Perfect forward secrecy (PFS)

Akash Patel (SJSU)
What is Perfect Forward Secrecy?[1]

- Perfect forward secrecy (PFS) is a property of the key-agreement protocol that ensures that session key used to encrypt the data will not be compromised even if long term private key compromised in future.

- The idea is to not to use single key (e.g. private key) to generate all the session keys. Also, same material or method should not use to generate session key.
Why PFS is required? [2]

- Lets say Alice and Bob want to safely communicate over network using Symmetric-key cryptography.

- Suppose Alice generates Session key using her private key and encrypt data using session key. She sent the resulting cipher to Bob. Trudy wants to acquire this information. She can record all the messages communicate over network. She does not know Alice’s private key so she cannot know how session key is derived and how data is encrypted.

- Later, Trudy breaks Alice computer and obtain her private key K. Trudy can now decrypt Session key and all the previously recorded messages.

- Above situation is the motivational factor for the Perfect forward secrecy.

- The main problem is, private key is used for two purposes: authentication and encryption. Authentication only matters while the communication is established, but encryption is expected to last for years.
How PFS works?[3]

- PFS periodically creates a new session key value based on values supplied by both parties in the exchange. Because both parties contribute a random value known only to them, each new key generated is dissimilar to previously created keys.

- Even if a third party managed to intercept a private key, that party can only use the intercepted key for a short time.

- The newly created key is not used same method or material as used in the previously intercepted key, the third party must begin a new brute force calculation to guess the new key value

- Ephemeral Diffie-Hellman key exchange and Elliptic curve Diffie-Hellman are two most elegant approach for PFS.

- In this protocol, $g$ and $p$ are public. Alice chooses a secret exponent $a$ and Bob a secret exponent $b$.

- Then Alice sends $g^a \mod p$ to Bob and Bob sends $g^b \mod p$ to Alice. Alice and Bob can then each compute the shared secret $g^{ab} \mod p$.

- In order to prevent the MiM attack, Alice and Bob can use their shared symmetric key $K_{AB}$ to encrypt the Diffie-Hellman exchange.

- Then to attain PFS, all that is required is that, once Alice has computed the shared session key $K_s = g^{ab} \mod p$, she must forget her secret exponent $a$ and, similarly Bob must forget his secret exponent $b$. 

Akash(008638799)
How to set up PFS?[5]

• Session key is key factor to determine whether connection has perfect forward secrecy is determined by how the session key is derived. And how the session key is derived is determined by the cipher suite in use.

• In the beginning SSL handshake, the client sends a list of supported cipher suites.

• The server then picks one of the cipher suites, based on a ranking and inform client which cipher suite will be used from next onwards communication.

• Last step determines whether or not the further connection will have perfect forward secrecy

• Cipher suites that use ephemeral Diffie-Hellman (DHE) or the elliptic-curve variant (ECDHE) will have perfect forward secrecy
How to set up PFS on Apache Server? [6]

- Web server probably has a cipher suite configuration in its SSL configuration. There are two relevant options: first, the cipher suites that you want your server to use, and second, how the server picks the cipher suite. The order of cipher suite also matters.

- For the Apache server Perfect Forward Secrecy requires Apache 2.3.3 or higher.

- Here is one example configuration for mod_ssl that will work to enable Perfect Forward Secrecy:

```
SSLProtocol +TLSv1 +TLSv1.1 +TLSv1.2
SSLHonorCipherOrder On
SSLCipherSuite ECDHE-RSA-RC4-SHA:ECDHE-RSA-AES128-SHA:AES128-SHA:RC4-SHA
```
Challenges for PFS\textsuperscript{[7]}

- DHE is significantly slower due to additional calculations.
- Web site operators tend to disable DHE suites in order to achieve better performance.
- Not all browsers support all the necessary suites.
- ECDHE too is slower, but not as much as DHE but ECDHE algorithms are relatively new and not as widely supported.
How to verify PFS?

There are many ways to verify PFS, I have listed below 3 ways to do that. I have used www.yioop.com as my test site.

