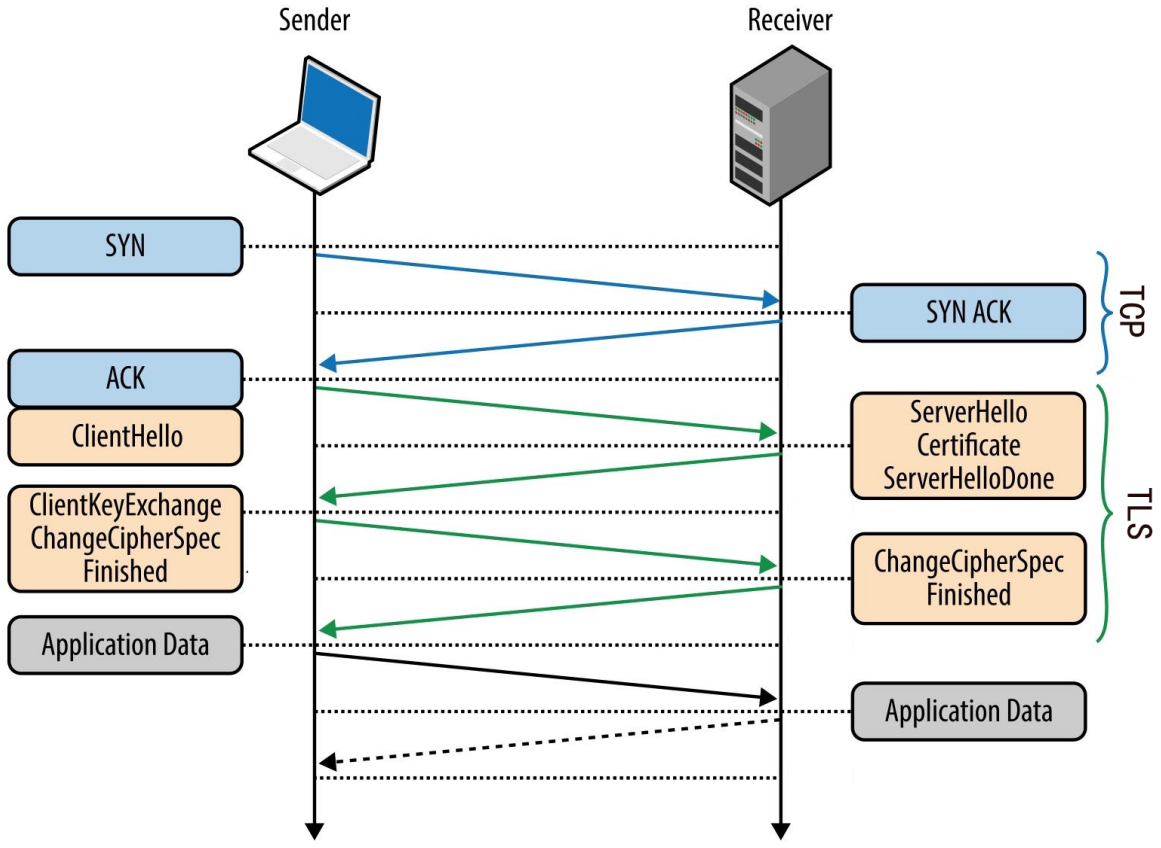
一、前言

sslscan用于扫描SSL/TLS证书并报告协议版本、密码套件、密钥交换、签名算法和证书详细信息等，帮助用户从安全角度加强数据传输安全性，同时，sslscan还可以将结果输出到XML文件中，以便外部程序使用。本文将详细介绍sslscan用法案例及其扫描原理。

终端输出的颜色编码及其含义：

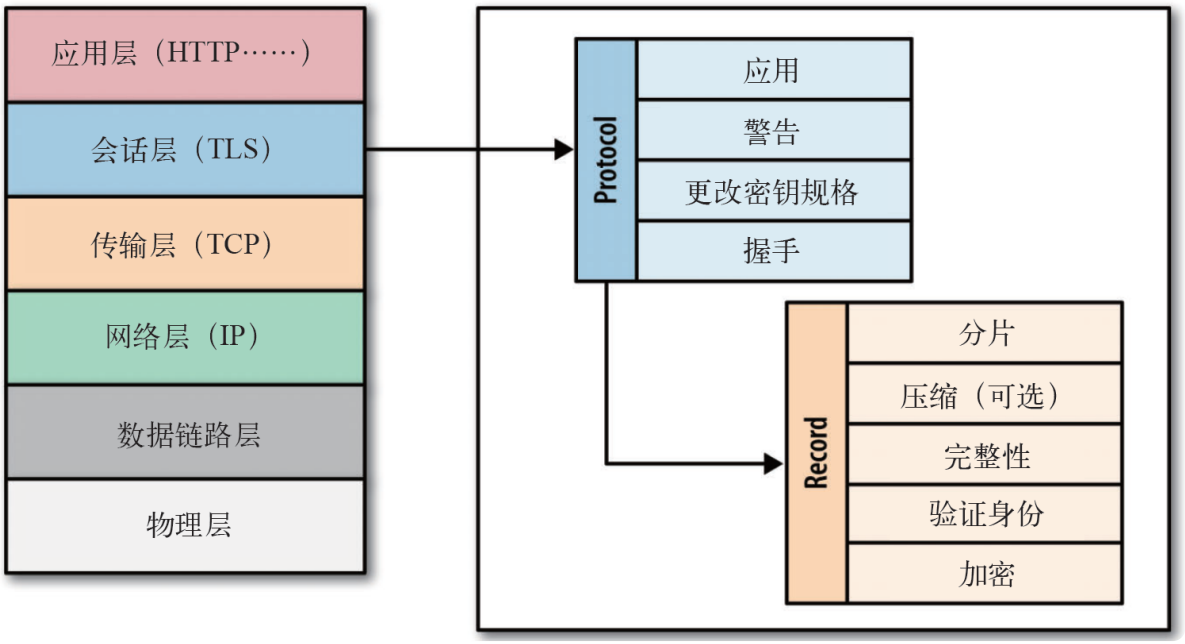
颜色	密码套件
红色背景	空套件
红色字体	破损套件（<=40 bit）、破损协议（SSLv2/SSLv3）或破损证书的签名算法（MD5）
黄色字体	弱密码套件（<=56 bit或RC4）或弱证书签名算法（SHA-1）
紫色字体	匿名套件（ADH或AECDH）

TLS协商过程



在TCP三次握手后的TLS握手阶段（即淡黄色部分）即为我们重点关注和检测的部分。

所在层级



TLS命名为安全传输层协议（*Transport Layer Security*），实际作用在传输层偏上的会话层，它既不影响上层协议又能保证上层协议的网络通信安全。

二、安装

sslscan在各大发行版的软件源中几乎都有，可以直接从软件源安装：

发行版	安装命令
ArchLinux	pacman -Sy sslscan
Centos/Redhat	yum install sslscan -y
Debian/Ubuntu	apt install sslscan -y
Gentoo	emerge --ask sslscan

也可以将源码git clone到本地进行编译安装：

```
$ git clone https://github.com/rbsec/sslscan
$ cd sslscan
$ make static
$ ./sslscan --version
```

三、输出结构

不加任何参数的情况下，直接执行 `sslscan <目标>` 进行扫描，会完整的将各个部分展示到结果上。

各部分的输出含义如下：

区域	原文	含义
第一部分	SSL/TLS Protocols	支持的SSL/TLS版本
第二部分	TLS Fallback SCSV	是否支持TLS Fallback SCSV
第三部分	TLS renegotiation	是否支持TLS重协商
第四部分	TLS Compression	是否支持TLS压缩
第五部分	Heartbleed	是否存在心脏滴血漏洞
第六部分	Supported Server Cipher(s)	服务端支持的加密套件
第七部分	Server Key Exchange Group(s)	服务端支持的密钥交换组
第八部分	SSL Certificate	证书详细信息

以下面这段输出为例：

```
$ sslscan www.google.com
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 172.217.27.4

Testing SSL server www.google.com on port 443 using SNI name www.google.com

  SSL/TLS Protocols:  # 第一部分
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   enabled

  TLS Fallback SCSV:  # 第二部分
Server supports TLS Fallback SCSV

  TLS renegotiation:  # 第三部分
Secure session renegotiation supported

  TLS Compression:    # 第四部分
Compression disabled

  Heartbleed:         # 第五部分
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
```

```
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):  # 第六部分
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
...略

  Server Key Exchange Group(s):  # 第七部分
TLSv1.3  128 bits  secp256r1 (NIST P-256)
TLSv1.3  128 bits  x25519
TLSv1.2  128 bits  secp256r1 (NIST P-256)
TLSv1.2  128 bits  x25519

  SSL Certificate:              # 第八部分
Signature Algorithm: sha256WithRSAEncryption
ECC Curve Name:      prime256v1
ECC Key Strength:    128

Subject:  www.google.com
Altnames: DNS:www.google.com
Issuer:   GTS CA 1C3

Not valid before: Dec 11 08:10:00 2023 GMT
Not valid after:  Mar  4 08:09:59 2024 GMT
```

四、用法案例及参数说明

需要注意，sslscan的所有参数选项必须写在被扫描的目标主机之前，不能写在后面，否则会报错，目前最新2.0版本是这样，后续随着版本更新可能有所优化。

正确示例：

```
sslscan --show-certificate google.com
```

错误示例：

```
sslscan google.com --show-certificate
```

1.扫描支持的SSL/TLS版本、cipher信息等

在『**输出结构**』中说过，不加任何参数的情况下，sslscan会显示总共八部分的内容。

同时，需要注意：

- 目标可以是域名、IP；

- 不指定端口的情况下默认为443，如需指定自定义端口，接`:port`即可，比如`domain.com:8443`；
- 输出结果中的服务端支持的加密套件(Supported Server Cipher(s))，Prefereed表示优先选中的套件，Accepeted表示支持的套件；
- SSL/TLS等级和安全性：TLSv1.3 > TLSv1.2 > TLSv1.1 > TLSv1.0 > SSLv3 > SSLv2。

如果此时已经满足需求，则无需加任何多余的参数：

```
sslscan <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan google.com
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 172.217.24.238

Testing SSL server google.com on port 443 using SNI name google.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
Preferred TLSv1.2  256 bits  ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  128 bits  AES128-SHA
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA
```

2.检测OCSP的状态（--ocsp）

1）CRL和OCSP

在此之前，首先了解下什么是**CRL**。**CRL**（Certificate Revocation List）证书吊销列表是 RFC 5280 定义的检查证书状态的机制。

