M³AAWG Messaging Malware Mobile Anti-Abuse Working Group

Messaging, Malware and Mobile Anti-Abuse Working Group

M³AAWG Initial Recommendations for Using Forward Secrecy to Secure Data

January 2016

Introduction

Deploying opportunistic encryption as described in "TLS for Mail: M³AAWG Initial Recommendations" is an excellent way to start protecting email traffic between messaging providers. However, implementers of TLS may not realize that it is not sufficiently secure unless forward secrecy is also employed for the connection. If an adversary captures and retains encrypted traffic, and is then able to acquire the private keys used to encrypt it, all retained traffic can be read as plaintext.

Forward secrecy is a set of cryptographic protocols that address this vulnerability. Enabling forward secrecy in conjunction with TLS assists in protecting captured traffic against any possibility of eventual decryption. This document is intended to demonstrate how forward secrecy[1] and ephemeral keys[2] can protect data transmitted during these sessions, and more importantly, how they can protect against future decryption if an untrustworthy party who has access to the data also acquires the keys.

Rationale for Using Forward Secrecy to Secure Data

The bulk content encryption/decryption that takes place during an encrypted SSL/TLS session makes use of a highly efficient *symmetric* cipher, such as AES. Symmetric ciphers get their name because they use the same key to both encrypt plaintext and decrypt enciphered text. Therefore, the parties use *public key cryptography*, which is more secure, to share the symmetric cipher.

Public key cryptography uses a *public/private key pair* and is asymmetrical, meaning there are separate keys to encrypt and decrypt the data. The public key is intended to be made widely available, but messaging providers are exceedingly careful to keep the private key secret because anyone in possession of it can decrypt anything encrypted with the corresponding, freely-shared public key. In SSL/TLS, the public/private key pair normally used is the RSA key pair generated at the time a certificate-signing request is created. Once established, these public/private key pairs rarely, if ever, change.

The approach outlined above generally works well with the exception of one scenario: What if an adversary intercepts and retains some or all of a messaging provider's SSL/TLS-encrypted traffic and *also* manages to obtain a copy of the provider's private key? If the provider has not been using a cipher suite that provides forward secrecy to safeguard the private key, the adversary will have everything needed to retrospectively decrypt *all* the previously retained traffic associated with that key.

Can an adversary really capture traffic? Yes. Some adversaries may be *routinely* capturing (and archiving) all traffic flowing across any links they can access. In other cases, traffic can be redirected by adversaries specifically so they can make a copy of it.

---

[1] **Forward secrecy,** https://en.wikipedia.org/wiki/Transport_Layer_Security#Forward_secrecy
[2] **Ephemeral key,** https://en.wikipedia.org/wiki/Ephemeral_key

How might an adversary obtain the private key? There are two primary ways the private key of a system targeted for collection might be acquired:

1. Private keys might be accidentally exposed as a result of software flaws. For example, private keys were potentially exposed to adversaries by the HeartBleed vulnerability. That is why all affected sites needed to create new public/private key pairs and obtain new certificates.

2. Alternatively, since many sites just store their private key in a conventional file rather than using a hardware security module (HSM), anyone who can arrange access to the file, and the keys contained in it, could decrypt any associated encrypted traffic. Strategies for getting access might include:
   - Suborning of a system administrator or other privileged user (bribery, extortion, physical coercion, etc.)
   - Accessing a poorly-secured copy of that file, perhaps by accessing an unencrypted backup stored at a third-party site or hacking/cracking that specifically targets the critical file's contents
   - Accidental or intentional disclosure of a current or previous private key by an unwitting administrator
   - Obtaining a court order compelling disclosure

Using Forward Secrecy to Secure Data

Fortunately, ephemeral key exchange offers a solution to this problem. When a site uses a key exchange mechanism that provides forward secrecy, such as Diffie-Hellman Ephemeral (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE/EECDH), a new public/private key pair is created for each connection and then discarded immediately after use. With this approach, even if traffic has been captured and the security of the RSA private key has been compromised, these adverse events will not allow an adversary to retrospectively decrypt any messages previously obtained.

In using ephemeral key exchange mechanisms, some care must be taken to ensure that Diffie-Hellman parameters are long and therefore strong enough. At least in some circumstances, the default Diffie-Hellman parameters may only be 1024 bits long. Fortunately, current versions of popular cryptographic libraries such as OpenSSL now allow the use of even 4096-bit DH parameters.[3]

Even though RSA public/private key crypto is not used as the basis for securely sharing a symmetric key during session setup, RSA public/private key crypto (or at some point in the future, elliptic curve crypto) still plays an important role in SSL/TLS crypto. It forms the basis for authenticating the remote system and ensures that an imposter site is not attempting to conduct an active Man-in-the-Middle attack. While RSA 2048-bit is the commonly seen industry norm, most certificate authorities can just as readily accommodate customer CSRs with RSA 3072- or 4096-bit keys. (Note: Before moving to RSA 3072 or 4096, carefully evaluate the impact that using these longer keys might have on system throughput and capacity.)

