Universitat Politècnica de Catalunya

Criptografía

Práctica 4

Joan Marc Pastor Mackenzie

1. Conexión Segura **www.wikipedia.org** (ECDHE-ECDSA)

PASOS PREVIOS

Establecemos una conexión segura con www.wikipedia.org guardando las *SSLPreMasterKeys* utilizadas en un fichero para posteriormente analizar el tráfico desencriptado. Alimentamos el fichero de llaves a Wireshark y procedemos a registrar el protocolo *handshake* de SSL/TLS. Obtenemos el siguiente *log*:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.101	172.217.17.19	TCP	55	53153 → 443 [ACK] Seq=1 Ack=1 Win=507 Len=1 [TCP segment of a reassembled PDU]
32	0.575744	192.168.1.101	91.198.174.192	TLSv1.2	571	Client Hello
35	0.620252	91.198.174.192	192.168.1.101	TLSv1.2	1506	Server Hello
37	0.620255	91.198.174.192	192.168.1.101	TLSv1.2	1223	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
39	0.623019	192.168.1.101	91.198.174.192	TLSv1.2	139	Client Key Exchange, Change Cipher Spec, Finished
40	0.623193	192.168.1.101	91.198.174.192	HTTP2	139	Magic, SETTINGS[0], WINDOW_UPDATE[0]
41	0.623419	192.168.1.101	91.198.174.192	HTTP2	490	HEADERS[1]: GET /

El fichero de la conexión se puede encontrar en: *1/wikipedia_connection_log.pcapng*.

En el protocolo de *handshake*, nos aseguramos que estamos utilizando los algoritmos ECDHE-ECDSA mirando la propiedad "*Cipher Suite*" del paquete "*Server Hello*". Obtenemos el siguiente *Cipher Suite*:

```
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
```

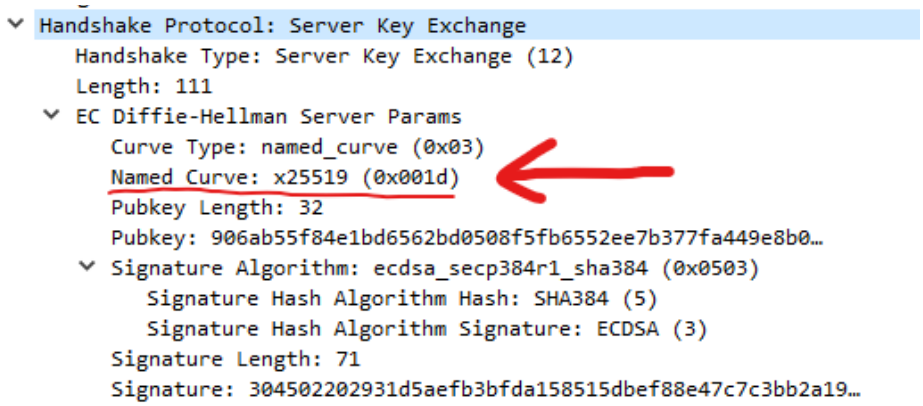
Ficheros Generados:

- Wireshark *log* de conexión desencriptada: *1/wikipedia_connection_log.pcapng*
- Detalles de conexión (openssl): *1/wikipedia_connection.txt*
- Certificado: *1/wikipedia_certificate.crt*
- Certificado en formato texto: *1/wikipedia_certificate.txt*
- Sage Script en Jupyter: *1/script_wikipedia.ipynb*
- Sage Script logs exportados a HTML: *1/script_wikipedia.html*

PREGUNTAS:

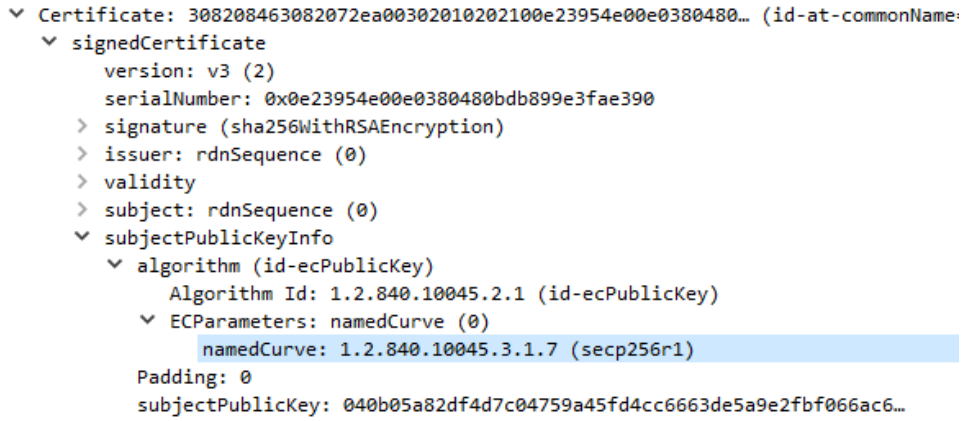
(a) Dar los nombres de las curvas que se usan para acordar la clave DH y la clave pública del servidor.

Por tal encontrar la curva de **DH**, en el protocolo *handshake*, buscamos el paquete "*Server Key Exchange*" y obtenemos el nombre de la curva:



Curva Diffie-Hellman: **Curve25519**

Para encontrar la curva utilizada en la **clave pública**, miramos las propiedades del certificado:



Curva Clave Pública: **NIST P-256**

Apartados (b), (c), (d) y (e) están demostrados en los siguientes ficheros:

Sage Script running on Jupyter: *1/script_wikipedia.ipynb*
Sage Script logs saved as HTML: *1/script_wikipedia.html*

2. Conexión Segura **google.com** (ECDHE-ECDSA)

PASOS PREVIOS

Establecemos una conexión segura con google.com desde el navegador guardando las *SSLPreMasterKeys* utilizadas en un fichero para posteriormente analizar el tráfico desencriptado. Alimentamos el fichero de llaves a Wireshark y procedemos a registrar el protocolo *handshake* de SSL/TLS. Obtenemos un *log* con el siguiente *cipher suite*:

```
▾ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 98
    Version: TLS 1.2 (0x0303)
  ▸ Random: 5e03fed3e58c14292065cff9648b2fa3930fc3c8b9572e6d…
    Session ID Length: 32
    Session ID: be1f000033af1c00481b657897b26d7f6b1c25c460ade028…
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Compression Method: null (0)
```

Como podemos observar, el Cipher Suite por defecto parece ser ECDHE_RSA en vez del ECDHE_ECDSA deseado. Por tal de forzar al servidor el uso de ECDSA, realizaremos la conexión a través de *openssl* restringiendo el *Client Hello* al uso de *Cipher Suites* que contengan ECDSA:

```
> openssl s_client -connect google.com:443 -cipher 'ECDSA' -tls1_2
```

Nota: Este comando solo funciona con versiones de *openssl* igual o superiores a v1.1.1.

Registramos las transacciones del handshake y lo desciframos proporcionando a Wireshark la *Session ID* y la *Secret Key* que nos facilita openssl. Obtenemos el siguiente volcado:

	Time	Source	Destination	Protocol	Length	Info
11	0.224887	192.168.1.101	216.58.211.46	TLSv1.2	243	Client Hello
13	0.271873	216.58.211.46	192.168.1.101	TLSv1.2	1484	Server Hello
16	0.272692	216.58.211.46	192.168.1.101	TLSv1.2	887	Certificate, Server Key Exchange, Server Hello Done
17	0.278990	192.168.1.101	216.58.211.46	TLSv1.2	139	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
18	0.294340	216.58.211.46	192.168.1.101	TLSv1.2	330	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

```
▾ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 59
    Version: TLS 1.2 (0x0303)
  ▸ Random: 5e03acff6295b2d4f848b9d38da2284fa3fcf9bf81499dc8…
    Session ID Length: 0
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
    Compression Method: null (0)
```

Ahora ya podemos trabajar sobre las preguntas con nuestra conexión ECDHE-ECDSA.

