Guide to Deploying Diffie-Hellman for TLS

Our study finds that the current real-world deployment of Diffie-Hellman is less secure than previously believed. This page explains how to properly deploy Diffie-Hellman on your server.

We have three recommendations for correctly deploying Diffie-Hellman for TLS:

1. **Disable Export Cipher Suites.** Even though modern browsers no longer support export suites, the FREAK (https://freakattack.com/) and Logjam (logjam.html) attacks allow a man-in-the-middle attacker to trick browsers into using export-grade cryptography, after which the TLS connection can be decrypted. Export ciphers are a remnant of 1990s-era policy that prevented strong cryptographic protocols from being exported from United States. No modern clients rely on export suites and there is little downside in disabling them.
2. **Deploy (Ephemeral) Elliptic-Curve Diffie-Hellman (ECDHE).** Elliptic-Curve Diffie-Hellman (ECDH) key exchange avoids all known feasible cryptanalytic attacks, and modern web browsers now prefer ECDHE over the original, finite field, Diffie-Hellman. The discrete log algorithms we used to attack standard Diffie-Hellman groups do not gain as strong of an advantage from precomputation, and individual servers do not need to generate unique elliptic curves.
3. **Use a Strong, Diffie Hellman Group.** A few 1024-bit groups are used by millions of servers, which makes them an optimal target for precomputation, and potential eavesdropping. Administrators should use 2048-bit or stronger Diffie-Hellman groups with "safe" primes.

Steps (1) and (2) can be accomplished simultaneously by configuring your server to only use modern, secure cipher suites. We describe how to define modern ciphers and to generate a Diffie-Hellman group for popular servers below.

You can test your server using the tool below, or by using the Qualsys SSL Server Test (https://ssllabs.com/ssltest/). If you have information on how to patch other software, please let us know (mailto:weakdh-team@umich.edu).

Using a Strong DH Group

You will first need to generate a new Diffie-Hellman group, regardless of the server software you use. Modern browsers, including Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer have increased the minimum group size to 1024-bit. We recommend that you generate a 2048-bit group. The simplest way of generating a new group is to use OpenSSL:

```
openssl dhparam -out dhparams.pem 2048
```

Common Server Products

For each sever product, we provide two configuration options: (1) safe cipher suites that you should use, and (2) how to specify the Diffie Hellman parameters you generated above. The selected ciphers are based on Mozilla's Moderate Cipher List.

Apache HTTP Server (mod_ssl)

SSL parameters can globally be set in `httpd.conf` or within specific virtual hosts.

Cipher Suites

Disable support for SSLv2 and SSLv3 and enable support for TLS, explicitly allow/disallow specific ciphers in the given order :

```
SSLProtocol             all -SSLv2 -SSLv3

SSLCipherSuite          ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA
384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RS
A-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA
384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AE
S128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA2
56:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!e
NULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

SSLHonorCipherOrder     on
```

DH Parameters

In newer versions of Apache (2.4.8 and newer) and OpenSSL 1.0.2 or later, you can directly specify your DH params file as follows:

```
SSLOpenSSLConfCmd DHParameters "{path to dhparams.pem}"
```

If you are using Apache with LibreSSL, or Apache 2.4.7 and OpenSSL 0.9.8a or later, you can append the DHparams you generated earlier to the end of your certificate file.

Reload configuration

```
sudo service apache2 reload
```

nginx

To be placed in the website configuration `server` block in `/etc/nginx/sites-enabled/default`:

Cipher Suites

```
ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-E
CDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SH
A256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-E
CDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DH
E-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-G
CM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPOR
T:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA';

ssl_prefer_server_ciphers on;
```

DH parameters

```
ssl_dhparam {path to dhparams.pem}
```

Reload configuration

```
sudo nginx -s reload
```

Microsoft IIS

1. Open the Group Policy Object Editor (i.e. run `gpedit.msc` in the command prompt).
2. Expand Computer Configuration, Administrative Templates, Network, and then click SSL Configuration Settings.
3. Under SSL Configuration Settings, open the SSL Cipher Suite Order setting.
4. Set up a strong cipher suite order. See this list of Microsoft's supported ciphers (https://msdn.microsoft.com/en-us/library/windows/desktop/aa374757%28v=vs.85%29.aspx) and Mozilla's TLS configuration instructions (https://wiki.mozilla.org/Security/Server_Side_TLS).