Guidance for Implementing Forward Secrecy to Secure Data

When configuring a Web server or other application that uses SSL/TLS, be sure to select "ephemeral" cipher suites. The ephemeral cipher suites require suitable Diffie-Hellman parameters. The exact process for selecting these suites will vary from cryptographic library to cryptographic library, so check the library's documentation.

[3]Alex Halderman and Nadia Heninger "How is NSA breaking so much crypto?," Freedom to Tinker, October 14, 2015, https://freedom-to-tinker.com/blog/haldermanheninger/how-is-nsa-breaking-so-much-crypto/

Several examples are provided here to illustrate what is typically required.

1. *Implement Forward Secrecy*

   a) Example: OpenSSL 1.0.1p with nginx[4] 1.9.5.

      i. Begin by creating suitable Diffie-Hellman parameters (the default may be too small):

         - `cd /etc/ssl/certs`
         - `openssl dhparam`[5] `-out dhparam.pem 4096`

         Note: It can take a long time to create the dhparam.pem file on many systems.

      ii. Now, in the nginx.conf configuration file, add a suitable cipher suite specification:

         - `ssl_ciphers 'AES256+EECDH:AES256+EDH';`
         - `ssl_dhparam /etc/ssl/certs/dhparam.pem;`

   b) Example: OpenSSL 1.0.1p with Dovecot 2.2.

      i. Create an SSL cert as done here: http://wiki2.dovecot.org/SSL/CertificateCreation

      ii. Typically, in the conf.d/10-ssl.conf:

         - `ssl_protocols = !SSLv2 !SSLv3`
         - `ssl_prefer_server_ciphers = yes`
         - `ssl_cipher_list =`
           `EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH+aRSA+RC4:EECDH:EDH+aRSA:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4`

   c) Example: OpenSSL 1.0.1p with Postfix 2.6

      i. Generate certs such as:

         - `openssl dhparam -out dh_4096.pem 4096`

      ii. In the main.cf

         - `smtpd_tls_eecdh_grade = strong # 128bit, "ultra" for 196bit`
         - `smtpd_tls_4096_param_file = /etc/postfix/dh_4096.pem`
         - `smtpd_tls_mandatory_exclude_ciphers = aNULL, MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL`
         - `smtpd_tls_protocols= !SSLv2,!SSLv3`
         - `smtpd_tls_mandatory_protocols= !SSLv2,!SSLv3`
         - `tls_preempt_cipherlist = yes`
         - `smtpd_tls_loglevel = 1`
         - `smtp_tls_loglevel = 1`

Note: Cryptographic cipher suite libraries and strong cipher suite specifications tend to evolve over time. While the above examples are believed to constitute a strong specification at the time of publication, configuration requirements are likely to change and should be checked.

---

[4] "Strong SSL Security on nginx," https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html

[5] Open SSL Cryptography and SSL/TLS Toolkit, "dhparam," https://www.openssl.org/docs/apps/dhparam.html

2. ***Verify Configuration***

Run an SSL tester that has the ability to validate FS functionality. This can be done via a website or a shell script.

d) Some example website testers can be found at https://www.ssllabs.com/ssltest/ or https://ssl-tools.net/

e) Check the handshake simulation on one of the preceding sites to confirm the cipher suite that should be used.

Conclusion

Unlike other key exchange mechanisms, Diffie-Hellman Ephemeral key exchanges create an environment that makes decryption of captured data with retrospectively obtained private keys effectively impossible. The Messaging, Malware and Mobile Anti-Abuse Working Group recommends that industry messaging providers use forward secrecy in conjunction with TLS to help keep data private.[6] Widespread deployment of this practice will benefit everyone involved.

These guidelines are focused on one aspect of cryptographic messaging security. Please review the other documents currently available from M³AAWG that offer advice on other cryptographic topics and keep an eye out for new documents that will be forthcoming in the future.

References

Open SSL Cryptography and SSL/TLS Toolkit, "dhparam," https://www.openssl.org/docs/apps/dhparam.html

Forward secrecy section of "Transport Layer Security," https://en.wikipedia.org/wiki/Transport_Layer_Security#Forward_secrecy

Alex Halderman and Nadia Heninger "How is NSA breaking so much crypto?," Freedom to Tinker, October 14, 2015, https://freedom-to-tinker.com/blog/haldermanheninger/how-is-nsa-breaking-so-much-crypto/

Ivan Ristić, "Increasing DHE strength on Apache 2.4.x," Blog: Ivan Ristić, August 15, 2013, http://blog.ivanristic.com/2013/08/increasing-dhe-strength-on-apache.html

"Strong SSL Security on nginx," https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html

"Why the Web Needs Perfect Forward Secrecy More Than Ever," https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy

As with all best practices that we publish, please check the M³AAWG website (www.m3aawg.org) for updates to this document.

[6] Yan Zhu and Yan Zhu, "Why the Web Needs Perfect Forward Secrecy More Than Ever," Electronic Frontier Foundation, April 8, 2014, https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy