80:82:c2:3b:a0:0c:df:cf:4f:eb:
        63:ec:dd:d4:b2:c2:b0:c3:b2:8d:d1:14:cc:8d:76:
        2f:25:0a:0a:50:ab:5c:99:8e:18:0f:72:a0:a3:9c:
        15:06:71:d8:4b:d3:0f:0c:f2:97:0a:77:fd:92:50:
        c6:89:a7:b1:c0:1a:ab:fb:f0:f7:4e:f5:0d:8c:47:
        59:90:6c:38:84:51:d9:a2:3c:12:e0:93:1e:a1:89:
        b4:c8:05:ae:00:cc:c5:7c:c0:fb:a2:02:dc:28:2a:
        e2:57:20:ba:8a:7d:42:bf:27:7e:93:02:18:68:ca:
        a8:8a:a5:38:e7:4b:52:9b:f4:68:69:b4:d3:68:50:
        6c:a3:79
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
```



```

    Digital Signature, Key Encipherment
X509v3 Extended Key Usage:
    TLS Web Server Authentication, TLS Web Client Authentication
X509v3 Basic Constraints: critical
    CA:FALSE
X509v3 Subject Key Identifier:
    12:AF:9A:BC:DA:63:7D:5A:35:14:E6:8F:1D:B1:70:ED:83:E0:81:AC
X509v3 Authority Key Identifier:
    keyid:A8:4A:6A:63:04:7D:DD:BA:E6:D1:39:B7:A6:45:65:EF:F3:A8:EC:A1

Authority Information Access:
    OCSP - URI:http://ocsp.int-x3.letsencrypt.org
    CA Issuers - URI:http://cert.int-x3.letsencrypt.org/

X509v3 Subject Alternative Name:
    DNS:w-e-b.site
X509v3 Certificate Policies:
    Policy: 2.23.140.1.2.1
    Policy: 1.3.6.1.4.1.44947.1.1.1
    CPS: http://cps.letsencrypt.org

CT Precertificate SCTs:
    Signed Certificate Timestamp:
      Version   : v1 (0x0)
      Log ID    : E7:12:F2:B0:37:7E:1A:62:FB:8E:C9:0C:61:84:F1:EA:
                  7B:37:CB:56:1D:11:26:5B:F3:E0:F3:4B:F2:41:54:6E
      Timestamp : May 12 04:42:46.716 2020 GMT
      Extensions: none
      Signature : ecdsa-with-SHA256
                  30:43:02:1F:2E:54:3C:F3:6C:97:01:B9:F2:FC:FD:42:
                  66:9F:C7:DE:5A:D8:E4:B9:FE:E0:AD:5B:A8:1A:04:A4:
                  53:99:2D:02:20:59:DD:68:95:C9:E1:4C:45:A9:44:2F:
                  B2:E3:2F:36:AB:B7:B0:1E:0F:BD:AD:53:21:E7:03:ED:
                  14:F8:BB:AC:F4
    Signed Certificate Timestamp:
      Version   : v1 (0x0)
      Log ID    : B2:1E:05:CC:8B:A2:CD:8A:20:4E:87:66:F9:2B:B9:8A:
                  25:20:67:6B:DA:FA:70:E7:B2:49:53:2D:EF:8B:90:5E
      Timestamp : May 12 04:42:46.701 2020 GMT
      Extensions: none
      Signature : ecdsa-with-SHA256
                  30:45:02:20:71:99:B6:8B:B0:B8:DE:F6:05:C4:8E:34:
                  18:72:BB:4C:64:E0:2C:2C:F0:75:1F:74:31:20:0A:49:
                  B8:C6:44:08:02:21:00:F3:07:3F:3F:25:98:58:8D:E0:
                  E5:60:75:77:67:AD:D2:F5:A3:32:00:2C:A9:DF:9F:01:
                  AB:34:5A:7B:E6:E9:24
Signature Algorithm: sha256WithRSAEncryption
    8a:87:c9:44:e1:d0:f2:23:a5:93:91:b7:f3:7c:5e:51:7f:70:
    a7:b4:a3:7f:10:22:2d:21:2b:c1:ae:d6:ec:b7:b1:44:88:02:
    5b:63:90:24:ad:80:1a:8b:9e:39:50:d5:2c:6f:46:74:cc:f1:
    11:4a:a8:bf:91:69:ce:fc:4d:7e:5f:4f:ae:c5:24:e8:22:21:
    12:33:b1:f8:a4:5f:81:57:54:a7:7c:a2:67:a0:29:a8:38:e3:
    57:d1:79:33:9f:65:09:da:dd:a8:8a:0f:10:d9:cc:1f:5d:04:
    69:be:f7:68:47:26:08:15:b3:eb:32:ab:d3:12:eb:40:ca:de:
    fe:d9:9b:14:97:23:5b:e7:2a:98:34:69:71:6d:76:de:f4:2e:
    ..

```

**Issuer** указывает на организацию, которая выдала (подписала) сертификат:

1 Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3

**Validity** — срок действия сертификата:

1 Validity

2 Not Before: May 12 03:42:46 2020 GMT

3 Not After : Aug 10 03:42:46 2020 GMT

**Subject: CN** — домен (IP адрес) для которого предназначен сертификат:

1 Subject: CN = w-e-b.site

Поддомены в группе X509v3 extensions → X509v3 Subject Alternative Name (подробности чуть позже):

```
1 X509v3 Subject Alternative Name:
2 DNS:w-e-b.site, DNS:www.w-e-b.site
```

Теперь бегло рассмотрим расширения X.509.

Расширение **Basic Constraints** используется для маркировки сертификатов как принадлежащих ЦС, давая им возможность подписывать другие сертификаты. В сертификатах, отличных от CA, это расширение будет либо пропущено, либо будет установлено значение **CA**, равное **FALSE**. Это расширение является критическим, что означает, что все программные сертификаты должны понимать его значение.

