TLS “Secrets”

Whitepaper presenting the security implications of the deployment of session tickets(RFC 5077) as implemented in OpenSSL

Matta Consulting Limited

+44 (0) 20 3051 3420

https://www.trustmatta.com

Summary

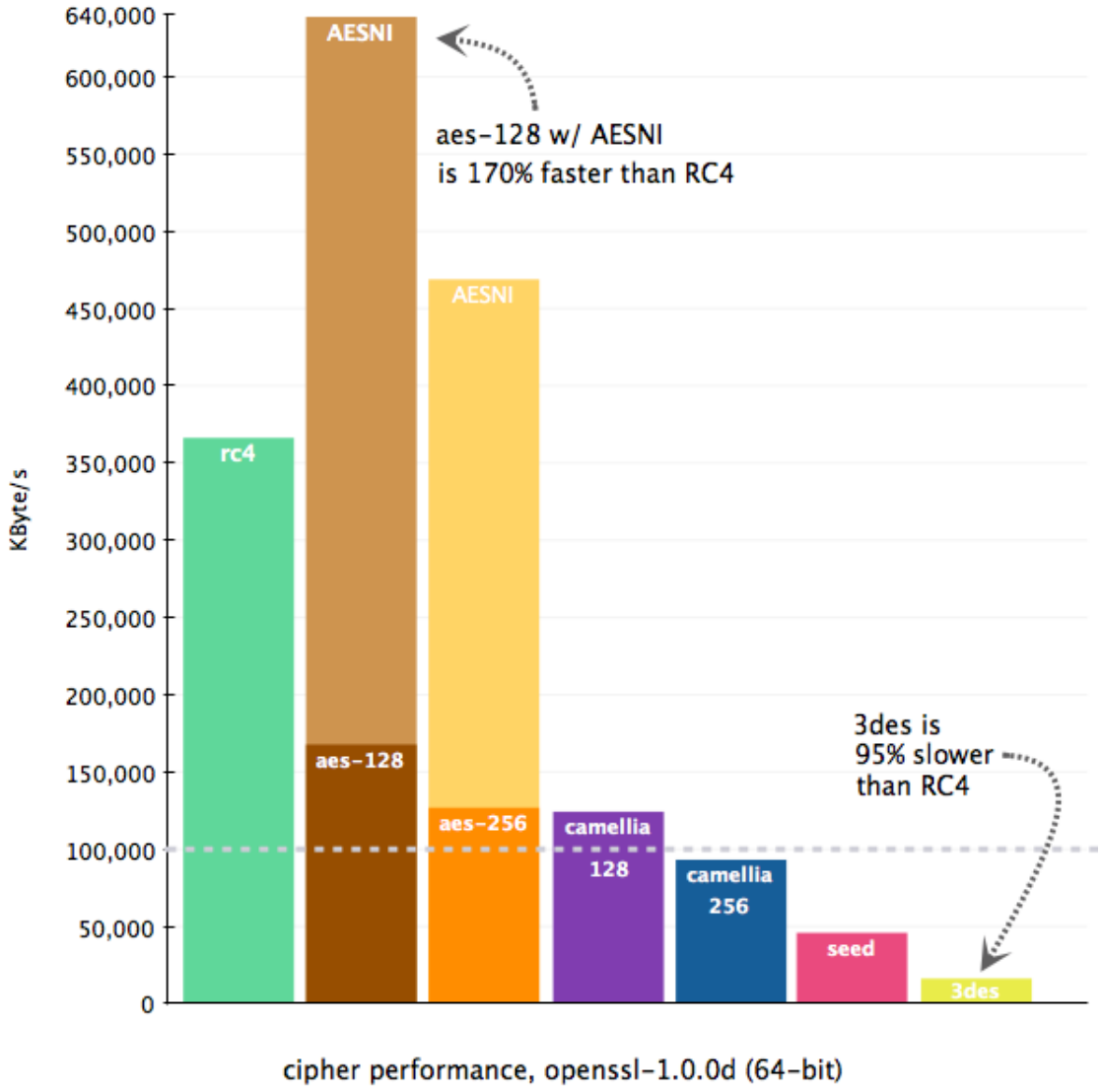
Overview

SSL and TLS have become the de-facto standards for transport-layer encryption. In recent years, many vulnerabilities have been uncovered in both the standards, their implementation and the way people configure and use them. The talk associated to this whitepaper is exploring in details a lesser-known and much less talked about part of the standard which breaks some of the security properties one would expect.