想象一种场景，客户端通过SSL/TLS连接到服务端，怎么确保证书本身是否可靠安全？比如证书是否由于各种原因被证书申请者申请在证书有效期内提前吊销证书或安全原因被机构主动吊销（比如泄漏私钥的场景）？

首先每个证书颁发机构会维护并持续更新的一个已吊销的证书列表，任何想验证证书是否被吊销的用户都能下载此列表，如果列表中有你要被验证的证书，说明证书已经被吊销了，不再安全可信。

随着证书越来越多，CRL文件也愈来愈冗长，所以也会导致一些问题：

- CRL列表随着被吊销的证书日益增多而变得冗长，每个客户端都必须取得包含所有证书序列号的CRL完整列表；
- 刚被吊销的证书，CRL列表更新并不及时，比如客户端已经缓存了CRL的场景，在缓存期内，证书会被认为一直有效。

基于上面两个历史原因，在 RFC 2560 又推出了**OCSP**（Online Certificate Status Protocol）在线证书状态协议，可以完美解决上面两个问题，首先支持实时检查证书状态的机制，并且支持查询需要被验证的证书序列号是否有效，而无需像CRL一样将整个CRL列表弄下来，也节省了网络带宽资源。

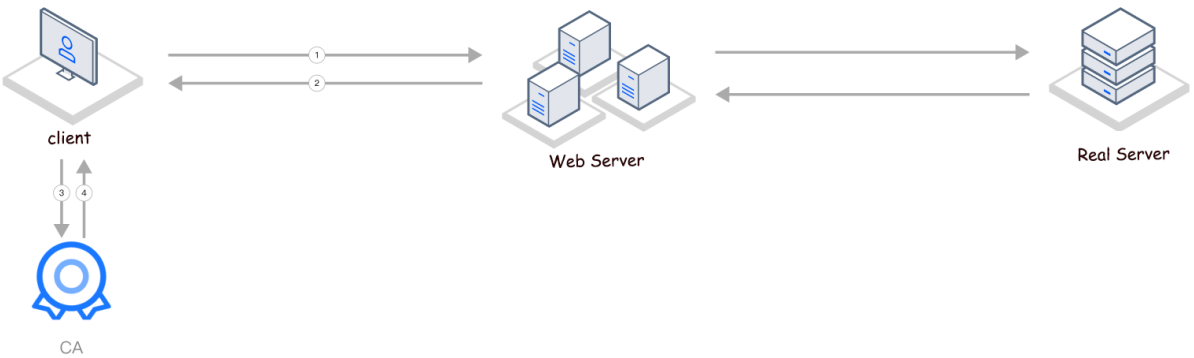
虽然解决了CRL的两大难题，但OCSP也有一些弊端：

- 证书机构需要实时处理来自世界各地的OCSP查询（OCSP Request），这对证书机构用于负责OSCP查询的服务器高可用性是一种挑战；
- 如果OCSP请求没有得到成功，在进一步协商阶段可能会处于阻塞状态，比如OCSP服务器在境外被大陆限制访问或者被DNS污染，此时请求始终无法完成；
- 通过OCSP Request发送给证书颁发机构进行实时查询证书可用性，也可能会导致泄漏客户端的隐私，知道客户端在访问谁。

2）OCSP预装订

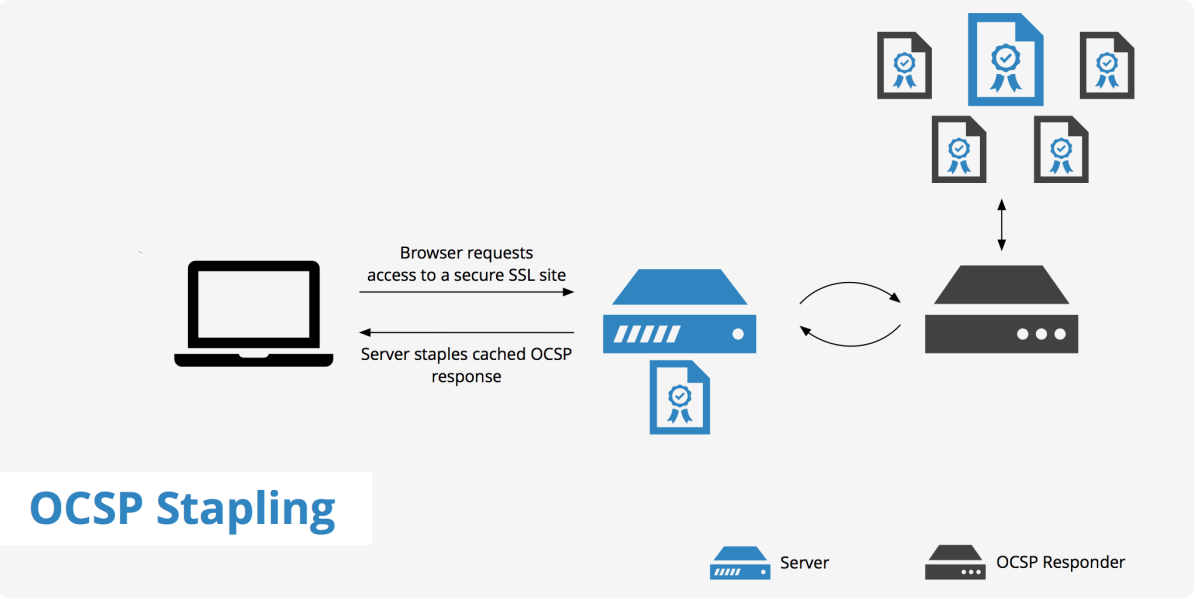
为了解决OCSP弊端，OCSP可以装订在Web服务器上，此时客户端直接向Web服务器发送OCSP Request，再由Web服务器统一去请求证书机构的OCSP服务器，并将结果缓存下来。

装订前：每台客户端第一次请求TLS证书时，都会先向CA证书颁发机构发送OCSP Request。



- 1.client向Web服务器发起TLS握手请求；
- 2.Web服务器响应TLS握手（返回证书）；
- 3.client向CA证书颁发机构的OCSP服务器发起OCSP查询；
- 4.CA证书颁发机构的OCSP服务器向client返回查询结果。

装订后：客户端发送OCSP Request给Web服务端，由Web服务端向CA证书颁发机构发送OCSP查询请求，再响应给客户端，并将结果缓存下来。



3）检测OCSP装订状态

--ocsp可直接检测对端是否支持OCSP装订：

```
sslscan --ocsp
```

```
┌──(root㉿kali)-[~]
└─# sslscan --ocsp [redacted]
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to [redacted]

Testing SSL server [redacted] on port 443 using SNI name [redacted]

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed

  OCSP Stapling Request:
OCSP Response Status: successful (0x0)
Response Type: Basic OCSP Response
Version: 1 (0x0)
Responder Id: 5F3A7C11107E0C677161DC8BA3B5000367F5571C
Produced At: Jan 23 04:39:09 2024 GMT
Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 0386F8E67CC75256C1D9F6EE76A7D8EABB7ADDF2
      Issuer Key Hash: 5F3A7C11107E0C677161DC8BA3B5000367F5571C
      Serial Number: 70959A60CE23A0EFEED0951232BFE393
    Cert Status: good
    This Update: Jan 23 04:39:09 2024 GMT
    Next Update: Jan 30 04:39:08 2024 GMT

  Supported Server Cipher(s):
Preferred TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
```

如果不支持则会明确显示：**No OCSP response received.**

```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name data.linux-code.com --ocsp 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name 192.168.1.81

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  OCSP Stapling Request:
No OCSP response received.
```

同时，myssl.com也支持在线检测：

证书信息

RSA 1 | RSA 2

信任状态	可信
通用名称	[redacted]
颁发者	TrustAsia RSA DV TLS CA G2
启用SNI	是
弱密钥检测	否
加密算法	RSA 2048 bits
签名算法	SHA384WithRSA
证书透明(CT)	是 (Apple: (来自证书，有效))
证书品牌	TrustAsia
证书类型	DV SSL
开始时间	2023-08-24 08:00:00
结束时间	2024-08-24 07:59:59
吊销状态	正常
OCSP装订状态	正常
OCSP必须装订	否
组织机构	--
部门	--
备用名称	[redacted]

顺便说下，Nginx可以在配置文件中启用OCSP装订，增加以下两行配置：

```
ssl_stapling on;
ssl_stapling_verify on;
```

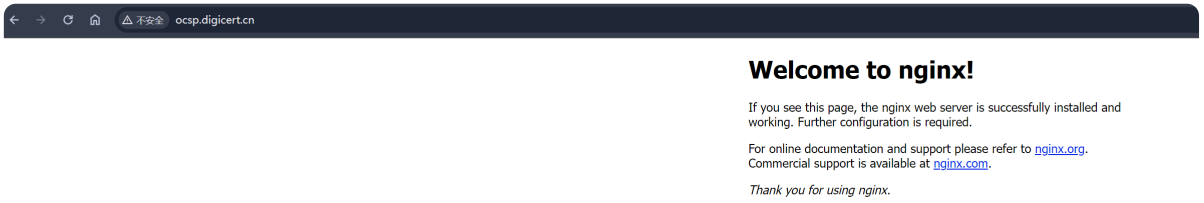
4）检测证书颁发机构的OCSP/CRL服务器

这项功能其实属于下面要讲的参数，即**--show-certificate**会展示证书的完整信息，也包括证书里指向的OCSP主机：

```
sslscan --show-certificate <目标>|grep -A2 'Authority Information'
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-certificate baidu.com|grep -A2 'Authority Information'
    Authority Information Access:
      OCSP - URI:http://ocsp.digicert.cn
      CA Issuers - URI:http://cacerts.digicert.cn/DigiCertSecureSiteProCNCAG3.crt
```