```
1 X509v3 Basic Constraints: critical
2 CA:FALSE
```

Расширения **Key Usage (KU)** и **Extended Key Usage (EKU)** ограничивают возможности использования сертификата. Если эти расширения присутствуют, то разрешены только перечисленные варианты использования. Если расширения отсутствуют, ограничений на использование нет. То, что вы видите в этом примере, типично для сертификата веб-сервера, который, например, не позволяет подписывать код:

```
1 X509v3 Key Usage: critical
2 Digital Signature, Key Encipherment
3 X509v3 Extended Key Usage:
4 TLS Web Server Authentication, TLS Web Client Authentication
```

Расширение **CRL Distribution Points** перечисляет адреса, по которым можно найти информацию о списке отзыва сертификатов (CRL) ЦС. Эта информация важна в случаях, когда сертификаты необходимо отозвать. CRL — это подписанные СА списки отозванных сертификатов, публикуемые через регулярные промежутки времени (например, семь дней).

```
1 X509v3 CRL Distribution Points:
2 Full Name:
3 URI:http://crl.starfieldtech.com/sfs3-20.crl
```

Примечание: возможно, вы заметили, что местоположение CRL не использует защищённый сервер, и вам может быть интересно, является ли ссылка небезопасной. Не является. Поскольку каждый CRL подписан центром сертификации, который его выпустил, браузеры могут проверить его целостность. В том же случае, если бы CRL были доступны по TLS (адрес включал бы в себя протокол HTTPS), то браузеры могут столкнуться с проблемой «курицы и яйца», в которой они хотят проверить статус отзыва сертификата, используемого сервером, доставляющим сам CRL!

Расширение **Certificate Policies** используется для указания политики, в соответствии с которой был выпущен сертификат. Например, именно здесь можно найти индикаторы расширенной проверки (EV). Индикаторы представлены в форме уникальных идентификаторов объектов (OID) и являются уникальными для выдающего ЦС. Кроме того, это расширение часто содержит один или несколько пунктов CPS, которые обычно являются веб-страницами или документами PDF.

```
X509v3 Certificate Policies:
1 Policy: 2.23.140.1.2.1
2 Policy: 1.3.6.1.4.1.44947.1.1.1
```

3 CPS: <http://cps.letsencrypt.org>

4

Расширение **Authority Information Access (AIA)** обычно содержит две важные части информации. Во-первых, он перечисляет адрес ответчика CA OSCP, который можно использовать для проверки отзыва сертификатов в режиме реального времени. Расширение также может содержать ссылку, где находится сертификат эмитента (следующий сертификат в цепочке). В наши дни серверные сертификаты редко подписываются непосредственно доверенными корневыми сертификатами, а это означает, что пользователи должны включать в свою конфигурацию один или несколько промежуточных сертификатов. Ошибки легко сделать, и сертификаты будут признаны недействительными. Некоторые клиенты (например, Internet Explorer) будут использовать информацию, представленную в этом расширении для исправления неполной цепочки сертификатов, но многие клиенты этого не сделают.

1 Authority Information Access:

2 OSCP - URI:<http://ocsp.int-x3.letsencrypt.org>

3 CA Issuers - URI:<http://cert.int-x3.letsencrypt.org>

Расширения **Subject Key Identifier** и **Authority Key Identifier** устанавливают уникальные идентификаторы ключа субъекта и ключа авторизации соответственно. Значение, указанное в расширении **Authority Key Identifier** сертификата, должно соответствовать значению, указанному в расширении **Subject Key Identifier** в выдающем сертификате. Эта информация очень полезна в процессе построения пути сертификации, когда клиент пытается найти все возможные пути от конечного (серверного) сертификата до доверенного корня. Центры сертификации часто используют один закрытый ключ с несколькими сертификатами, и это поле позволяет программному обеспечению надёжно определять, какой сертификат может быть сопоставлен с каким ключом. В реальном мире многие цепочки сертификатов, предоставляемые серверами, недействительны, но этот факт часто остаётся незамеченным, поскольку браузеры могут находить альтернативные пути доверия.

1 X509v3 Subject Key Identifier:

2 12:AF:9A:BC:DA:63:7D:5A:35:14:E6:8F:1D:B1:70:ED:83:E0:81:AC

3 X509v3 Authority Key Identifier:

4 keyid:A8:4A:6A:63:04:7D:DD:BA:E6:D1:39:B7:A6:45:65:EF:F3:A8:EC:A1

Наконец, расширение **Subject Alternative Name** используется для перечисления всех имен хостов, для которых действителен сертификат. Это расширение раньше было необязательным; если его нет, клиенты возвращаются к использованию информации, представленной в **Common Name (CN)**, которое является частью поля «**Subject**». Если расширение присутствует, то содержимое поля **CN** игнорируется во время проверки.

1 X509v3 Subject Alternative Name:

2 DNS:w-e-b.site, DNS:www.w-e-b.site

Рассмотрим опции команды **x509**, которые позволяют извлечь разнообразную информацию из сертификатов.

Чтобы показать издателя сертификата используйте опцию **-issuer**:

1 openssl x509 -in w-e-b.site.crt -noout -issuer

Пример вывода:

1 issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3

Опция **-fingerprint** вычисляет и выводит дайджест DER-кодированной версии всего сертификата. Это обычно называют «отпечатком». Из-за характера дайджестов сообщений, отпечаток сертификата является уникальным для этого сертификата, и два сертификата с одинаковым отпечатком могут считаться одинаковыми. Для сертификатов обычно не нужно сверять сертификаты по отпечаткам, но это имеет смысл при использовании самоподписанных сертификатов (например, получении [сертификата для VNC сессии](#), когда нет другого способа проверить, что сертификат не был подменён при пересылке). В этом случае можно сверить отпечаток сертификата, например, по телефону или электронной почте.

Для показа отпечатка сертификата:

```
1 openssl x509 -in w-e-b.site.crt -noout -fingerprint
```

Пример вывода:

```
1 SHA1 Fingerprint=43:4E:55:5B:27:09:33:00:3F:43:B0:B7:B2:5E:96:D5:10:42:3B:44
```

Чтобы вывести сертификат в виде строки [символов в стиле C — char](#), используйте опцию **-C**:

```
1 openssl x509 -in w-e-b.site.crt -noout -C
```

Чтобы вывести расширения сертификата в текстовой форме, используйте опцию **-ext**. Несколько расширений можно перечислить через запятую, например **"subjectAltName,subjectKeyIdentifier"**. Чтобы посмотреть весь список расширений:

```
1 man x509v3_config
```

Пример команды для вывода альтернативных имён в домене:

```
1 openssl x509 -in suip.biz.cert -noout -ext subjectAltName
```

Пример информации об альтернативных именах домена:

```
1 X509v3 Subject Alternative Name:
2      DNS:suip.biz, DNS:www.suip.biz
```

Для вывода почтовых адресов, содержащихся в сертификате, используйте опцию **-email**. Адреса электронной почты могут отсутствовать в сертификате.

```
1 openssl x509 -in w-e-b.site.crt -noout -email
```

Для вывода адресов респондентов OCSP, если они присутствуют в сертификате, укажите опцию **-ocsp\_uri**:

```
1 openssl x509 -in w-e-b.site.crt -noout -ocsp_uri
```

Пример вывода:

```
1 http://ocsp.int-x3.letsencrypt.org
```

Для показа дат из сертификата имеются следующие опции:

- **-startdate**: Распечатывает дату начала сертификата, то есть дату notBefore (не ранее).
- **-enddate**: Распечатывает дату истечения срока действия сертификата, то есть дату notAfter (не позднее).
- **-dates**: Распечатывает даты начала и окончания срока действия сертификата.

Для вывода имени subject укажите опцию **-subject**:

```
1 openssl x509 -in w-e-b.site.crt -noout -subject
```

Пример вывода:

```
1 subject=CN = w-e-b.site
```

Чтобы показать имя subject сертификата в форме RFC2253 используйте сочетание опций -**subject -nameopt RFC2253**:

```
1 openssl x509 -in w-e-b.site.crt -noout -subject -nameopt RFC2253
```

Пример вывода:

```
1 subject=CN=w-e-b.site
```

Пример вывода имени subject сертификата в форме схемы на терминале, поддерживающем UTF8:

```
1 openssl x509 -in w-e-b.site.crt -noout -subject -nameopt oneline,-esc_msb
```

Опция **-serial** выводит серийный номер:

```
1 openssl x509 -in w-e-b.site.crt -noout -serial
```

Пример вывода:

```
1 serial=03AD15E134F5A4D76FEF369D86A4C1ED5B27
```

Чтобы извлечь публичный ключ из сертификата используйте опцию **-pubkey**:

```
1 openssl x509 -in w-e-b.site.crt -noout -pubkey
```

Для глубокого понимания OpenSSL смотрите также полное руководство: [«OpenSSL: принципы работы, создание сертификатов, аудит»](#).

## Связанные статьи:

- [Как создать сертификаты SSL \(TLS\)](#) (44.9%)
- [Как верифицировать сертификат SSL](#) (44.9%)
- [Как включить SMTPS \(465\) postfix в Linux](#) (37.7%)
- [Исправление смещенного контента HTTPS в системе пользовательского поиска Google](#) (37.7%)
- [Установка последних версий Node.js и NPM в Linux](#) (37.7%)
- [Практические примеры использования команды find в Linux](#) (RANDOM - 12.4%)

## Как создать сертификаты SSL (TLS)

### Создание корневого приватного ключа

Внимание: этот ключ используется для подписи запросов сертификатов, любой, кто получил этот ключ, может подписывать сертификаты от вашего имени, поэтому храните его в безопасном месте:

Генерация приватного ключа RSA используя параметры по умолчанию (ключ будет сохранён в файл с именем **rootCA.key**):

```
1 openssl genpkey -algorithm RSA -out rootCA.key
```

Опция **-out** указывает на имя файла для сохранения, без этой опции файл будет выведен в стандартный вывод (на экран). Имя выходного файла не должно совпадать с именем входного файла.

Для безопасности ключа его следует защитить паролем. Генерация приватного ключа RSA используя 128-битное AES шифрование (**-aes-128-cbc**) и парольную фразу "hello" (**-pass pass:hello**):

```
1 openssl genpkey -algorithm RSA -out rootCA.key -aes-128-cbc -pass pass:hello
```

Конечно, опцию **-pass pass:hello** можно не указывать, тогда вам будет предложено ввести пароль во время генерации ключа.

Список поддерживаемых симметричных алгоритмов шифрования приватного ключа можно узнать в документации (раздел **SUPPORTED CIPHERS**):

