

使用 OpenSSL 與 cURL 檢查網站伺服器支援哪幾種 Cipher Suites

 blog.miniasp.com/post/2021/05/12/Checking-Cipher-Suites-using-OpenSSL-and-cURL

當我們從用戶端透過 TLS 協定連接伺服器時，用戶端會準備一組 Cipher Suites 清單給伺服器端，伺服器端必須選用一個強度最高的 Cipher 加密演算法回應，之後才會進行金鑰交換，並開始使用加密連線，確保網路安全。本篇文章將分享如何透過 OpenSSL 查詢有哪些 Cipher Suites 可用，並示範如何透過 cURL 指定特定 Cipher 進行連線與檢查伺服器是否支援特定 Cipher Suite 加密演算法清單。

使用 OpenSSL 查詢有哪些 Ciphers 可用

首先，你要先能夠區分 Cipher Suite 與 Cipher 的差別：

- **Cipher** (加密演算法)

一種負責將 明文 (Plaintext) 透過一系列數學運算，達到將訊息**加密**為 密文 (Ciphertext)；或將**密文**訊息**解密**為明文的演算法(algorithm)。

```
openssl ciphers -stdname
```

- **Cipher Suite** (加密演算法清單)

代表一系列可用的 Cipher (加密演算法) 清單。

目前從 [Cipher Suite Info](#) 網站可以發現 339 個在 [IANA](#) (Internet Assigned Numbers Authority) 登記立案的標準名稱。我們以 [TLS DHE DSS WITH AES 128 GCM SHA256](#) 為例，上面會顯示 **IANA name** (標準名稱)、**OpenSSL name** 與 **TLS Version** 可供參考。

參考資料: [Mapping OpenSSL cipher suite names to IANA names](#)

你可以用以下命令查詢目前的 OpenSSL 支援哪些標準名稱 (不會所有都支援)：

```
openssl ciphers
openssl ciphers -stdname
openssl ciphers -v
openssl ciphers -V
```

上述 4 個命令輸出的內容都差不多，第一個只會顯示 **OpenSSL name** 而已，使用 **-stdname** 會額外顯示 **IANA name** (標準名稱)，用 **-v** 會顯示 **OpenSSL name** 與相關屬性，而 **-V** 則會額外顯示 cipher suite 的 Hex code 內容。要用哪個參數，全看你想寫自動化的腳本而定。

如果想查詢 OpenSSL 目前支援的 Cipher Suite 可以用以下命令篩選出來：

```
openssl ciphers -s
openssl ciphers -s -psk
openssl ciphers -s -srp
```

預設 PSK 與 SRP 這兩類 Ciphers 是停用的，需要加上 **-psk** 或 **-srp** 才能看到。

當你知道有多少 Cipher Suite 可用之後，我們就可以來設法讀懂 Cipher Suite 的結構。以下我們以 [TLS AES 256 GCM SHA384](#) 為例：

1. TLS

要使用 Cipher Suite 的加密協定，網站通常是 TLS 為主。

2. AES

要使用的一種訊息加解密的演算法 (session cipher)

3. 256

加密演算法採用的加密金鑰長度 (session encryption key size for cipher) (bits)

4. GCM

一種加密的類型 (type of encryption) (cipher-block dependency and additional options)

5. SHA384

選用的雜湊演算法 (hash function)

使用 OpenSSL 查詢網站伺服器的所有憑證與 TLS 連接資訊

以下是一段非常實用的 OpenSSL 命令，他不但可以查出伺服器目前所使用的完整憑證資訊，也可以看出整個 TLS 交握的完整過程！

- OpenSSL 預設會使用 TLS 1.3 (最新版) 進行連線

```
openssl s_client -connect www.yahoo.com:443
```

我們可以跳過憑證的內容，直接顯示 **New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256** 這段內容，得知本次連線使用的 Cipher Suite：

```
New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
Server public key is 256 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
```