百度的证书是DigiCert机构颁发的，因此OCSP指向的也是DigiCert的OCSP服务器，它甚至可以被浏览器进行访问：



解析到的是国内CDN节点：

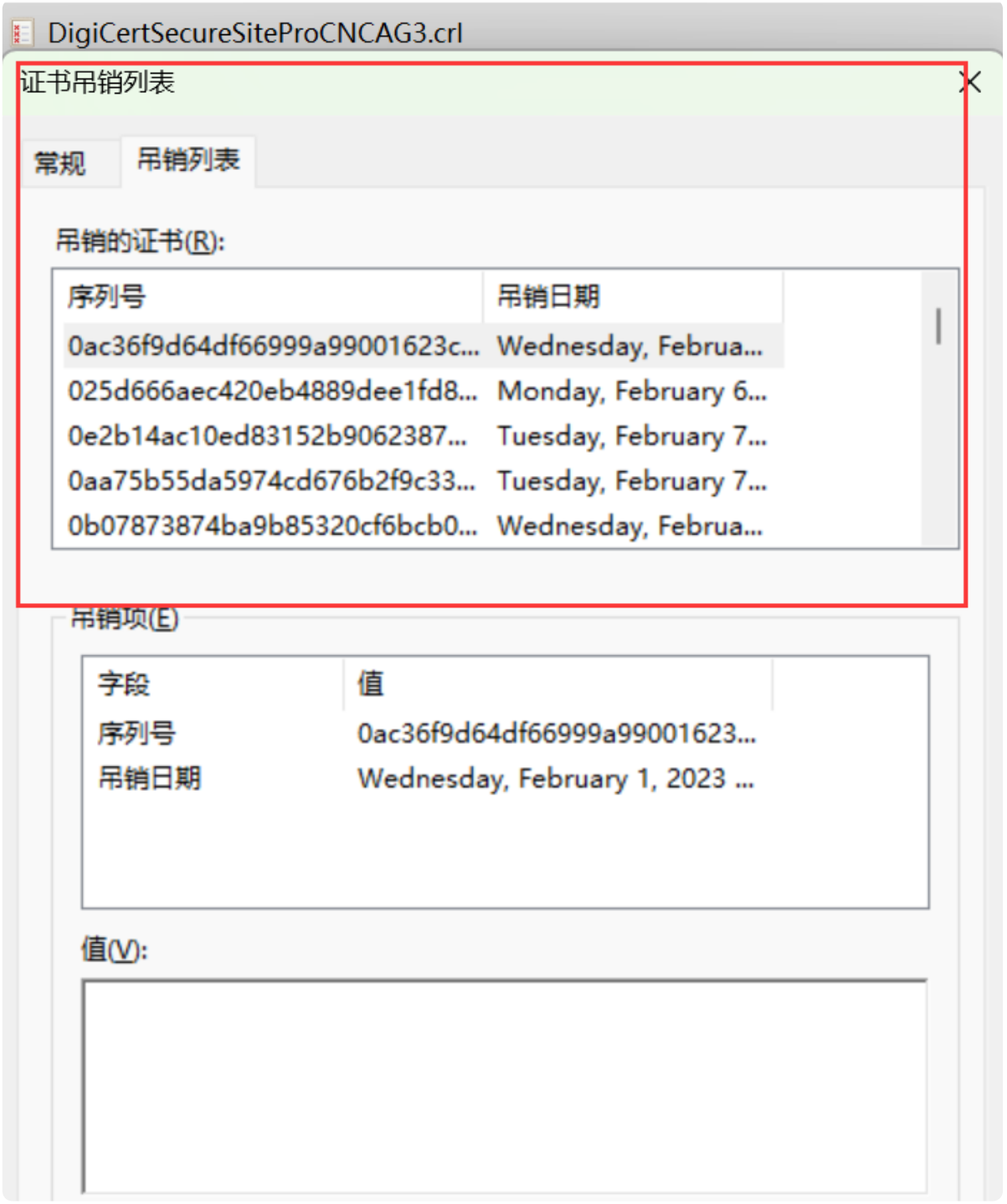
```
02:44:05  ~  dig ocsp.digicert.cn

; <<>> DiG 9.16.41 <<>> ocsp.digicert.cn
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49882
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;ocsp.digicert.cn.              IN      A

;; ANSWER SECTION:
ocsp.digicert.cn.       272     IN      CNAME   ocsp.digicert.cn.w.cdngslb.com.
ocsp.digicert.cn.w.cdngslb.com. 42 IN   A       222.217.94.113[广西河池市宜州市 电信]

;; Query time: 3 msec
;; SERVER: 192.168.1.1[局域网 对方和您在同一内部网]#53(192.168.1.1[局域网 对方和您在同一内部网])
;; WHEN: Thu Jan 25 02:44:13 CST 2024
;; MSG SIZE  rcvd: 105
```

同理，CRL证书吊销列表也会包含在证书信息里：

```
sslscan --show-certificate <目标> | grep -A2 'CRL Distribution'
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-certificate baidu.com |& grep -A2 'CRL Distribution'
      X509v3 CRL Distribution Points:
        Full Name:
          URI:http://crl.digicert.cn/DigiCertSecureSiteProCNCAG3.crl
```

这个crl文件时可以下载下来的，里面包含了被DigiCert吊销的证书序列号和吊销日期：



3.显示证书详细信息（--show-certificate）

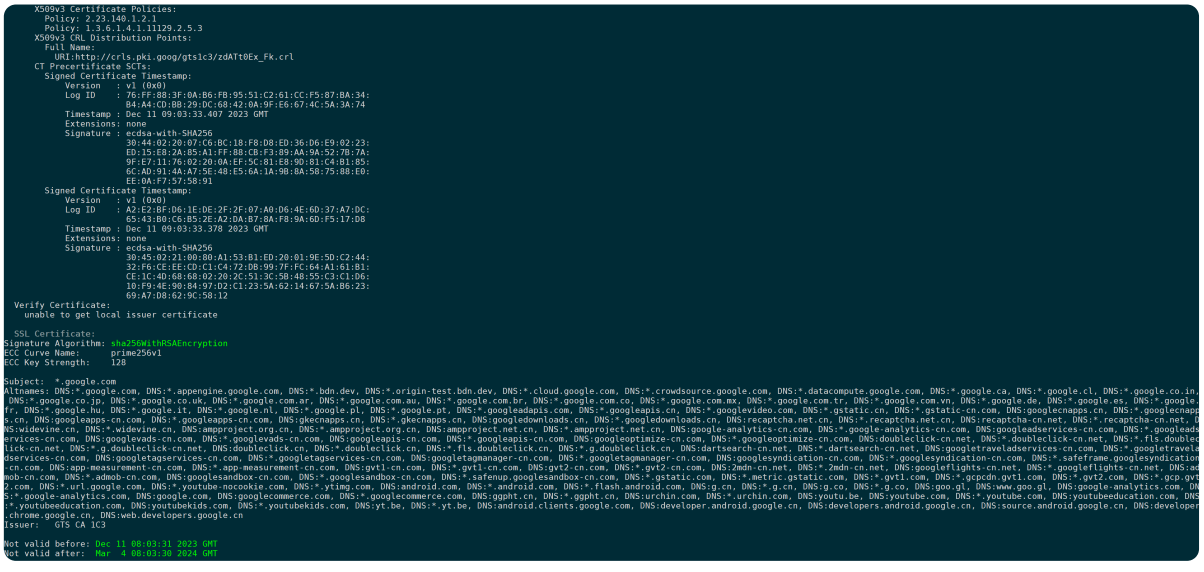
--show-certificate选项用于显示服务器提供的SSL/TLS证书的详细信息。`sslscan`会在扫描结果中包含服务器证书的详细信息，包括证书颁发机构（CA）、证书有效期、证书序列号、证书签名算法、公钥算法、公钥大小、主题备用名和颁发等信息。

使用此参数，会将证书的详细信息在最后一部分展示：

```
sslscan --show-certificate <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-certificate google.com
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 172.217.24.238

Testing SSL server google.com on port 443 using SNI name google.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256         Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384         Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
Preferred TLSv1.2  256 bits  ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
```

篇幅过长，只截取部分：



4.指定SNI（--sni-name）

SNI（Server Name Indication）是TLS协议的一个扩展，在TLS握手时用来标记客户端的关键信息，它改变了TLS协议在握手阶段只能协商一个服务器名的限制，使得在一个IP地址和端口上可以同时支持多个域名或多个SSL证书，Web服务器可以检查SNI主机名，选择适当的证书，继续完成握手。

TLS+SNI和HTTP中发送Host头部是一样的，后者是客户端在请求头中包含主机名，但作用都是同一个IP地址服务于不同的域名，而区分不同域名的方法则是SNI或者Host。

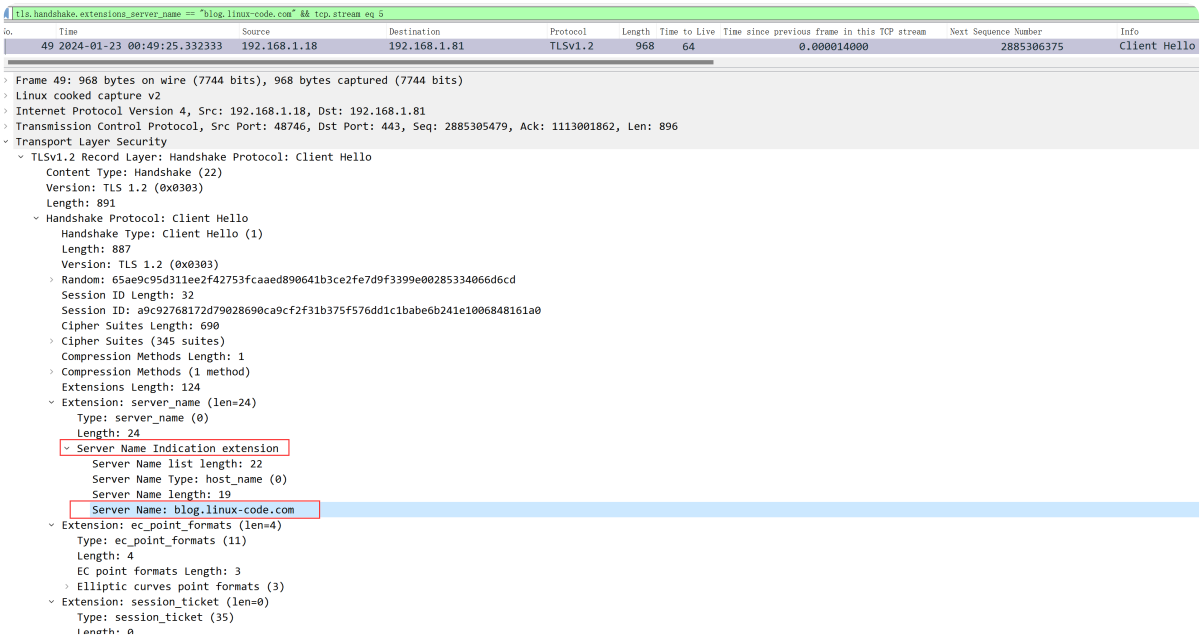
示例：

```
sslscan --sni-name=<domain.com> <目标>
```