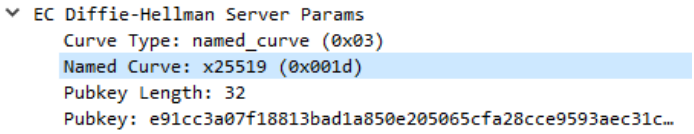
Ficheros Generados:

- Wireshark *log* de conexión desencriptada: *2/google_connection_log.pcapng*
- Detalles de conexión (*openssl*): *2/google_connection.txt*
- Certificado: *2/google_certificate.crt*
- Certificado en formato texto: *2/google_certificate.txt*
- Sage Script en Jupyter: *2/script_google.ipynb*
- Sage Script logs exportados a HTML: *2/script_google.html*

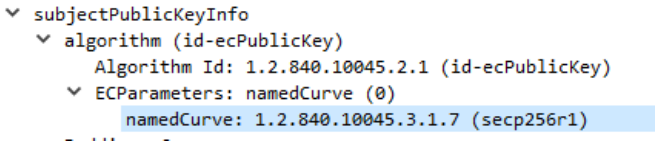
PREGUNTAS:

(a) Dar los nombres de las curvas que se usan para acordar la clave DH y la clave pública del servidor.

Curva de Diffie-Hellman: **Curve25519**



Curva de Clave Pública: **NIST P-256**



Apartados (b), (c), (d) y (e) están demostrados en los siguientes ficheros:

Sage Script running on Jupyter: *2/script_google.ipynb*
Sage Script logs saved as HTML: 2*/script_google.html*

3. Conexión Segura www.fib.upc.edu

Con la ayuda de Wireshark, se ha registrado el TLS handshake de una conexión segura a www.fib.upc.edu, se han extraído sus certificados y se han generado los siguientes ficheros:

Certificados Generados:

- 3/fib.upc.edu.cer
- 3/fib.upc.edu.pem
- 3/fib.upc.edu.txt (Certificado versión texto)
- 3/TERENA.cer
- 3/TERENA.pem
- 3/DigiCert.cer
- 3/DigiCert.pem

Jerarquía de certificados:

- DigiCert
  - TERENA SSL CA 3
    - www.fib.upc.edu.com

Por tal de extraer el CRL, se ha inspeccionado el certificado *3/fib.upc.edu.txt* en busca del enlace al CRL. Una vez localizado (http://crl3.digicert.com/TERENASSLCA3.crl), al abrirlo en un explorador, el CRL se descargará. Éste se puede encontrar en *3/TERENASSLCA3.crl*.

PREGUNTAS:

(a) En el certificado se da un punto de distribución del CRL de la autoridad certificadora. ¿Cuántos certificados revocados contiene la CRL?

El siguiente comando nos informa de la cantidad de certificados revocados por la CRL:

```
> openssl crl -text -noout -in TERENASSLCA3.crl -inform DER | grep -c
"Serial Number:"
10125
```

Obtenemos un total de **10125** certificados revocados.

(b) ¿Cuál es el estatus del certificado i hasta cuando es válido este estatus?

Por tal de verificar la validez del certificado, realizamos los siguientes pasos:

- Comprobar que el certificado de la FIB tenga como CN = www.fib.upc.edu
- Comprobar que la fecha actual se encuentre entre las fechas de validez definidas en el certificado
- Verificar el certificado de TERENA con el certificado raiz DigiCert:
  ```
  > openssl verify -CAfile 3_DigiCert.pem 2_TERENA.pem
  OK
  ```
- Verificar el certificado fib.upc.edu con el certificado de TERENA:
  ```
  > openssl verify -CAfile 3_DigiCert.pem -untrusted 2_TERENA.pem
  1_fib.upc.edu.pem
  OK
  ```
- Comprobar que el *serial number* del certificado de www.fib.upc.edu no esté presente en la CRL:
  ```
  > openssl crl -text -noout -in TERENASSLCA3.crl -inform DER | grep
  -c "064dda5068d4c7c23ac379caa35930a3"
  0
  ```

Ya que se cumplen las condiciones de la página anterior, el certificado de www.fib.upc.edu es válido en este preciso momento.

Dejará de ser válido si se revoca el certificado o si se superan las fechas de validad definidas en el certificado. En nuestro caso:

```
Validity
      Not Before: Feb 28 00:00:00 2017 GMT
      Not After : Mar  4 12:00:00 2020 GMT
```