```
1 man enc
```

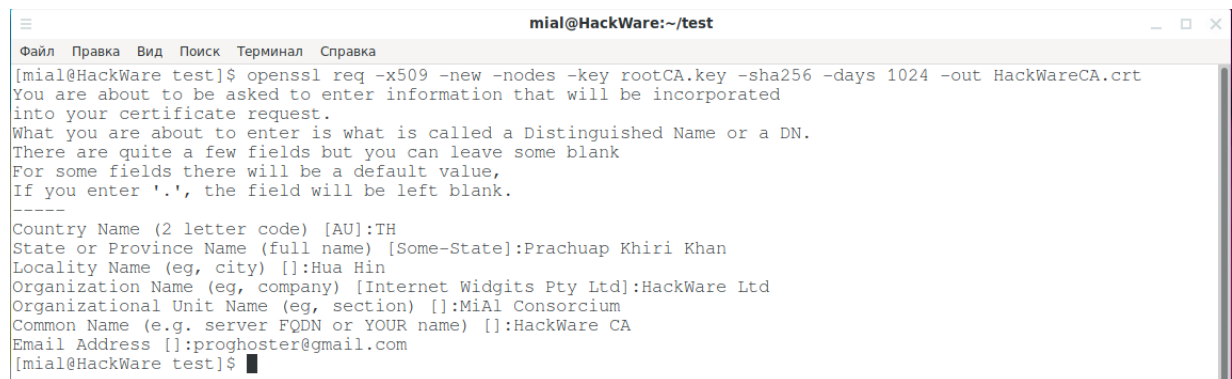
Если для генерируемого ключа не указано количество бит, то по умолчанию используется 2048, вы можете указать другое количество бит с помощью команды вида (будет создан 4096-битный ключ):

```
1 openssl genpkey -algorithm RSA -out rootCA.key -aes-128-cbc -pkeyopt rsa_keygen_b
```

## Создание самоподписанного корневого сертификата

```
1 openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt
```

Здесь мы использовали наш корневой ключ для создания корневого сертификата (файл **rootCA.crt**), который должен распространяться на всех компьютерах, которые нам доверяют. А приватный ключ (файл **rootCA.key**) должен быть секретным, поскольку он будет использоваться для подписи сертификатов серверов.



Создание сертификатов (делается для каждого домена) включает в себя несколько этапов. Эту процедуру необходимо выполнить для каждого домена/сервера, которым требуется доверенный сертификат от нашего ЦС.

## Создание приватного ключа сертификата

```
1 openssl genpkey -algorithm RSA -out mydomain.com.key
```

Обратите внимание, что это та же самая команда, которой мы создавали пару приватный-публичный ключ Центра Сертификации (изменено только имя файла с ключами).



# Создание файла с запросом на подпись сертификата (csr)

Получив закрытый ключ, вы можете приступить к созданию запроса на подпись сертификата — Certificate Signing Request (CSR). Это официальный запрос к CA о подписании сертификата, который содержит открытый ключ объекта, запрашивающего сертификат, и некоторую информацию об объекте. Все эти данные будут частью сертификата. CSR всегда подписывается закрытым ключом, соответствующим открытому ключу, который он несёт.

Создание CSR обычно представляет собой интерактивный процесс, в ходе которого вы будете предоставлять элементы отличительного имени сертификата (вводить информацию о стране, городе, организации, email и т.д.). Внимательно прочитайте инструкции, предоставленные инструментом openssl; если вы хотите, чтобы поле было пустым, вы должны ввести одну точку (.) в строке, а не просто нажать «Enter». Если вы сделаете последнее, OpenSSL заполнит соответствующее поле CSR значением по умолчанию. (Такое поведение не имеет никакого смысла при использовании с конфигурацией OpenSSL по умолчанию, что и делают практически все. Это имеет смысл, когда вы осознаете, что можете изменить значения по умолчанию, либо изменив конфигурацию OpenSSL, либо предоставив свои собственные конфигурации в файлах).

В запросе на подпись сертификата вы указываете информацию для сертификата, который хотите сгенерировать. Этот запрос будет обработан владельцем корневого ключа (в данном случае вы его создали ранее) для генерации сертификата.

Важно: имейте в виду, что при создании запроса на подпись важно указать **Common Name**, предоставляющее IP-адрес или доменное имя для службы, в противном случае сертификат не может быть проверен.

Я опишу здесь два способа:

- Метод А (интерактивный)

Если вы создадите CSR таким способом, openssl задаст вам вопросы о сертификате, который необходимо сгенерировать, например, сведения об организации и **Common Name (CN)**, которое является веб-адресом, для которого вы создаёте сертификат, например, mydomain.com.

```
1 openssl req -new -key mydomain.com.key -out mydomain.com.csr
```

## Метод Б (в одну команду без запросов)

Этот метод генерирует тот же результат, что и метод А, но он подходит для использования в вашей автоматизации.

```
1 openssl req -new -sha256 -key mydomain.com.key -subj "/C=US/ST=CA/O=MyOrg, Inc./O=MyOrg, Inc."/ -out mydomain.com.csr
```

Если вам нужно передать дополнительную конфигурацию, вы можете использовать параметр **-config**, например, здесь я хочу добавить альтернативные имена в мой сертификат.

```
1 openssl req -new -sha256 -key mydomain.com.key -subj "/C=US/ST=CA/O=MyOrg, Inc./O=MyOrg, Inc." -config mydomain.com.cnf -out mydomain.com.csr
```

## Проверка содержимого CSR

После создания CSR используйте его, чтобы подписать собственный сертификат и/или отправить его в общедоступный центр сертификации и попросить его подписать сертификат. Оба подхода описаны в следующих разделах. Но прежде чем сделать это, неплохо бы ещё раз проверить правильность CSR. Это делается так:

```
1 openssl req -in mydomain.com.csr -noout -text
```

## Создание сертификата

Создайте сертификат, используя csr для mydomain.com, корневые ключ и сертификат CA.

Если вы устанавливаете сервер TLS для своего собственного использования, вы, вероятно, не хотите идти в ЦС для покупки публично доверенного сертификата. Намного проще использовать сертификат, подписанный вашим собственным CA. Если вы являетесь пользователем Firefox, при первом посещении веб-сайта вы можете создать исключение для сертификата, после которого сайт будет защищён так, как если бы он был защищён общедоступным сертификатом.

Если у вас уже есть CSR, создайте сертификат, используя следующую команду:

```
1 openssl x509 -req -in mydomain.com.csr -CA rootCA.crt -CAkey rootCA.key -CAcreate
```

В результате выполнения этих команд были созданы следующие файлы:

- **rootCA.key** — приватный ключ Центра Сертификации, должен храниться в секрете в CA
- **rootCA.crt** — публичный корневой сертификат Центра Сертификации — должен быть установлен на всех пользовательских устройствах
- **mydomain.com.key** — приватный ключ веб-сайта, должен храниться в секрете на сервере. Указывает в конфигурации виртуального хоста при настройке веб-сервера
- **mydomain.com.csr** — запрос на подпись сертификата, после создания сертификата этот файл не нужен, его можно удалить
- **mydomain.com.crt** — сертификат сайта. Указывает в конфигурации виртуального хоста при настройке веб-сервера, не является секретным.

Для глубокого понимания OpenSSL смотрите также полное руководство: [«OpenSSL: принципы работы, создание сертификатов, аудит»](#).

## Связанные статьи:

- [Как добавить сертификат Центра Сертификации \(CA\) в доверенные в Linux](#) (60%)
- [Где в Linux хранятся корневые сертификаты Центров Сертификации \(CA\)](#) (60%)
- [Как верифицировать сертификат SSL](#) (49.9%)
- [Установка и использование Docker в Linux \(Debian, Ubuntu, Arch Linux\)](#) (42%)
- [Как просмотреть содержимое ключей и сертификатов SSL](#) (35.2%)
- [Как записать символы строки в обратном порядке](#) (RANDOM - 16.9%)

## Как верифицировать сертификат SSL

Для проверки действительности сертификата используется команда

```
1 openssl verify СЕРТИФИКАТ
```

В разделе [«Как создать сертификаты SSL \(TLS\) для сайтов»](#) мы создали сертификат **mydomain.com.crt**, попробуем проверить его:

```
1 openssl verify mydomain.com.crt
```

Вывод:

```
1 C = US, ST = CA, O = "MyOrg, Inc.", CN = mydomain.com
2 error 20 at 0 depth lookup: unable to get local issuer certificate
3 error mydomain.com.crt: verification failed
```

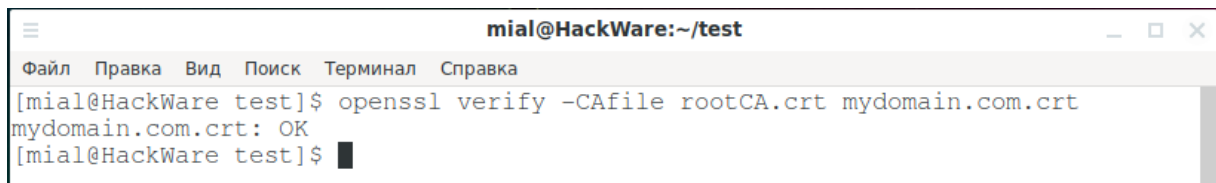
Верификация провалилась — причина в том, что мы использовали для подписи сертификата приватный ключ от своего собственного СА, который создали чуть ранее.

С помощью опции **-CAfile** можно указать сертификат доверенного Центра Сертификации:

```
1 openssl verify -CAfile rootCA.crt mydomain.com.crt
```

Как можно увидеть, проверка подлинности прошла успешно:

```
1 mydomain.com.crt: OK
```

A screenshot of a terminal window titled "mial@HackWare:~/test". The window has a menu bar with "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal shows the command "[mial@HackWare test]\$ openssl verify -CAfile rootCA.crt mydomain.com.crt" and the output "mydomain.com.crt: OK". The prompt "[mial@HackWare test]\$ " is visible at the bottom with a cursor.

```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify -CAfile rootCA.crt mydomain.com.crt
mydomain.com.crt: OK
[mial@HackWare test]$
```

Чтобы было поинтереснее, возьмём реальные сертификаты сайта и промежуточного центра сертификации:

```
1 openssl s_client -showcerts -connect w-e-b.site:443 </dev/null
```

```
mial@HackWare:~$ openssl s_client -showcerts -connect w-e-b.site:443 </dev/null
CONNECTED(00000003)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
verify return:1
depth=0 CN = w-e-b.site
verify return:1
---
Certificate chain
 0 s:CN = w-e-b.site
   i:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
-----BEGIN CERTIFICATE-----
MIIGSjCCBTKgAwIBAgI5A60V4TT1pNdv7zadhqTB7VsnMA0GCSqGSIb3DQEBCwUA
MExoCzAJBgNVBAYTA1VTMRYwFAYDVQQKEw1MZXRzZGw1MzYwYzA1MTIwMzYyNDZa
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMTZaAeFw0yMDA1MTIwMzYyNDZa
MDA4MTAwMzYyNDZaMBUxExARBgNVBAMTCnctZS1iLnNpdGUwggIiMA0GCSqGSIb3
DQEBAQUAA4ICDAwggIKAoICAQDt9bX5Lw4fusMh8UujZibCnITUvqo0a7b3G46p
fIaLdjbpfFHN1Vb6fnt0bYMNk6nRkq5eLPXuhN650yA7nznznDsKnfgTHNQ/CaA5F
FbF/aWpqC6kGKIaEwYK/JNzXzw5ApWiohV6A0JEun7TLW5iff5Ukw9xqnU3SGzZR
DEQmcnhoLwN13Y+U+prrqD3ud5O3AnzoR92vbPqA/OmDbSKvAessdd/sck4LgBop
Zf0+txzscThBtK1LBH7rwoAc68NjDA/wkfw2IJydL412tUnrUDHovPZvz4SSxt+L
7kdNUYHbJFBt2pxtco24two6Hw5CIPHSScdVJ1TRKw7AlReRjIb+zlz1SS0LP4Lma
soB03EqEN2hnuOowlx4a5XrakvPDvIT0ochPZUCuP83YY7DzkC3fOU+6qf/y9ICw
NRYExQUYbJjF2M/tp3SFCiaGTFBUBGBoDBFussOEUE6bLf/2U7aCnXRdF7VOUzqkr1
ZJxZ6EwEcojWBtTHTM53ygILCO6AAM389P62Ps3dSyrwDdso3RFMyNdi8lCgpQq1yZ
jhgcPqCjnbUGGcdhL0w8M8pcKd/2SUMaJp7HAGqv78Pd09Q2MR1mQbDiEUdmiPBLg
kx6hibTIBa4AZmV8wPuiAtwokuJXILqKfUK/J36TAhhoyqiKpTjnS1Kb9GhptNNO
UGyjeQIDAQABo4ICXTCALkwDgYDVR0PAQH/BAQDAgWgMBOGA1UdJQQWMBQGCCSg
AQUFBwMBBggrBgEFBQcDQjAMBGNVHRMBAf8EAjAAMB0GA1UdDgQWBQSR5q82mn9
WjUU5o8dsXDtg+CBRDafBgNVHSMEGDAWgBSosmpjBH3duubRobemRWXv86jsotBv
BggrBgEFBQcCBAQRjMGEwLgYlKwYBBQUHMAAGImh0dHA6Ly9vY3NwLmludC14My5s
ZXRzZW5jcnldwC5vcmcwLWYlKwYBBQUHMAAGI2h0dHA6Ly9jZXJ0LmludC14My5s
ZXRzZW5jcnldwC5vcmcwMBUGA1UdEQQOMAYCCnctZS1iLnNpdGUwTAYDVR0GBEUV
QzAIBgZngQwBAGewNwYlKwYBBAGC3xMBAQEwKDAmBggrBgEFBQcCARYaaHR0cDov
L2Nwcy5sZXRzZW5jcnldwC5vcmcwggECBgorBgEEAdZ5AgQCBiHziBIHwA0AdAdN
EvKwN34aYvuOyQxhhPHqezf1Vh0RJlvz4PNL8kfUbgAAAXIHMTw8AAEAwBFMEMC
Hy5UPPNslwG58vz9Qmaf95a2OS5/uctW6gaBKRTms0CIFndaJXJ4UxFqUQvsvuMv
Nqu3sB4Pva1TtecD7RT4u6zOAHYAsh4FzIuizYogTodm+Su5iiUgZ2va+nDnsk1T
Le+Lkf4AAAFyBzE8LQAABAMARzBFAiBxbmbaLsLje9GxejjQYcrtMZoAsLPB1H3Qx
IapJuM2ECCAihAPMHPz8lmf1A4NOvgdxndrdL1ozIALknfnwGrNfp75ukksAGCSqG
SIb3DQEBCwUAA4IBAQCkh81e4dDyI6Wtkbfzff5Rf3CntKN/ECItIsVBrbtbst7FE
iAJbY5AKryAai545UNUSb0Z0zPERSqi/kWno/E1+X0+uxStoIiESM7H4pf+BV1Sn
fKJnoCmoOONX0Xkzn2UJ2t2oig8Q2cwfXQRpvvdorYIFbPrMgdvTEutAyt7+2ZsU
lyNk5yqYngL1xbXbe9C4LjL6g/phIoZ4y2navjVxa4E+Kwh0lGd9Enrws/jknmQJJ
GF24Bc/PbNhUDJ1DvVxshoL9tQJRMq7voQ9QmIZUo4wVU4kn6+Dql8R/D+WtMthj
ATWjWlgn9+gpCmBmWeHpaOn8wg0Tj+bEHSJ208M1
-----END CERTIFICATE-----
 1 s:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
   i:O = Digital Signature Trust Co., CN = DST Root CA X3
-----BEGIN CERTIFICATE-----
MIIEkjCCA3gAwwIBAgIQCGfBQgAAVOfc2oLhevnCDANBakghkiG9w0BAQsFADA/
```

Вы увидите сертификаты (один или несколько), найдите по полю **CN** сначала сертификат домена **w-e-b.site**:

```
1 0 s:CN = w-e-b.site
```

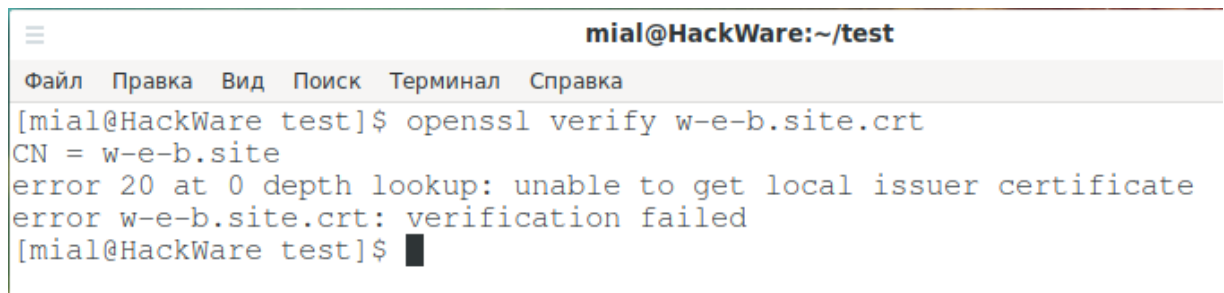
И скопируйте содержимое начиная с **-----BEGIN CERTIFICATE-----** и заканчивая **-----END CERTIFICATE-----**. Затем сохраните это в файл **w-e-b.site.crt**.

Затем найдите сертификат центра сертификации и сохраните его в файл **x3.crt**:

```
mial@HackWare:~  
Файл Правка Вид Поиск Терминал Справка  
1 s:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3  
i:O = Digital Signature Trust Co., CN = DST Root CA X3  
-----BEGIN CERTIFICATE-----  
MIIEKjCCA3qgAwIBAgIQCgFBQgAAVOfc2oLheynCDANBgkqhkiG9w0BAQsFADA/  
MSQwIgYDVQQKEExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDb250YXZAVBgNVBAMT  
DkRtVCSB290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTE2MDMxNzE2NDA0Nlow  
SjELMAkGA1UEBhMCVVMxIjAUBgNVBAoTUDUxldCdzIEVY3J5cHQxIzAhBgNVBAMT  
GkxldCdzIEVY3J5cHQxIzAUBgNVBAMTUDUxldCdzIEVY3J5cHQxIzAhBgNVBAMT  
AQ8AMIIBCgKCAQEAAnMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF  
q6menQhY7LEqXGihC6PjdeTm86dicbp5gWaf15Gan/PQeGdxyGkOlZHP/uaZ6WA8  
SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWan8jKkKQDIZO  
Z8h/pZq4UmeUEz9l6YKH9v6Dlb2honzhT+Xhq+w3Brvaw2VF3EK6BlspkENnWA  
a6xK8xuQsXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvj  
/PIzark5McWvxIONHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T  
AQH/BAgwBgEB/wIBADAQBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEdzBxMDIG  
CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv  
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k  
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUXKexpHsscfrb4UuQdf/EFWCfiRAw  
VAYDVR0gBE0wSzAIBgZnqQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC  
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbnNyeXB0Lm9yZzA8BgNVHR8ENTAz  
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu  
Y3JsMB0GA1UdDgQWBBS0SmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF  
AAOCAQEATPXEfnjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo  
uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/  
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUGwu  
X4Po1QYz+3dszkDqMp4fklxBwXRsw10KXzPMTZ+sOPaveyxindmjkW8lGy+QsRlG  
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iWBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6  
KOqkqm57TH2H3eDJAKsnh6/DNFu0Qg==  
-----END CERTIFICATE-----  
---  
Server certificate  
subject=CN = w-e-b.site  
  
issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3  
  
---  
No client certificate CA names sent  
Peer signing digest: SHA256  
Peer signature type: RSA-PSS  
Server Temp Key: X25519, 253 bits  
---  
SSL handshake has read 3609 bytes and written 392 bytes  
Verification: OK  
---  
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384  
Server public key is 4096 bit  
Secure Renegotiation IS NOT supported  
Compression: NONE  
Expansion: NONE  
No ALPN negotiated  
Early data was not sent  
Verify return code: 0 (ok)
```

Итак, для наших экспериментов у нас появилось два новых файла: **w-e-b.site.crt** (сертификат сайта) и **x3.crt** (сертификат промежуточного Центра Сертификации). Попробуем проверить действительность сертификата сайта:

```
1 openssl verify w-e-b.site.crt
```

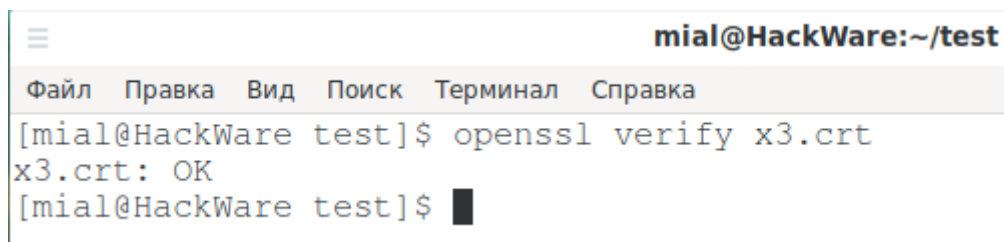


```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify w-e-b.site.crt
CN = w-e-b.site
error 20 at 0 depth lookup: unable to get local issuer certificate
error w-e-b.site.crt: verification failed
[mial@HackWare test]$
```

Поскольку сайт открывается нормально и веб браузер показывает, что с сертификатом сайта всё в порядке, то тот факт, что проверка подлинности не пройдена может показаться странным.

Проверим подлинность промежуточного сертификата организации, которая подписала сертификат сайта:

```
1 openssl verify x3.crt
```



```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify x3.crt
x3.crt: OK
[mial@HackWare test]$
```

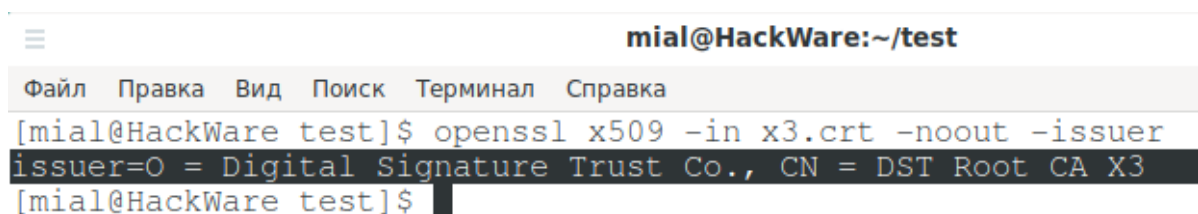
Как можно увидеть на скриншоте, с этим сертификатом всё в порядке.

Дело в том, что команда **openssl verify** сверяет подпись в сертификате с коревыми Центрами Сертификации, чьи сертификаты установлены в операционной системе как доверенные. Мы можем убедиться в этом. Следующая команда покажет организацию, подписавшую сертификат **x3.crt**:

```
1 openssl x509 -in x3.crt -noout -issuer
```

Вывод:

```
1 issuer=O = Digital Signature Trust Co., CN = DST Root CA X3
```



```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl x509 -in x3.crt -noout -issuer
issuer=O = Digital Signature Trust Co., CN = DST Root CA X3
[mial@HackWare test]$
```

Имя издателя в поле **CN** (Common Name) **DST Root CA X3**.

Убедимся, что сертификат этого CA действительно есть в системе:

```
1 awk -v cmd='openssl x509 -noout -subject' ' /BEGIN/{close(cmd)};{print | cmd}' < /e
```



```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ awk -v cmd='openssl x509 -noout -subject' ' ' /BEGIN/{close
(cmd)};{print | cmd}' < /etc/ssl/certs/ca-certificates.crt | grep 'DST Root CA
X3'
unable to load certificate
139993028351296:error:0909006C:PEM routines:get_name:no start line:crypto/pem/p
em_lib.c:745:Expecting: TRUSTED CERTIFICATE
subject=O = Digital Signature Trust Co., CN = DST Root CA X3
[mial@HackWare test]$
```

Если мы посмотрим, кто выдал сертификат сайта [w-e-b.site](http://w-e-b.site):

```
1 openssl x509 -in w-e-b.site.crt -noout -issuer
```

То увидим, что это **Let's Encrypt Authority X3**.

Посмотрим имя организации в сертификате **x3.crt**:

```
1 openssl x509 -in x3.crt -noout -subject
```

Имя совпало — это также **Let's Encrypt Authority X3**.

```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl x509 -in w-e-b.site.crt -noout -issuer
issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
[mial@HackWare test]$ openssl x509 -in x3.crt -noout -subject
subject=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
[mial@HackWare test]$
```

То есть в целом цепочка понятна — сертификат **w-e-b.site.crt** выдан организацией **Let's Encrypt Authority X3**, а её полномочия по выдаче сертификатов заверяет **DST Root CA X3**, которая в операционной системе находится среди доверенных корневых Центрах Сертификации.

Чтобы автоматически выполнить проверку всей цепочки с помощью **openssl verify**, нам нужно указать промежуточный сертификат после опции **-untrusted**. Если промежуточных сертификатов несколько, то нужно указать их все — опцию **-untrusted** можно использовать несколько раз.

```
1 openssl verify -untrusted x3.crt w-e-b.site.crt
```

```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify -untrusted x3.crt w-e-b.site.crt
w-e-b.site.crt: OK
[mial@HackWare test]$
```

Опция **-show\_chain** покажет информацию о цепочке сертификатов, которая была построена:

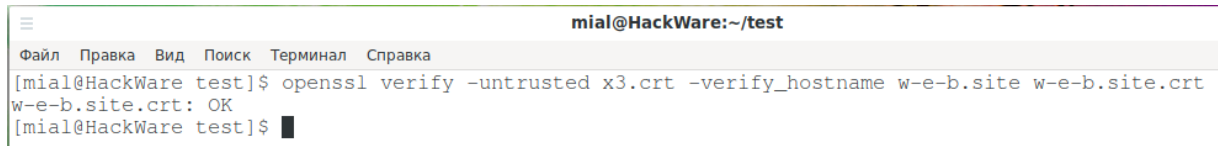
```
1 openssl verify -untrusted x3.crt -show_chain w-e-b.site.crt
```

```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify -untrusted x3.crt -show_chain w-e-b.site.crt
w-e-b.site.crt: OK
Chain:
depth=0: CN = w-e-b.site (untrusted)
depth=1: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3 (untrusted)
depth=2: O = Digital Signature Trust Co., CN = DST Root CA X3
[mial@HackWare test]$
```

Опции **-verify\_hostname ИМЯ-ХОСТА** и **-verify\_ip IP** позволяют проверить, действительно ли сертификат выдан для указанного хоста или IP адреса. Чтобы проверка прошла успешно, имя

хоста должно быть в Subject Alternative Name или Common Name в subject сертификата. А IP адрес должен быть в Subject Alternative Name в subject сертификата. Пример:

```
1 openssl verify -untrusted x3.crt -verify_hostname w-e-b.site w-e-b.site.crt
```



```
mial@HackWare:~/test
Файл  Правка  Вид  Поиск  Терминал  Справка
[mial@HackWare test]$ openssl verify -untrusted x3.crt -verify_hostname w-e-b.site w-e-b.site.crt
w-e-b.site.crt: OK
[mial@HackWare test]$
```

Следующими тремя командами вы можете проверить, соответствует ли SSL сертификат приватному ключу:

Для SSL сертификата выполните команду вида:

```
1 openssl x509 -noout -modulus -in ФАЙЛ.crt | openssl md5
```

Для приватного ключа RSA выполните команду вида:

```
1 openssl rsa -noout -modulus -in ФАЙЛ.key | openssl md5
```

Для CSR:

```
1 openssl req -noout -modulus -in ФАЙЛ.csr | openssl md5
```

Замените **ФАЙЛ** на имя ваших файлов. Для всего перечисленного выше — SSL сертификата, приватного ключа и запроса на подпись — хеш должен быть одинаковым.

Для глубокого понимания OpenSSL смотрите также полное руководство: [«OpenSSL: принципы работы, создание сертификатов, аудит»](#).

## Связанные статьи:

- [Как создать сертификаты SSL \(TLS\)](#) (100%)
- [Как просмотреть содержимое ключей и сертификатов SSL](#) (77.7%)
- [Лучшие терминальные мультиплексные инструменты](#) (50%)
- [Как просмотреть или отредактировать метаданные pdf или изображениях из командной строки Linux](#) (50%)
- [Как установить 7zip на Linux](#) (50%)
- [Как передавать данные между контейнером Docker и хостом](#) (RANDOM - 50%)