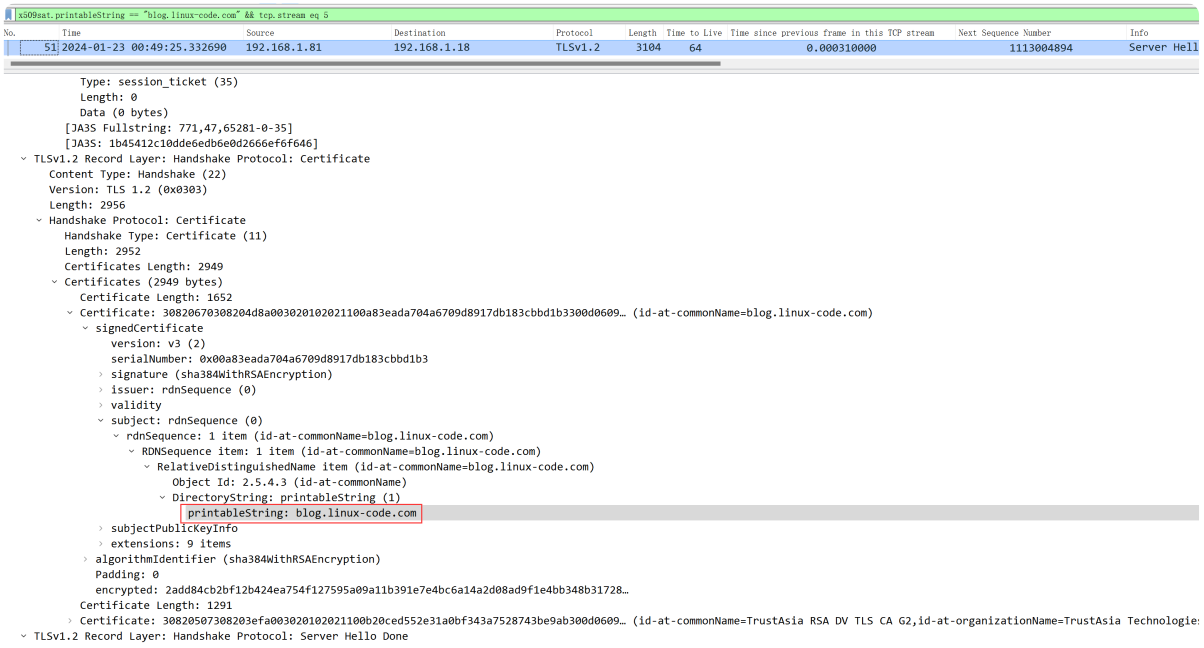
```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name=blog.linux-code.com 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name blog.linux-code.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
```

不指定SNI的情况下，目标是IP则为IP，目标是域名则为域名。

通过指定SNI，在Client Hello阶段可以看到，TLS头部里的SNI扩展字段，携带了我们指定的SNI主机名：



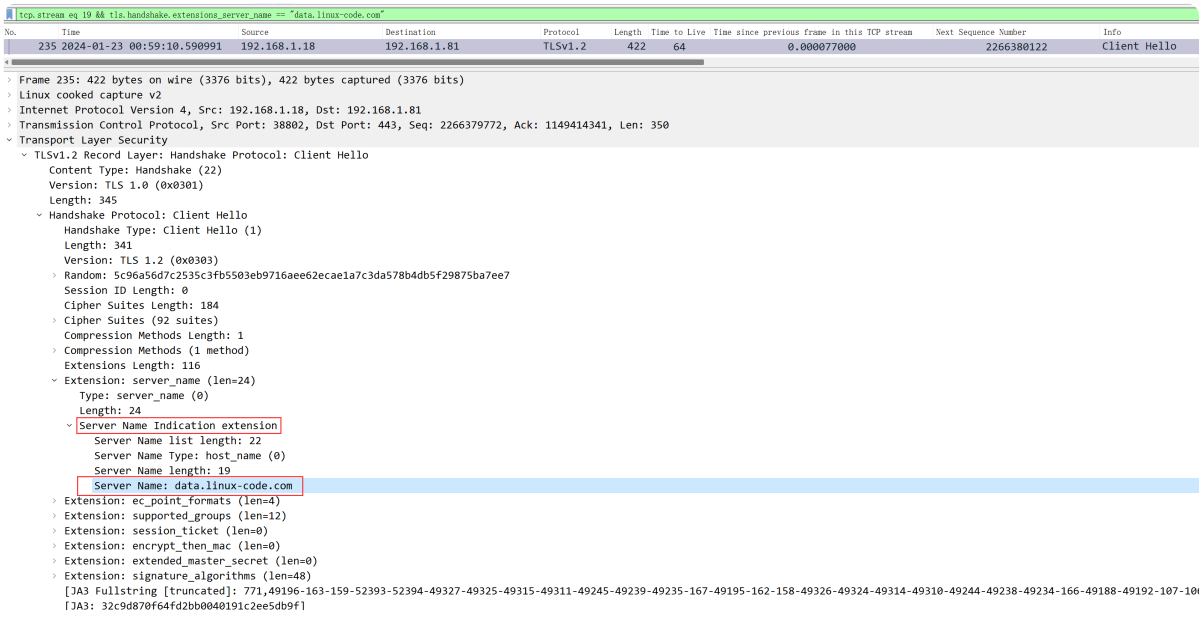
TLS的Server Hello阶段返回的证书信息，也是此域名：



sslscan则是通过Server Hello包返回的信息，最终呈现在结果上：

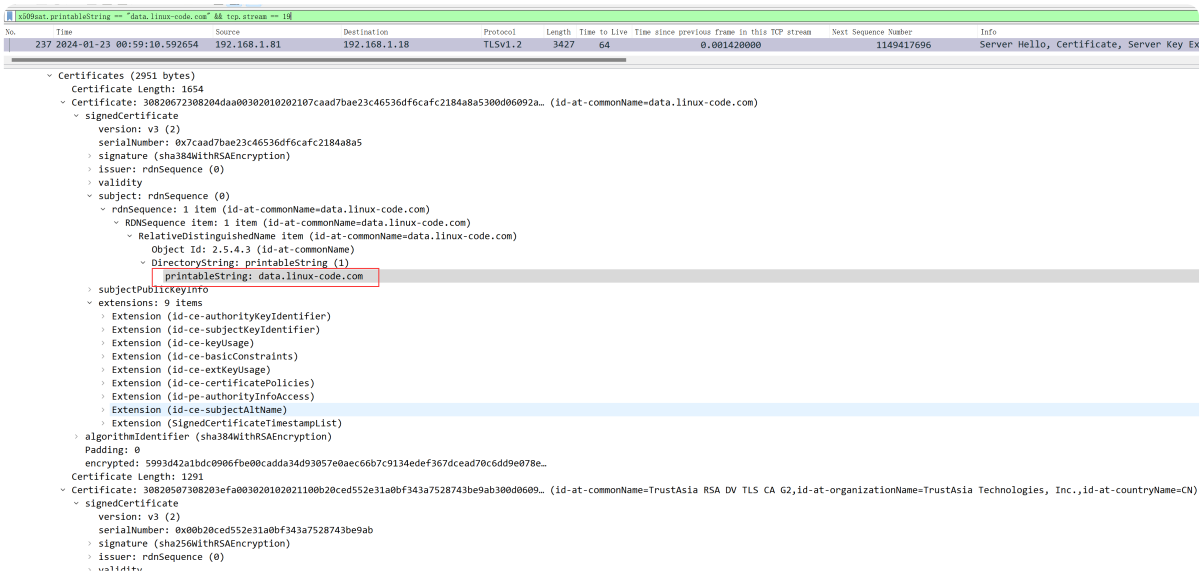
```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name=blog.linux-code.com 192.168.1.81 |tail
  SSL Certificate:
Signature Algorithm: sha384WithRSAEncryption
RSA Key Strength:    2048

Subject:  blog.linux-code.com
Altnames: DNS:blog.linux-code.com
Issuer:   TrustAsia RSA DV TLS CA G2

Not valid before: Jun 10 00:00:00 2023 GMT
Not valid after:  Jun  9 23:59:59 2024 GMT
```

同一个IP，SNI指定为另一个域名：



返回的证书结果也是SNI对应的域名：

```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name=data.linux-code.com 192.168.1.81 |tail
  SSL Certificate:
Signature Algorithm: sha384WithRSAEncryption
RSA Key Strength:    2048

Subject:  data.linux-code.com
Altnames: DNS:data.linux-code.com
Issuer:   TrustAsia RSA DV TLS CA G2

Not valid before: Aug  8 00:00:00 2023 GMT
Not valid after:  Aug  7 23:59:59 2024 GMT
```



因此，作用和通过/etc/hosts指定某几个域名解析到同一个IP分别进行访问出现不同的结果存在异曲同工之处，此时校验的则是HTTP头部里的Host字段，TLS则可以通过SNI扩展来实现。

5.不检查证书有效性（--no-check-certificate）

此参数和curl的-k参数，wget的--no-check-certificate参数作用类似，不检测证书有效性。

但也有所区别，区别在于sslscan加上此类参数，只是不输出『**输出结构**』中的第八部分证书信息，curl和wget命令的相关参数如果不加，如果证书无效，则直接报错无法往下进行，TLS握手都无法完成。

比如下面这个示例：

```
sslscan --no-check-certificate <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name=console.linux-code.com 192.168.1.81 |nl|tail -n 10
    69    SSL Certificate:
    70  Signature Algorithm: sha256WithRSAEncryption
    71  RSA Key Strength:    2048

    72  Subject:  console.linux-code.com
    73  Altnames: DNS:console.linux-code.com
    74  Issuer:   TrustAsia TLS RSA CA

    75  Not valid before: Feb 23 00:00:00 2022 GMT
    76  Not valid after:  Mar 15 23:59:59 2023 GMT

┌──(root㉿kali)-[~]
└─# sslscan --sni-name=console.linux-code.com --no-check-certificate 192.168.1.81 |nl|tail -n 10
    60  Accepted  TLSv1.0  256 bits  CAMELLIA256-SHA
    61  Accepted  TLSv1.0  128 bits  AES128-SHA
    62  Accepted  TLSv1.0  128 bits  CAMELLIA128-SHA

    63    Server Key Exchange Group(s):
    64  TLSv1.2  128 bits  secp256r1 (NIST P-256)
    65  TLSv1.2  192 bits  secp384r1 (NIST P-384)
    66  TLSv1.2  260 bits  secp521r1 (NIST P-521)
    67  TLSv1.2  128 bits  x25519
    68  TLSv1.2  224 bits  x448
```