Lighttpd

Changes should be made in `/etc/lighttpd/lighttpd.conf`

Cipher Suites

```
ssl.cipher-list = "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:E
CDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES
128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:E
CDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AE
S256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:
!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA "
```

DH parameters

```
ssl.dh-file="{path to dhparams.pem}"
```

Reload configuration

```
sudo service lighttpd restart
```

Apache Tomcat

In the `server.xml` file (for JSSE)

Cipher Suites

```
<Connector
ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_DSS_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_SHA256,TLS_ECDHE_RSA_WITH_AES_128_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_SHA,TLS_ECDHE_RSA_WITH_AES_256_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_SHA384,TLS_ECDHE_RSA_WITH_AES_256_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_SHA,TLS_DHE_RSA_WITH_AES_128_SHA256,TLS_DHE_RSA_WITH_AES_128_SHA,TLS_DHE_DSS_WITH_AES_128_SHA256,TLS_DHE_RSA_WITH_AES_256_SHA256,TLS_DHE_DSS_WITH_AES_256_SHA,TLS_DHE_RSA_WITH_AES_256_SHA"
/>
```

Note: To be able to use the 256 bit AES Ciphers, it is necessary to install the JCE Unlimited Strength Jurisdiction Policy Files, which can be found here. (http://www.oracle.com/technetwork/java/javase/downloads/index.html)

Postfix SMTP

Both parameters should be set in `/etc/postfix/main.cf`.

Cipher suites

```
smtpd_tls_exclude_ciphers = aNULL, eNULL, EXPORT, DES, RC4, MD5, PSK, aECDH, EDH-DSS-DES-CBC3-SHA, EDH-RSA-DES-CBC3-SHA, KRB5-DES, CBC3-SHA
```

DH params

```
smtpd_tls_dh1024_param_file = ${config_directory}/dhparams.pem
```

Reload configuration

```
sudo postfix reload
```

Sendmail

These changes can be made in the LOCAL_CONFIG section of your `/etc/mail/sendmail.mc`

Cipher Suites

```
O CipherList=ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA
```

DH Parameters

```
O DHParameters={path to dhparams.pem}
```

Reload configuration

```
sudo service sendmail restart
```

Dovecot

These changes should be made in `/etc/dovecot.conf`

Cipher Suites

```
ssl_cipher_list=ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDH
E-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128
-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDH
E-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA
:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES25
6-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EX
PORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA
ssl_prefer_server_ciphers = yes (Dovecot 2.2.6 or greater)
```

DH parameters

```
#regenerates every week
ssl_dh_parameters_length = 2048
```

Reload configuration

```
sudo doveadm reload
```

HAProxy

These changes should be made in the global section of your configuration.

Cipher Suites

```
ssl-default-bind-ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SH
A384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-R
SA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SH
A384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-A
ES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA
256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!
eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA
```

DH Parameters

Append the DH parameter file generated using OpenSSL to your certificate (crt file).

Note: while there is configuration option named `tune.ssl.default-dh-param` to set the maximum size of primes used for DHE, placing custom parameters in your certificate file overrides it.

Reload configuration

```
sudo haproxy -f haproxy.cfg -p $(</var/run/haproxy-private.pid) -st $(</var/run/haproxy-private.pid)
```

Amazon Elastic Load Balancing

The latest set of predefined SSL parameters (2015-05) use ECDHE ciphers, not DHE, and are therefore not vulnerable to Logjam. See the Amazon documentation. (http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-security-policy-table.html)

OpenSSH

The SSH protocol is safe from the LogJam attack in which an active attacker can force the connection down to export-grade cryptography. However, many SSH implementations, including OpenSSH use fixed primes, including the

1024-bit Oakley Group 2. There are a couple of options. The first and easiest option is to force clients to use elliptic-curve Diffie-Hellman. Specificially, Curve 25519. This can be accomplished by setting your Key Exchange algorithms as follows:

```
KexAlgorithms curve25519-sha256@libssh.org
```

If you want to continue to support non-elliptic-curve Diffie-Hellman, at the very least, you should disable Group 1 support, by removing the `diffie-hellman-group1-sha1` Key Exchange. It is fine to leave `diffie-hellman-group14-sha1`, which uses a 2048-bit prime.

It is also an option to generate new Diffie-Hellman groups:

```
ssh-keygen -G moduli-2048.candidates -b 2048
ssh-keygen -T moduli-2048 -f moduli-2048.candidates
```

You then need to install `moduli-2048` to your system's moduli file. In Debian/Ubuntu, this is located at `/etc/ssh/moduli`. SSH chooses (practically randomly) from this file, so you should replace your existing moduli file with the new groups you generated instead of appending these new groups.