1. OPENSSL Utility
2. SSL Scan Utility
OPENSSL Utility

OpenSSL> s_client -connect 173.13.143.74:443
Server certificate
-----BEGIN CERTIFICATE-----
MIIF3DCClCzggAgQIBAgIDHlAJBgUrDgMBAgEGCSqGCCIGAoIBAQ5gR0fQ6ZJ
aWbZC+A2Y0AgEAMBIhGkOYBhUZQUF6B3+/P0U6884uv0v31hO4sOgD8u83Pf
j8/bjxQpk+l5J5dOZ9nKl6BhZyZB1QZG/njJ3R2wD/9xUOw+6k8QOZ1q8J
-----END CERTIFICATE-----

subject=/OU=Domain Control Validated/CN=seek quarry.com
issuer=/C=US/ST=Arizona/L=Scottsdale/O=Go Daddy.com, Inc./OU=http://certificates

godaddy.com/repository/CN=Go Daddy Secure Certification Authority

No client certificate CA names sent

SSL handshake has read 578 bytes and written 322 bytes

New, TLSv1/SSLv3. Cipher is DHE-RSA-AES256-SHA
Server public key is 2048 bit

Compression: NONE

Expansion: NONE

SSL-Session:
  Protocol : TLSv1
  Cipher : DHE-RSA-AES256-SHA
  Server ID-FF4789A502256D4E012DB8F4E0639939C67B09C11E49974D81B41BFDD45
  Session-ID-ctx:
  Master-Keys: 9653E5AEEF9332E4755E4721D944B70327DF11DB4C7F08B3BB08D68D5B4F1A
  Key-Arg : None
  Start Time: 138186437
  Timeout : 300 (sec)
SSL Scan Utility

C:\Users\Akash\Desktop\SSLScan-1.8.2-win-r7>SSLScan --no-failed www.yioop.com

sslscan

Version 1.8.2-win
http://www.titania.co.uk
Copyright Ian Ventura-Whiting 2009
Compiled against OpenSSL 0.9.8m 25 Feb 2010

Testing SSL server www.yioop.com on port 443

Supported Server Cipher(s):
Accepted SSLv3 256 bits DHE-RSA-AES256-SHA
Accepted SSLv3 256 bits AES256-SHA
Accepted SSLv3 128 bits DHE-RSA-AES128-SHA
Accepted SSLv3 128 bits AES128-SHA
Accepted SSLv3 168 bits EDH-RSA-DES-CBC3-SHA
Accepted SSLv3 168 bits DES-CBC3-SHA
Accepted SSLv3 128 bits RC4-SHA
Accepted SSLv3 128 bits RC4-MD5
Accepted TLSv1 256 bits DHE-RSA-AES256-SHA
Accepted TLSv1 256 bits AES256-SHA
Accepted TLSv1 128 bits DHE-RSA-AES128-SHA
Accepted TLSv1 128 bits AES128-SHA
Accepted TLSv1 168 bits EDH-RSA-DES-CBC3-SHA
Accepted TLSv1 168 bits DES-CBC3-SHA
Accepted TLSv1 128 bits RC4-SHA
Accepted TLSv1 128 bits RC4-MD5

Preferred Server Cipher(s):
SSLv3 256 bits DHE-RSA-AES256-SHA
TLSv1 256 bits DHE-RSA-AES256-SHA

Akash(008638799)
**SSL Lab Test**

You are here: Home > Projects > SSL Server Test > yioop.com

**SSL Report: yioop.com (173.13.143.74)**

Assessed on: Tue Oct 15 19:43:19 UTC 2013 | Clear cache

<table>
<thead>
<tr>
<th>Cipher Suites (sorted by strength; the server has no preference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_RC4_128_MD5 (0x4)</td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_SHA (0x5)</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 1024 bits (p: 128, g: 1, Ys: 128) FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_SEED_CBC_SHA (0x96)</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_SEED_CBC_SHA (0x9a) DH 1024 bits (p: 128, g: 1, Ys: 128) FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16) DH 1024 bits (p: 128, g: 1, Ys: 128) FS</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA (0x35)</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 1024 bits (p: 128, g: 1, Ys: 128) FS</td>
</tr>
</tbody>
</table>

**Forward Secrecy**

With some browsers (more info)

Akash(008638799)
References


2. Section 9.3.4 of Information Security Principles and practice by Dr. Mark Stamp, Published by JohnWiley & Sons, Inc


4. Section 9.3.4 of Information Security Principles and practice by Dr, Mark Stamp