可以看到，不加一共输出76行，默认输出有效性等信息，加上此参数后一共输出68行，69-76行的证书信息全部省略。

6.显示服务端允许的客户端CA列表（--show-client-cas）

在HTTPS双向认证中使用，客户端校验服务端证书，服务端也校验客户端证书，此参数可以检测到服务端支持哪些客户端证书的CA机构。

以通过openssl自签的双向证书为例：

```
sslscan --show-client-cas <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-client-cas [redacted]:444
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to [redacted]

Testing SSL server [redacted] on port 444 using SNI name [redacted]

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   disabled
TLSv1.3   enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.3 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253

  Server Key Exchange Group(s):
TLSv1.3  128 bits  secp256r1 (NIST P-256)
TLSv1.3  192 bits  secp384r1 (NIST P-384)
TLSv1.3  260 bits  secp521r1 (NIST P-521)
TLSv1.3  128 bits  x25519
TLSv1.3  224 bits  x448

  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    4096

Subject:  docs.linux-code.com
Issuer:   docs.linux-code.com

Not valid before: Aug  9 06:58:14 2022 GMT
Not valid after:  Aug  6 06:58:14 2032 GMT

  Acceptable client certificate CA names:
/C=CN/ST=GuangDong/L=ShenZhen/O=tencent/OU=tencent/CN=docs.linux-code.com/emailAddress=[redacted]
```

在最后的**Acceptable client certificate CA names**部分可以看到服务端输出的允许的客户端CA列表。

需要注意的是，此参数对于IIS/Schannel服务器，将返回空，暂不支持此类场景。

7.输出支持的cipher完整列表（--show-ciphers）

--show-ciphers将显示sslscan支持的密码的完整列表，目前有1200多个cipher。

```
sslscan --show-ciphers <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-ciphers qcloud.com|sed -n '14,1303p'|nl|shuf -n 30
  385	DHE-RSA-CAMELLIA256-SHA256
  820	TLS_RSA_PSK_WITH_NULL_SHA256
  579	ADH-CAMELLIA256-SHA
  873	TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA
  586	AECDH-AES128-SHA
  209	DHE-RSA-AES256-SHA256
  485	ECDHE-PSK-CAMELLIA128-SHA256
  720	TLS_KRB5_WITH_DES_CBC_SHA
  765	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
  114	CAMELLIA128-SHA256
 1238	PRIVATE_CIPHER_202
 1035	TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256
 1227	PRIVATE_CIPHER_191
 1251	PRIVATE_CIPHER_215
  216	ADH-CAMELLIA256-SHA256
  975	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
 1156	PRIVATE_CIPHER_120
  226	ADH-CAMELLIA128-SHA256
  337	RSA-PSK-NULL-SHA256
 1166	PRIVATE_CIPHER_130
 1194	PRIVATE_CIPHER_158
  872	TLS_SRP_SHA_WITH_AES_256_CBC_SHA
  204	DHE-DSS-ARIA128-GCM-SHA256
 1137	PRIVATE_CIPHER_101
  455	AES256-SHA256
  595	RSA-PSK-AES256-GCM-SHA384
  751	TLS_RSA_WITH_AES_256_CBC_SHA256
 1010	TLS_PSK_DHE_WITH_AES_128_CCM_8
   50	ECDHE-RSA-CAMELLIA128-SHA256
  662	PSK-AES128-CBC-SHA256
```

以上只随机展示了30个ciphers，完整列表你可以重定向到文件或者通过more、less等命令慢慢翻看。

8.打印十六进制cipher ID（--show-cipher-ids）

此参数将显示服务器支持的加密套件的十六进制ID。这些ID是由IANA（Internet Assigned Numbers Authority）分配的，用于唯一标识每个加密套件。

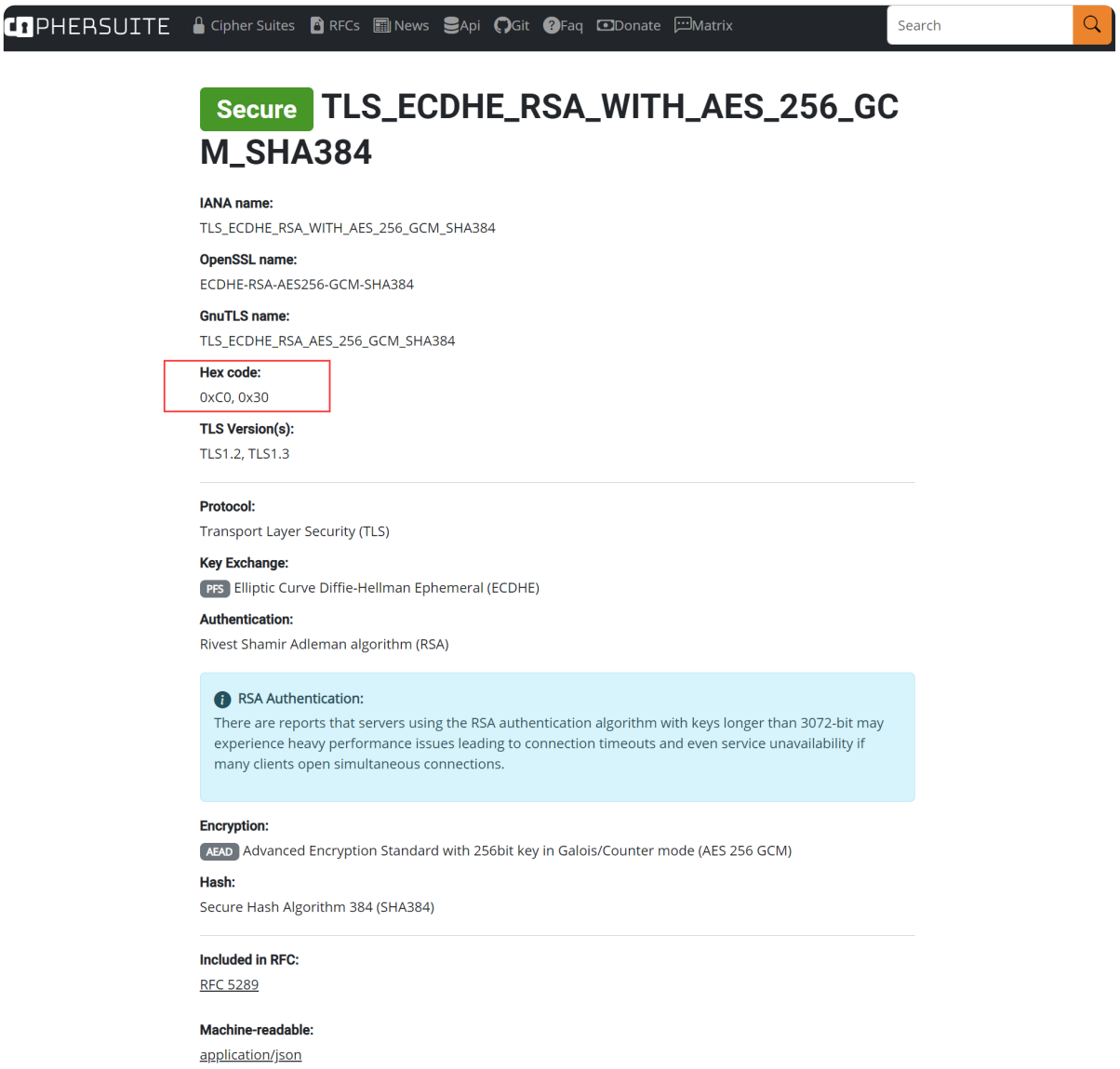
示例：

```
sslscan --show-cipher-ids <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-cipher-ids 192.168.1.81|sed -n '30,62p'
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits    0xC030 ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0xCCA8 ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0xC061 ECDHE-ARIA256-GCM-SHA384      Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits    0xC02F ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits    0xC060 ECDHE-ARIA128-GCM-SHA256      Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0xC028 ECDHE-RSA-AES256-SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0xC077 ECDHE-RSA-CAMELLIA256-SHA384  Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits    0xC027 ECDHE-RSA-AES128-SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits    0xC076 ECDHE-RSA-CAMELLIA128-SHA256  Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0xC014 ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits    0xC013 ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits    0x009D AES256-GCM-SHA384
Accepted  TLSv1.2  256 bits    0xC0A1 AES256-CCM8
Accepted  TLSv1.2  256 bits    0xC09D AES256-CCM
Accepted  TLSv1.2  256 bits    0xC051 ARIA256-GCM-SHA384
Accepted  TLSv1.2  128 bits    0x009C AES128-GCM-SHA256
Accepted  TLSv1.2  128 bits    0xC0A0 AES128-CCM8
Accepted  TLSv1.2  128 bits    0xC09C AES128-CCM
Accepted  TLSv1.2  128 bits    0xC050 ARIA128-GCM-SHA256
Accepted  TLSv1.2  256 bits    0x003D AES256-SHA256
Accepted  TLSv1.2  256 bits    0x00C0 CAMELLIA256-SHA256
Accepted  TLSv1.2  128 bits    0x003C AES128-SHA256
Accepted  TLSv1.2  128 bits    0x00BA CAMELLIA128-SHA256
Accepted  TLSv1.2  256 bits    0x0035 AES256-SHA
Accepted  TLSv1.2  256 bits    0x0084 CAMELLIA256-SHA
Accepted  TLSv1.2  128 bits    0x002F AES128-SHA
Accepted  TLSv1.2  128 bits    0x0041 CAMELLIA128-SHA
Preferred TLSv1.1  256 bits    0xC014 ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  128 bits    0xC013 ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  256 bits    0x0035 AES256-SHA
Accepted  TLSv1.1  256 bits    0x0084 CAMELLIA256-SHA
Accepted  TLSv1.1  128 bits    0x002F AES128-SHA
```

第五列展示为cipher的十六进制ID。

以第一行的cipher：**TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384** 为例，你可以在**cipher套件网站**找到它，并且也有显示套件的详细信息，16进制ID：



和sslscan输出的16进制ID完全吻合。

9.输出IANA/RFC cipher名称（--iana-names）

默认情况下，sslscan输出的cipher名称都为OpenSSL定义的名称，而IANA/RFC则是另一套名称标准。

还是以第一个输出cipher为例，不加参数的cipher名称为：**ECDHE-RSA-AES256-GCM-SHA384**

```
┌──(root㉿kali)-[~]
└─# sslscan 192.168.1.81|sed -n '30,62p'
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ARIA256-GCM-SHA384      Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ARIA128-GCM-SHA256      Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CAMELLIA256-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-CAMELLIA128-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  256 bits  AES256-CCM8
Accepted  TLSv1.2  256 bits  AES256-CCM
Accepted  TLSv1.2  256 bits  ARIA256-GCM-SHA384
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256
Accepted  TLSv1.2  128 bits  AES128-CCM8
Accepted  TLSv1.2  128 bits  AES128-CCM
Accepted  TLSv1.2  128 bits  ARIA128-GCM-SHA256
Accepted  TLSv1.2  256 bits  AES256-SHA256
Accepted  TLSv1.2  256 bits  CAMELLIA256-SHA256
Accepted  TLSv1.2  128 bits  AES128-SHA256
Accepted  TLSv1.2  128 bits  CAMELLIA128-SHA256
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.2  128 bits  AES128-SHA
Accepted  TLSv1.2  128 bits  CAMELLIA128-SHA
Preferred TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  256 bits  AES256-SHA
Accepted  TLSv1.1  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.1  128 bits  AES128-SHA
```

加上--iana-names的效果：

```
sslscan --iana-names <目标>
```

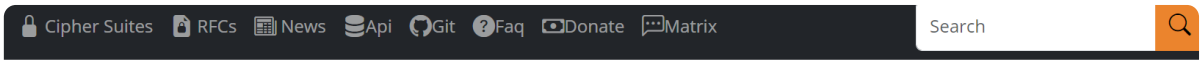
```
┌──(root㉿kali)-[~]
└─# sslscan --iana-names 192.168.1.81|sed -n '30,62p'
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384         Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256      Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_AES_256_GCM_SHA384
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_AES_256_CCM_8
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_AES_256_CCM
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_ARIA_256_GCM_SHA384
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_AES_128_GCM_SHA256
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_AES_128_CCM_8
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_AES_128_CCM
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_ARIA_128_GCM_SHA256
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_AES_256_CBC_SHA256
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_AES_128_CBC_SHA256
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_AES_256_CBC_SHA
Accepted  TLSv1.2  256 bits  TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_AES_128_CBC_SHA
Accepted  TLSv1.2  128 bits  TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
Preferred TLSv1.1  256 bits  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  128 bits  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  256 bits  TLS_RSA_WITH_AES_256_CBC_SHA
Accepted  TLSv1.1  256 bits  TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
Accepted  TLSv1.1  128 bits  TLS_RSA_WITH_AES_128_CBC_SHA
```

名称从ECDHE-RSA-AES256-GCM-SHA384变为了TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384，它们是同一个cipher，只不过不同标准的命名不一样。

输出十六进制ID，也是一样的：

```
┌──(root㉿kali)-[~]
└─# sslscan --show-cipher-ids 192.168.1.81|sed -n '30,31p'
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits    0xC030 ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253

┌──(root㉿kali)-[~]
└─# sslscan --show-cipher-ids --iana-names 192.168.1.81|sed -n '30,31p'
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits    0xC030 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384         Curve 25519 DHE 253
```

它们在ciphersuite.info中也有展示：



Secure **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384**

IANA name:
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

OpenSSL name:
ECDHE-RSA-AES256-GCM-SHA384

GnuTLS name:
TLS_ECDHE_RSA_AES_256_GCM_SHA384

Hex code:
0xC0, 0x30

TLS Version(s):
TLS1.2, TLS1.3

Protocol:
Transport Layer Security (TLS)

Key Exchange:
PFS Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)

Authentication:
Rivest Shamir Adleman algorithm (RSA)

以及在openssl.org的man文档中也有展示对应映射关系：

TLS v1.2 cipher suites

```
TLS_RSA_WITH_NULL_SHA256                NULL-SHA256

TLS_RSA_WITH_AES_128_CBC_SHA256         AES128-SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256         AES256-SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256         AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384         AES256-GCM-SHA384

TLS_DH_RSA_WITH_AES_128_CBC_SHA256      DH-RSA-AES128-SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA256      DH-RSA-AES256-SHA256
TLS_DH_RSA_WITH_AES_128_GCM_SHA256      DH-RSA-AES128-GCM-SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384      DH-RSA-AES256-GCM-SHA384

TLS_DH_DSS_WITH_AES_128_CBC_SHA256      DH-DSS-AES128-SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA256      DH-DSS-AES256-SHA256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256      DH-DSS-AES128-GCM-SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384      DH-DSS-AES256-GCM-SHA384

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256     DHE-RSA-AES128-SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256     DHE-RSA-AES256-SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256     DHE-RSA-AES128-GCM-SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384     DHE-RSA-AES256-GCM-SHA384

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256     DHE-DSS-AES128-SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256     DHE-DSS-AES256-SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256     DHE-DSS-AES128-GCM-SHA256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384     DHE-DSS-AES256-GCM-SHA384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256   ECDHE-RSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384   ECDHE-RSA-AES256-SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256   ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384   ECDHE-RSA-AES256-GCM-SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 ECDHE-ECDSA-AES256-SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ECDHE-ECDSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 ECDHE-ECDSA-AES256-GCM-SHA384

TLS_DH_anon_WITH_AES_128_CBC_SHA256     ADH-AES128-SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256     ADH-AES256-SHA256
TLS_DH_anon_WITH_AES_128_GCM_SHA256     ADH-AES128-GCM-SHA256
TLS_DH_anon_WITH_AES_256_GCM_SHA384     ADH-AES256-GCM-SHA384

RSA_WITH_AES_128_CCM                    AES128-CCM
RSA_WITH_AES_256_CCM                    AES256-CCM
DHE_RSA_WITH_AES_128_CCM                DHE-RSA-AES128-CCM
DHE_RSA_WITH_AES_256_CCM                DHE-RSA-AES256-CCM
RSA_WITH_AES_128_CCM_8                  AES128-CCM8
RSA_WITH_AES_256_CCM_8                  AES256-CCM8
DHE_RSA_WITH_AES_128_CCM_8              DHE-RSA-AES128-CCM8
DHE_RSA_WITH_AES_256_CCM_8              DHE-RSA-AES256-CCM8
ECDHE_ECDSA_WITH_AES_128_CCM            ECDHE-ECDSA-AES128-CCM
ECDHE_ECDSA_WITH_AES_256_CCM            ECDHE-ECDSA-AES256-CCM
ECDHE_ECDSA_WITH_AES_128_CCM_8          ECDHE-ECDSA-AES128-CCM8
ECDHE_ECDSA_WITH_AES_256_CCM_8          ECDHE-ECDSA-AES256-CCM8
```

10.打印每次握手耗时（--show-times）

此参数将显示每次握手所花费的时间（以毫秒为单位）。需要注意，每个密码只发出一个请求，并且Client Hello的大小不是恒定的，因此不适用于基准测试或性能测试中。

每个套件进行的握手时间都将会展示：

```
sslscan --show-times <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-times google.com|sed -n '25,60p'
  Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253 44ms
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253 47ms
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253 41ms
Preferred TLSv1.2  256 bits  ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253 60ms
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253 61ms
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253 62ms
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253 65ms
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253 67ms
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253 63ms
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253 55ms
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253 63ms
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253 53ms
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253 63ms
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256            62ms
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384            70ms
Accepted  TLSv1.2  128 bits  AES128-SHA                   60ms
Accepted  TLSv1.2  256 bits  AES256-SHA                   59ms
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA                 70ms
Preferred TLSv1.1  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253 70ms
Accepted  TLSv1.1  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253 67ms
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253 63ms
Accepted  TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253 69ms
Accepted  TLSv1.1  128 bits  AES128-SHA                   66ms
Accepted  TLSv1.1  256 bits  AES256-SHA                   66ms
Accepted  TLSv1.1  112 bits  DES-CBC3-SHA                 63ms
Preferred TLSv1.0  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253 60ms
Accepted  TLSv1.0  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253 65ms
Accepted  TLSv1.0  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253 57ms
Accepted  TLSv1.0  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253 61ms
```

11.只检查是否支持特定版本的SSL/TLS（--ssl/--tls）

参数范围有如下表格：

参数	含义
--ssl2	只检查是否开启SSLv2
--ssl3	只检查是否开启SSLv3
--tls10	只检查是否开启TLS1.0
--tls11	只检查是否开启TLS1.1
--tls12	只检查是否开启TLS1.2
--tls13	只检查是否开启TLS1.3
--tlsall	只检查TLS1.0、TLS1.1、TLS1.2、TLS1.3

按需选择参数，不一一列举，以TLS1.1为例：

```
sslscan --tls11 <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --tls11 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name 192.168.1.81

  SSL/TLS Protocols:
TLSv1.1    enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.1 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
```

或者检测tls所有版本（默认行为），则是：

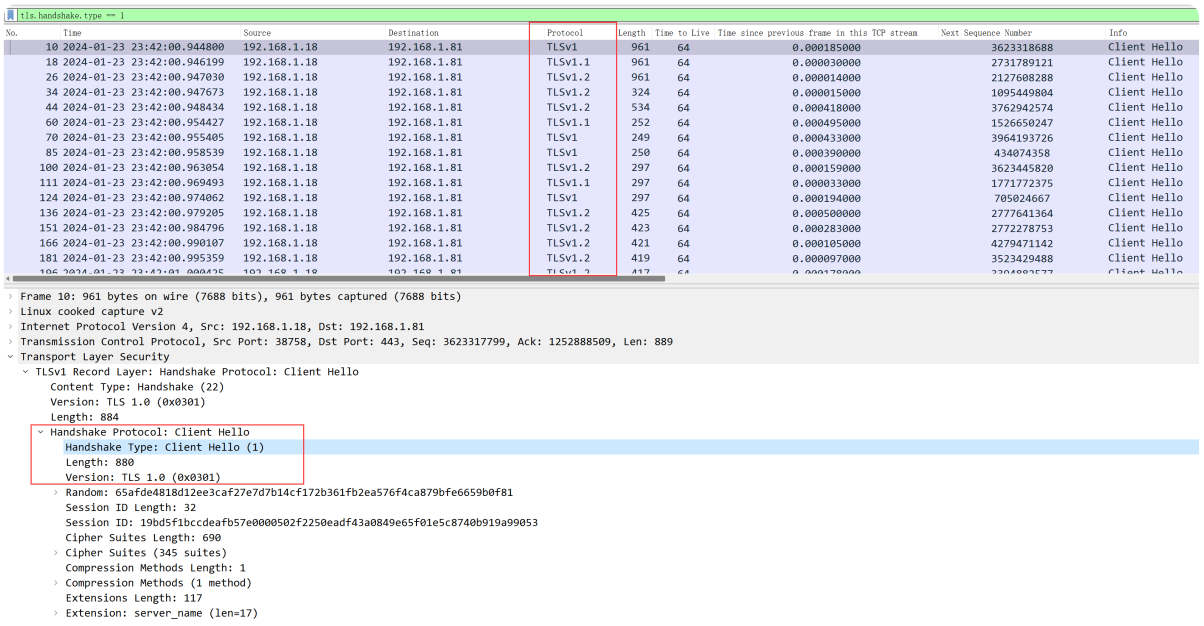
```
sslscan --tlsall <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --tlsall 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name 192.168.1.81

  SSL/TLS Protocols:
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
```

可以看到除了TLS1.3未启用，其它都是开启状态。

那么它是怎么判断哪些支持和不支持的呢?