- 你也可以指定 TLS 1.2 進行加密連線

```
openssl s_client -connect www.yahoo.com:443 -tls1_2
```

```
New, TLSv1.2, Cipher is ECDHE-ECDSA-AES128-GCM-SHA256
```

```
Server public key is 256 bit
```

```
Secure Renegotiation IS supported
```

```
Compression: NONE
```

```
Expansion: NONE
```

```
No ALPN negotiated
```

```
SSL-Session:
```

```
Protocol : TLSv1.2
```

```
Cipher : ECDHE-ECDSA-AES128-GCM-SHA256
```

```
Session-ID:
```

```
72CB003E38A8CFC8F9085AFDB1AF26E6F6E11BF44CE7D0074B774E478D876E6F
```

```
Session-ID-ctx:
```

```
Master-Key:
```

```
A3DA1FB3E11FDAC50929F2D30339BB0F82094B7B503D5E8B018B844F169E7E479B0E11465F16F8
```

```
PSK identity: None
```

```
PSK identity hint: None
```

```
SRP username: None
```

```
TLS session ticket lifetime hint: 7200 (seconds)
```

```
TLS session ticket:
```

```
0000 - 07 8f f3 fa a9 32 7f 7f-eb 78 07 82 2c 8d 32 49 .....2...x...2I
0010 - 95 ee 44 c7 97 07 e6 f9-96 ba 60 97 14 0a 9b df ..D.....`.....
0020 - 05 2c f7 77 17 bf 75 b7-0d d7 d8 20 45 32 67 90 ., .w..u.... E2g.
0030 - 25 1d 2d 78 f8 0d 73 2b-55 92 bc 81 10 3f d0 29 %.-x..s+U....?.)
0040 - 14 3b 53 9b 84 6f a8 37-ca 67 5b b8 55 1e e6 04 .;S..o.7.g[.U...
0050 - 55 85 4c 9d b9 41 7c b3-d8 df 54 9f f4 85 72 89 U.L..A|...T...r.
0060 - a7 d6 f3 ca 49 79 15 31-f9 4f fd 2a d1 c9 0d 0d ....Iy.1.0.*....
0070 - de c4 ed e3 08 b5 00 73-62 1d 4b e1 eb 71 de 83 .....sb.K..q..
0080 - 13 34 73 92 9b 3d 35 92-01 48 1c 65 69 1f 76 8c .4s..=5..H.ei.v.
0090 - 3f af 07 1a c8 4e 70 64-42 a5 24 d0 5a 06 af 1d ?....NpdB.$..Z...
00a0 - 89 3c ff f0 e2 e1 4a f4-ff e6 2f e4 83 db f6 94 .<....J.../.....
00b0 - 00 00 b3 4b 10 60 dd e6-34 e7 63 ac a0 81 9a e6 ...K.`...4.c.....
00c0 - 61 85 9a 0f 84 3c df 5d-62 9f 09 6b be cc aa d3 a....<.]b..k....
```

```
Start Time: 1620800451
```

```
Timeout : 7200 (sec)
```

```
Verify return code: 0 (ok)
```

```
Extended master secret: yes
```

使用 cURL 指定 Cipher Suite 連接網站伺服器

你必須從 cURL 官方的 [SSL Ciphers](#) 網頁得知目前支援哪幾種 Cipher Suites，若以上一段的命令為例，這個 **ECDHE-ECDSA-AES128-GCM-SHA256** 是有在 cURL 的 TLS v1.2 cipher suites 清單中，因此我們可以指定用來連接遠端的伺服器。

- 檢測 TLS 連線的資訊

```
curl -s -S -v -o /dev/null --no-progress-meter https://www.yahoo.com
```

從回應的訊息可以看到完整的 TLS 交握過程，並且你也會發現 cURL 預設採用 **TLSv1.3** 最新版協定進行連線，並使用 **TLS_AES_128_GCM_SHA256** 這個 Cipher Suite：

```
* Trying 180.222.102.202:443...
* TCP_NODELAY set
* Connected to www.yahoo.com (180.222.102.202) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
*   Cpath: /etc/ssl/certs
} [5 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [512 bytes data]
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [122 bytes data]
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
{ [19 bytes data]
* TLSv1.3 (IN), TLS handshake, Certificate (11):
{ [2908 bytes data]
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
{ [80 bytes data]
* TLSv1.3 (IN), TLS handshake, Finished (20):
{ [36 bytes data]
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
} [1 bytes data]
* TLSv1.3 (OUT), TLS handshake, Finished (20):
} [36 bytes data]
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN, server accepted to use h2
```

- 指定最高使用 TLS v1.2 進行連線與指定 Cipher Suite 加密演算法清單

```
curl -s -S -v -o /dev/null --no-progress-meter --tls-max 1.2 --ciphers ECDHE-ECDSA-AES128-GCM-SHA256 https://www.yahoo.com
```

從以下訊息可以看出 **ECDHE-ECDSA-AES128-GCM-SHA256** 就是連線時選中的 Cipher Suite

```
* Trying 180.222.102.201:443...
* TCP_NODELAY set
* Connected to www.yahoo.com (180.222.102.201) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ECDHE-ECDSA-AES128-GCM-SHA256
* successfully set certificate verify locations:
* CAfile: /etc/ssl/certs/ca-certificates.crt
  Cpath: /etc/ssl/certs
} [5 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [512 bytes data]
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [122 bytes data]
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
{ [19 bytes data]
* TLSv1.3 (IN), TLS handshake, Certificate (11):
{ [2908 bytes data]
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
{ [79 bytes data]
* TLSv1.3 (IN), TLS handshake, Finished (20):
{ [36 bytes data]
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
} [1 bytes data]
* TLSv1.3 (OUT), TLS handshake, Finished (20):
} [36 bytes data]
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN, server accepted to use h2
```

檢查伺服器是否支援特定 Cipher Suite 加密演算法清單

以下我寫了一段 Bash 腳本，可以從 OpenSSL 取出 Cipher Suites 清單，然後透過 cURL 測試一個網站是否支援該 Cipher Suites，如果有支援，就會顯示 OK 字樣。

- TLS v1.3

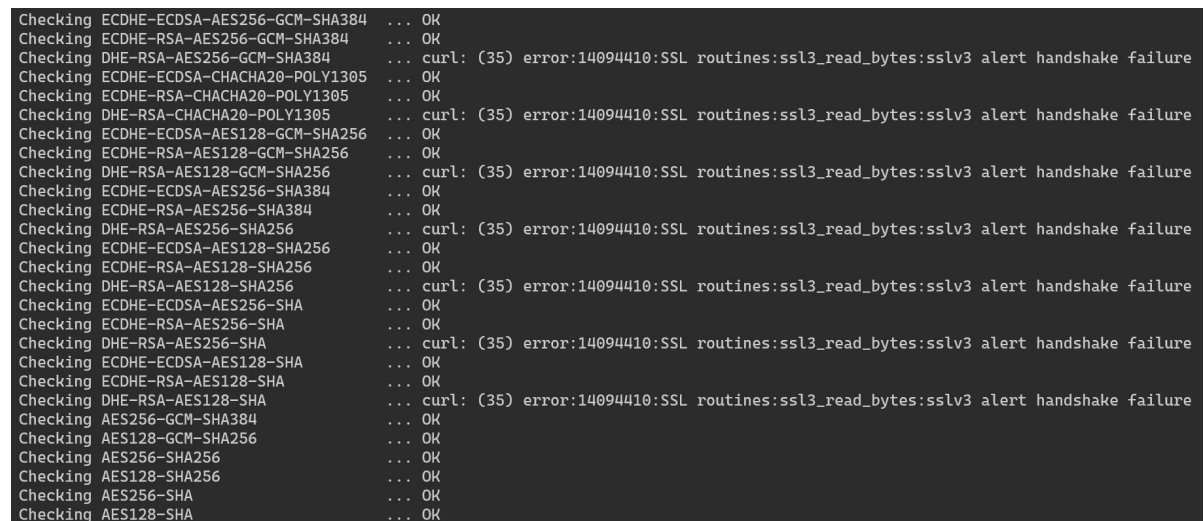
```
SUPPORTED_CIPHERS=$(openssl ciphers -s -tls1_3)
IFS=: read -r -a SUPPORTED_CIPHERS <<< $SUPPORTED_CIPHERS
for c in "${SUPPORTED_CIPHERS[@]}"
do
    printf -v pad %30s
    cipher=$c$pad
    printf "Checking ${cipher:0:30} ... "
    curl -s -S -o /dev/null --no-progress-meter --tls-max 1.3 --tls13-ciphers
$ curl https://www.yahoo.com
    if [ $? -eq 0 ]; then
        echo OK
    fi
done
```

注意: OpenSSL 1.1.1 之後的版本才能使用 `--tls13-ciphers` 參數指定 TLS v1.3 的 Cipher Suites, 舊版請改用 `--ciphers` 參數。

- TLS v1.2

```
SUPPORTED_CIPHERS=$(openssl ciphers -s -tls1_2)
IFS=: read -r -a SUPPORTED_CIPHERS <<< $SUPPORTED_CIPHERS
for c in "${SUPPORTED_CIPHERS[@]}"
do
    printf -v pad %30s
    cipher=$c$pad
    printf "Checking ${cipher:0:30} ... "
    curl -s -S -o /dev/null --no-progress-meter --tls-max 1.2 --ciphers $c
$ curl https://www.yahoo.com
    if [ $? -eq 0 ]; then
        echo OK
    fi
done
```

從下圖可以發現, 其實 Yahoo 首頁的網站伺服器已經有許多 Cipher Suites 不支援了!



```
Checking ECDHE-ECDSA-AES256-GCM-SHA384 ... OK
Checking ECDHE-RSA-AES256-GCM-SHA384 ... OK
Checking DHE-RSA-AES256-GCM-SHA384 ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-CHACHA20-POLY1305 ... OK
Checking ECDHE-RSA-CHACHA20-POLY1305 ... OK
Checking DHE-RSA-CHACHA20-POLY1305 ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-AES128-GCM-SHA256 ... OK
Checking ECDHE-RSA-AES128-GCM-SHA256 ... OK
Checking DHE-RSA-AES128-GCM-SHA256 ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-AES256-SHA384 ... OK
Checking ECDHE-RSA-AES256-SHA384 ... OK
Checking DHE-RSA-AES256-SHA256 ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-AES128-SHA256 ... OK
Checking ECDHE-RSA-AES128-SHA256 ... OK
Checking DHE-RSA-AES128-SHA256 ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-AES256-SHA ... OK
Checking ECDHE-RSA-AES256-SHA ... OK
Checking DHE-RSA-AES128-SHA ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking ECDHE-ECDSA-AES128-SHA ... OK
Checking DHE-RSA-AES128-SHA ... curl: (35) error:14094410:SSL routines:ssl3_read_bytes:ssl3 alert handshake failure
Checking AES256-GCM-SHA384 ... OK
Checking AES128-GCM-SHA256 ... OK
Checking AES256-SHA256 ... OK
Checking AES128-SHA256 ... OK
Checking AES256-SHA ... OK
Checking AES128-SHA ... OK
```

相關連結