通过抓包可以看到，sslscan在每个TLS握手阶段的ClientHello包里携带不同的的TLS版本，向服务端宣告客户端支持的TLS版本信息，每个版本都进行遍历，最终拿到能正常返回ServerHello的则视为对应版本的TLS协议是启用的：



12.指定私钥文件/密码、客户端证书（--pk/--pkpass/--certs）

需要使用私钥进行SSL/TLS握手的场景，通过此参数指定私钥文件即可，**--pk**即Private Key的含义，示例：

```
sslscan --pk=xxx.key <目标>
```

当私钥有密码时，则通过**--pkpass**来指定密码：

```
sslscan --pkpass=password <目标>
```

同理，在双向认证场景，需要指定客户端证书时，则为：

```
sslscan --certs=client.pem <目标>
```

13.不检测指定部分的信息（--no-<xxx>）

前面说过，sslscan不加任何参数时默认会扫描并展示8个部分的内容，如果不想扫描cipher套件信息，则可以加上**--no-ciphersuites**参数：

```
sslscan --no-ciphersuites <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --no-ciphersuites 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name 192.168.1.81

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed


  Server Key Exchange Group(s):
TLSv1.2  128 bits  secp256r1 (NIST P-256)
TLSv1.2  192 bits  secp384r1 (NIST P-384)
TLSv1.2  260 bits  secp521r1 (NIST P-521)
TLSv1.2  128 bits  x25519
TLSv1.2  224 bits  x448

  SSL Certificate:
Signature Algorithm: sha384WithRSAEncryption
RSA Key Strength:    2048

Subject:  blog.linux-code.com
Altnames: DNS:blog.linux-code.com
Issuer:   TrustAsia RSA DV TLS CA G2

Not valid before: Jun 10 00:00:00 2023 GMT
Not valid after:  Jun  9 23:59:59 2024 GMT
```

此时输出结果已经简略了不少。

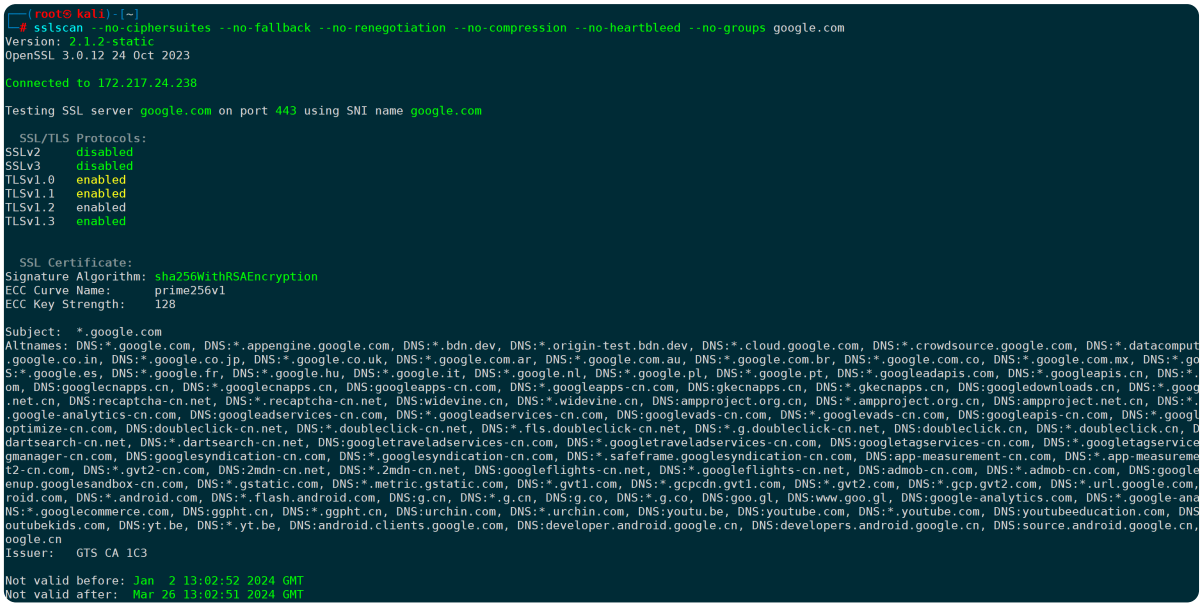
同理，如下这些参数都能定义哪些信息不扫描，进而最大化节省扫描耗时：

参数	含义
--no-ciphersuites	不扫描加密套件信息
--no-fallback	不扫描是否支持TLS Fallback

参数	含义
--no-renegotiation	不检测是否支持TLS重协商
--no-compression	不检测是否支持TLS压缩
--no-heartbleed	不扫描是否存在OpenSSL心脏滴血漏洞
--no-groups	不枚举密钥交换组
--no-cipher-details	隐藏NIST EC曲线名称和EDH/RSA密钥长度

以上参数可以叠加使用，比如不扫描cipher套件信息、fallback、tls重协商、tls压缩、密钥交换组等，可以是：

```
sslscan --no-ciphersuites --no-fallback --no-renegotiation --no-compression --no-heartbleed --no-groups <目标>
```



此时输出结果已经简略了许多，只扫描想要的信息，大大提高了扫描速度。

14.以文件形式指定目标列表（--targets）

有多个目标需要扫描时，可将目标写入到一个文件内，一行一个，格式为 `host:port` ，不指定端口默认为443。

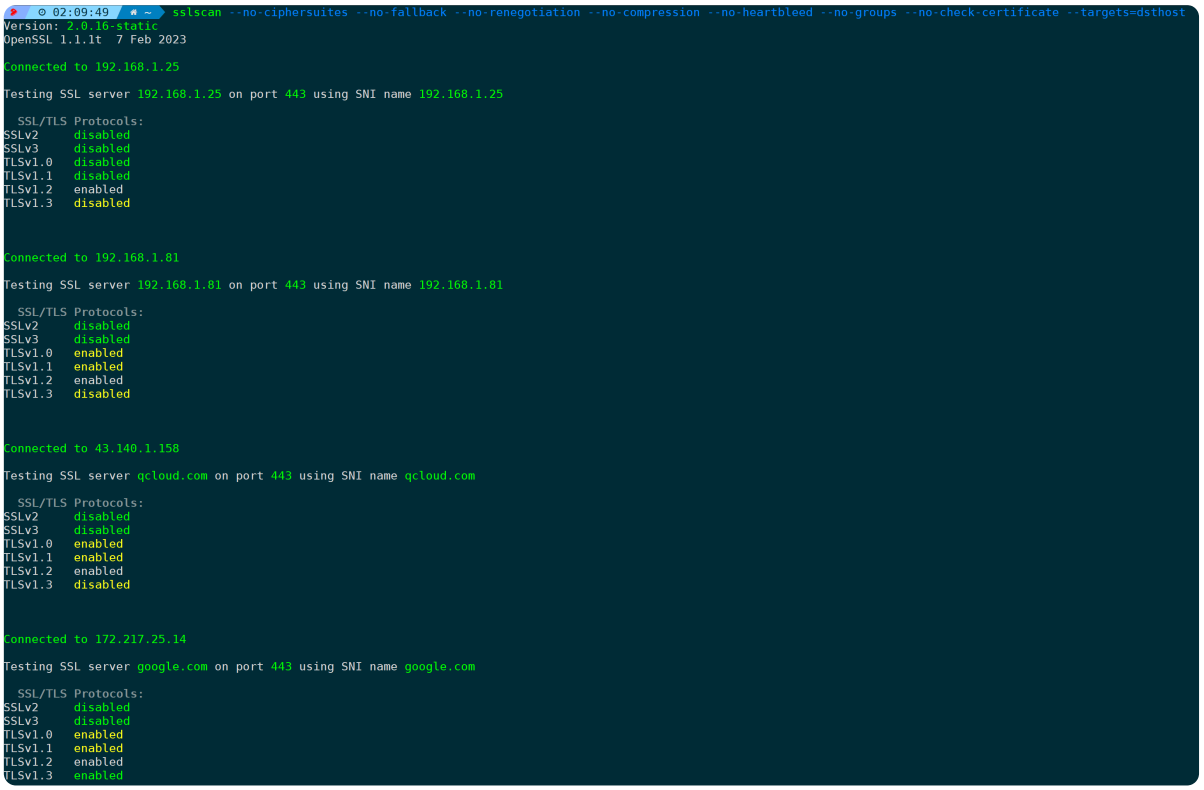
提到这，务必要说一下，很多人有一个误区：**443一定是https、80端口一定是http**。这仅仅是作为默认端口的存在，可以任意指定没有限制，你理解的仅仅是默认，而非自定义端口。只要你乐意，HTTP可以是443，HTTPS可以是80，甚至可以是范围内的任何端口，比如URL为：**https://domain.com:8443**、**http://domain.com:12345**，都是合法URL，重要的是这些端口在web服务器上的监听业务是HTTP还是HTTPS。

用法示例：

```
$ cat dsthost
192.168.1.25
192.168.1.81
qcloud.com:443
google.com
$ sslscan --targets=dsthost
```

如果只想扫描指定的内容，比如只需要检测目的端支持的SSL/TLS版本，那么把前面提到过的不扫描参数加上，输出结果会简洁很多，扫描速度也会成倍增加：

```
sslscan --no-ciphersuites --no-fallback --no-renegotiation --no-compression --no-heartbleed --no-groups --no-check-certificate --targets=<dsthost>
```



15.枚举服务器证书的签名算法（--show-sigs）

默认情况下不展示此部分内容，如需枚举服务器证书支持的签名算法，可加上此参数：

```
sslscan --show-sigs <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --show-sigs --sni-name=data.linux-code.com 192.168.1.81 | grep -A 8 'Server Signature'
  Server Signature Algorithm(s):
TLSv1.2  rsa_pkcs1_sha1
TLSv1.2  rsa_pkcs1_sha224
TLSv1.2  rsa_pkcs1_sha256
TLSv1.2  rsa_pkcs1_sha384
TLSv1.2  rsa_pkcs1_sha512
TLSv1.2  rsa_pss_rsae_sha256
TLSv1.2  rsa_pss_rsae_sha384
TLSv1.2  rsa_pss_rsae_sha512
```

同时，使用浏览器访问时，会显示TLS协商时使用的签名算法：

16.启用SSL错误的解决方案（--bugs）

当SSL证书存在错误时，**--bugs**可提供建议的解决方案：

```
sslscan --bugs <目标>
```

17.设置socket超时时间（--timeout）

不指定的情况下默认是3秒，对于无法响应sslscan不理解的cipher套件很有效。

指定超时时间为1s，可以是：

```
sslscan --tmeout=1 <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --timeout=1 192.168.1.6
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.6

Testing SSL server 192.168.1.6 on port 443 using SNI name 192.168.1.6

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-GCM-SHA384     DHE 2048 bits
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-SHA256         DHE 2048 bits
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  DHE-RSA-AES128-GCM-SHA256     DHE 2048 bits
Accepted  TLSv1.2  128 bits  DHE-RSA-AES128-SHA256         DHE 2048 bits

  Server Key Exchange Group(s):
TLSv1.2  128 bits  secp256r1 (NIST P-256)

  SSL Certificate:
Signature Algorithm: sha384WithRSAEncryption
RSA Key Strength:    2048

Subject:  bmc.linux-code.com
Altnames: DNS:bmc.linux-code.com
Issuer:   TrustAsia RSA DV TLS CA G2

Not valid before: Mar 12 00:00:00 2022 GMT
Not valid after:  Mar 12 23:59:59 2023 GMT
```

18.设置初始超时时间（--connect-timeout）

当对端主机响应很慢的时候，不想等太长时间，则可通过此参数设定超时时间，不指定的情况下默认为75秒。

设置0.01秒和1秒为例：

```
sslscan --connect-timeout=0.01 <目标>
sslscan --connect-timeout=1 <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --connect-timeout=0.01 192.168.1.6
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

ERROR: Could not open a connection to host 192.168.1.6 (192.168.1.6) on port 443 (connect: Timed out).

┌──(root㉿kali)-[~]
└─# sslscan --connect-timeout=1 192.168.1.6
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.6

Testing SSL server 192.168.1.6 on port 443 using SNI name 192.168.1.6

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Session renegotiation not supported
```

可以看到，设置为0.01秒（10ms）时，sslscan报连接对端主机超时，时间太短连接无法成功建立起来，1秒的情况下则正常建联。以上仅作为极限的模拟测试，实际使用建议按需调整超时时间。

19.将输出结果报错为XML文件（--xml）

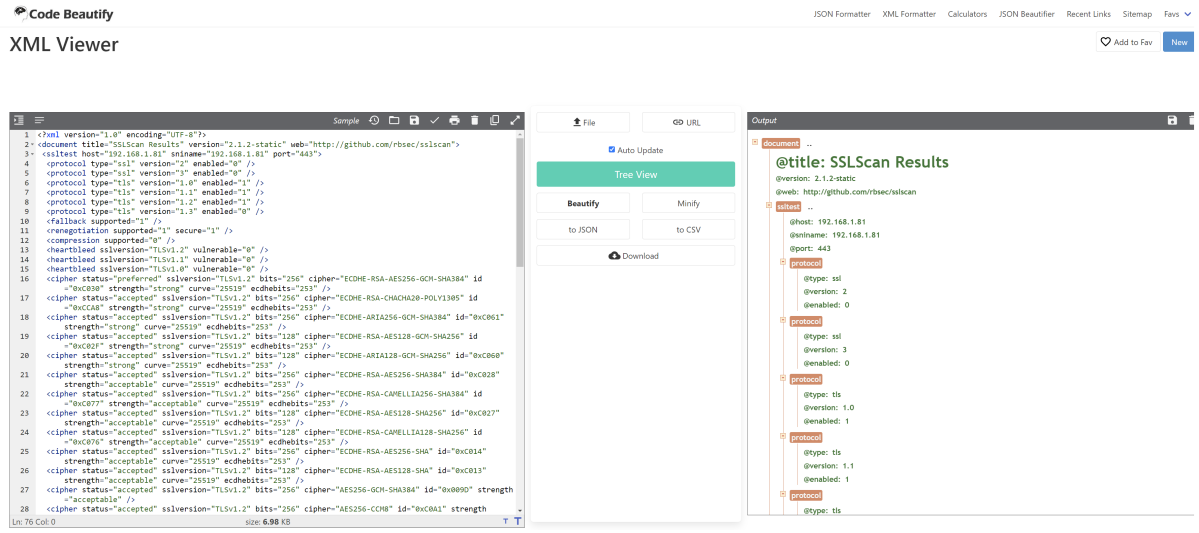
要将输出信息输出为xml格式文件时，可以是：

```
sslscan --xml=file.xml <目标>
```

```
┌──(root㉿kali)-[~]
└─# sslscan --sni-name data.linux-code.com --xml=test.xml 192.168.1.81
Version: 2.1.2-static
OpenSSL 3.0.12 24 Oct 2023

Connected to 192.168.1.81

Testing SSL server 192.168.1.81 on port 443 using SNI name 192.168.1.81

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed
TLSv1.1 not vulnerable to heartbleed
TLSv1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384    Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305    Curve 25519 DHE 253
```

此输出参数可以配合上面所讲的所有参数搭配使用，它不仅仅在屏幕上输出结果，也会将结果输出到指定的xml文件内。为什么需要导出XML的功能？因为有利于进行二次统计分析和检索，xml更为方便。

xml文件可以使用浏览器打开，或其它支持此格式文件的专用软件打开，或者上传到在线xml文档解析器上打开或进行其它文件格式的转换，比如 codebeautify.org/xmlviewer 支持解析xml并进行格式转换：



20.更为详细的输出（--verbose）

需要更详细的输出报告时，**--verbose**正好符合，会尽可能详细的输出更多信息：

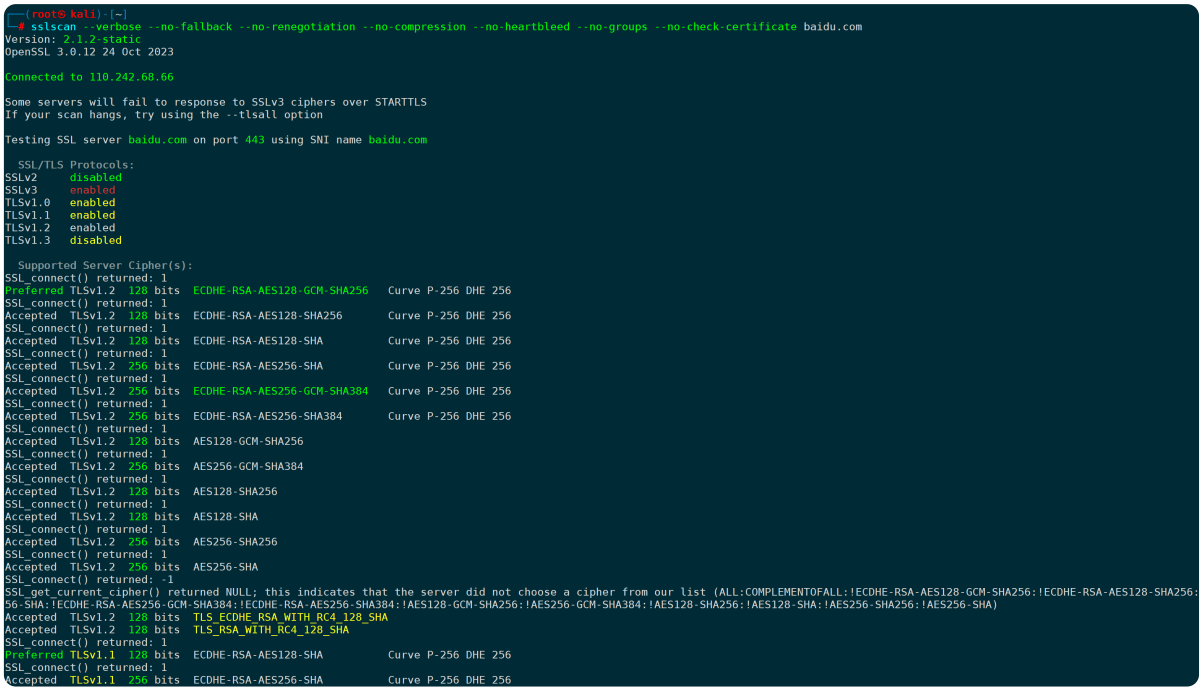
```
sslscan --verbose <目标>
```

主要注意：

- 不能使用-v替代，-v并不是--verbose的简写；
- --verbose只能加一次，加多次效果如同加一次。

比如详细输出cipher套件的信息，但其它阶段的信息不需要：

```
sslscan --verbose --no-fallback --no-renegotiation --no-compression --no-heartbleed --no-groups --no-check-certificate <目标>
```



图中可以看到，多了一长串提示信息：**SSL_get_current_cipher() returned NULL; this indicates that the server did not choose a cipher from our list (ALL:COMPLEMENTOFALL:!ECDHE-RSA-AES128-GCM-SHA256:!ECDHE-RSA-AES128-SHA256:!ECDHE-RSA-AES128-SHA:!ECDHE-RSA-AES256-SHA:!ECDHE-RSA-AES256-GCM-SHA384:!ECDHE-RSA-AES256-SHA384:!AES128-GCM-SHA256:!AES256-GCM-SHA384:!AES128-SHA256:!AES128-SHA:!AES256-SHA256:!AES256-SHA)**

在『前言』部分说过，黄色字体表示弱密码套件，通过详细的输出日志可以看到**SSL_get_current_cipher**函数返回了一个空值，因为对端的cipher套件都不在上述这一长串的套件里，则认为是弱密码套件，字体标黄，**SSL_get_current_cipher**函数返回NULL，**SSL_connect**函数返回-1。

五、总结

sslscan作为业内一款强大的证书扫描工具，能够帮助我们检测 SSL/TLS 证书的版本信息、cipher套件、密钥交换组、证书颁发机构、OCSP服务器、证书有效期等等，对于确保网络安全和数据保护至关重要，在总结部分不一一列举。

sslscan的易用性使其成为网络安全专家的必备工具。其命令行界面简洁明了，参数丰富，可以根据用户的特定需求进行定制。无论是进行基本的服务器扫描，还是深入分析特定的密码套件或证书信息，都能提供强大的支持。期间也通过抓包分析了sslscan是如何拿到扫描结果。

总的来说，使用sslscan可以大大提升网站和应用程序的安全性，保护用户的敏感信息和隐私不被泄露，比如弱密码套件、过时的协议版本、证书吊销等，提前确认安全隐患，防患于未然。