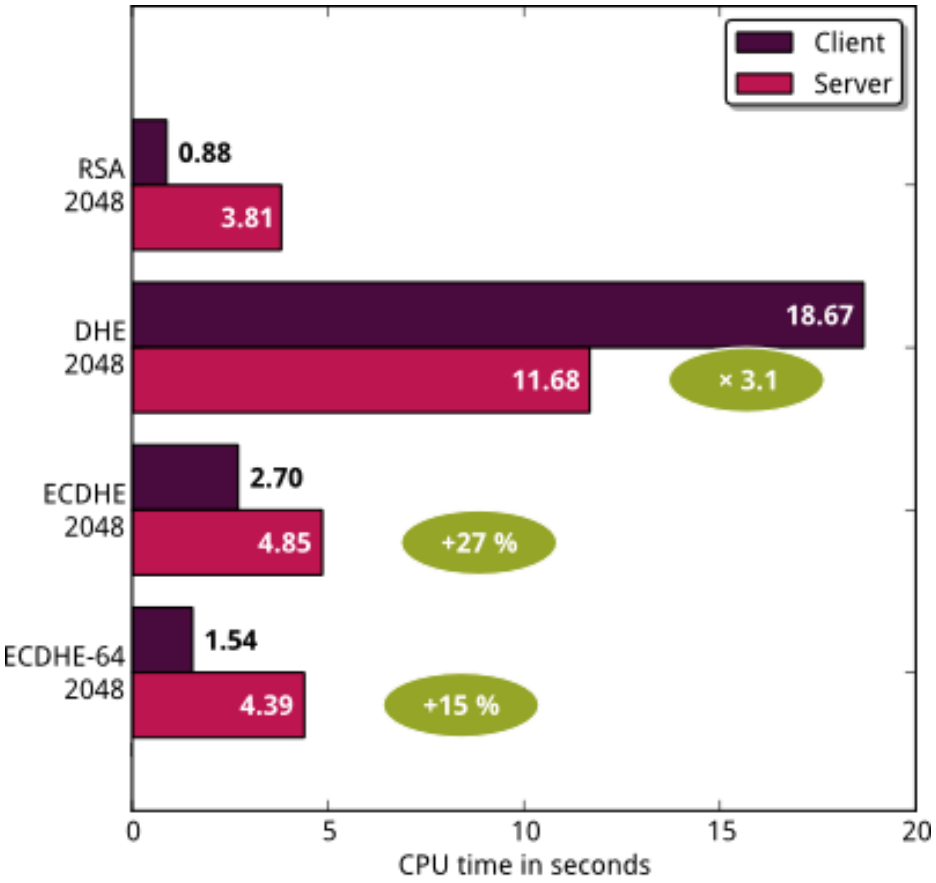
Performance of SSL/TLS

SSL sessions are costly to establish but relatively cheap to maintain. Cipher choice and key size are the main cost factor CPU-cycle wise. This is illustrated in by the following diagrams:

Impact of cipher choice on performance:

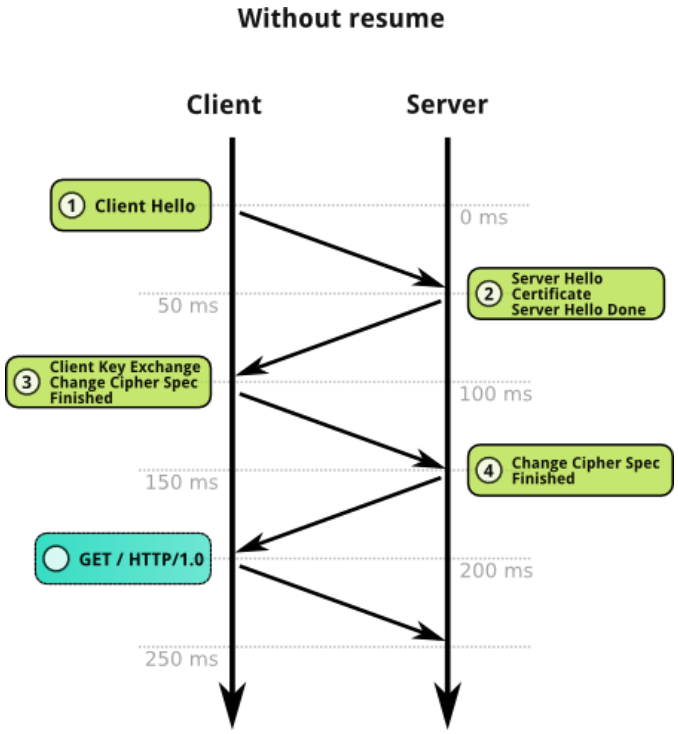


Impact of Key size on performance:

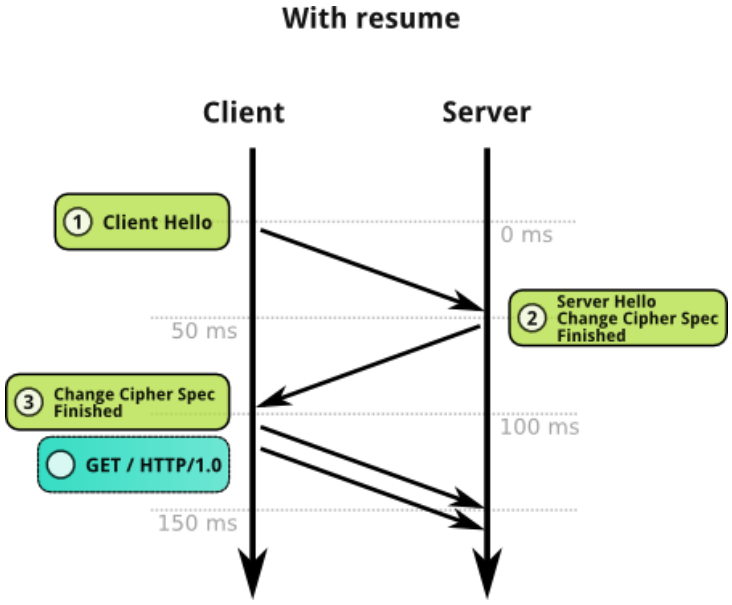


To optimize the performance of an SSL setup, one is usually well advised to look into minimizing the number of handshakes (involving asymmetric cryptographic operations). This is done by using abbreviated handshakes through session resumption, as illustrated below:

Full SSL Handshake:



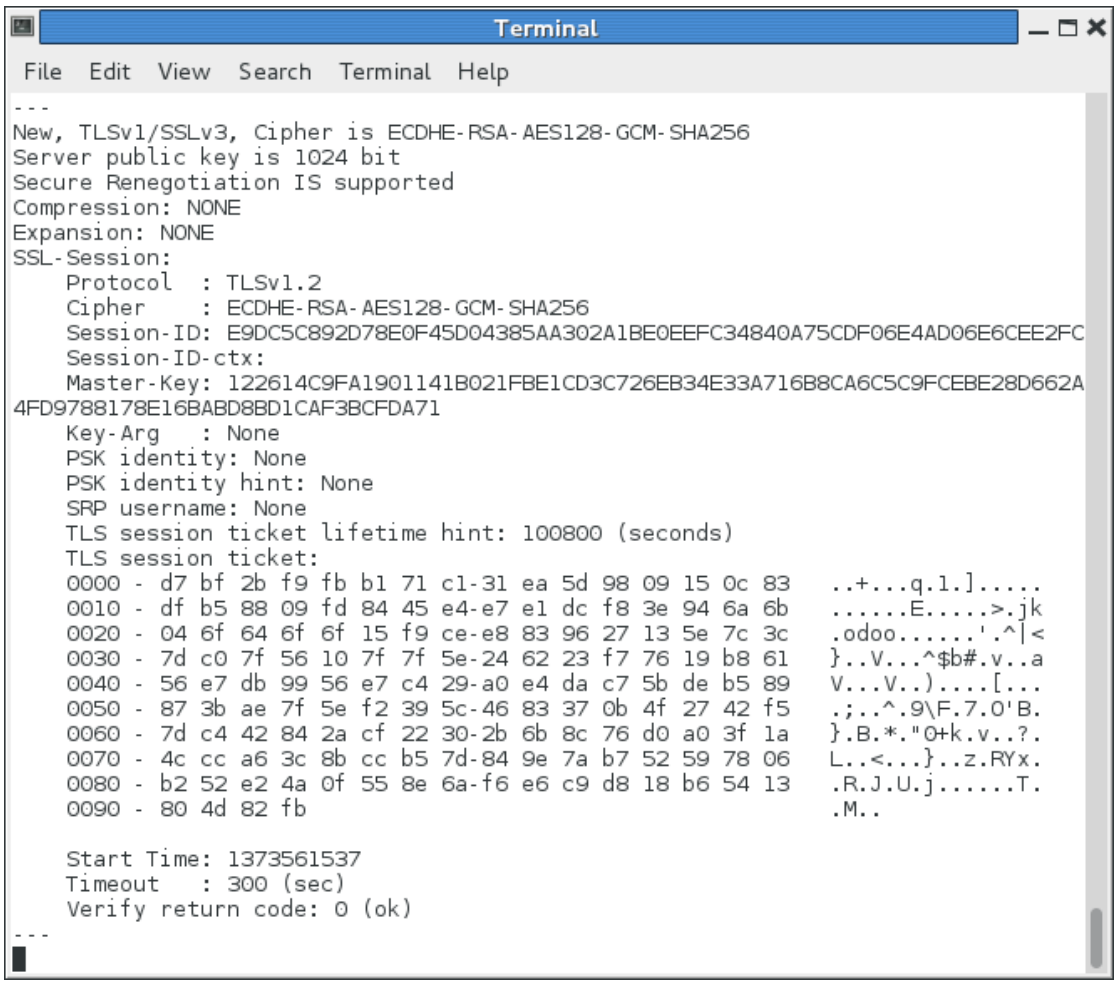
Abbreviated SSL Handshake:



Introduction to Session resumption

There are two main mechanisms to perform session resumption: Using session-IDs and using session-tickets (RFC5077). The latter mechanism is only available since TLS1.0 whereas session-IDs exist since SSL.

The following illustration illustrates what both a Session ID and a TLS ticket are:



Security implications of using RFC 5077 with OpenSSL

RFC 5077 is augmenting the Transport Security Layer by allowing for speedy handshake resumption without requiring servers to keep shared state (shared keying material). While this is simplifying the server-side infrastructure, it demands that the client retains the encrypted key material across sessions and present it to the server upon connection.

The RFC describes how one can format the data in the ticket. This has been followed to the letter in the OpenSSL implementation.

4. Recommended Ticket Construction

This section describes a recommended format and protection for the ticket. Note that the ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. If implementations do diverge from this format, they must take security concerns seriously. Clients MUST NOT examine the ticket under the assumption that it complies with this document.

The server uses two different keys: one 128-bit key for Advanced Encryption Standard (AES) [AES] in Cipher Block Chaining (CBC) mode [CBC] encryption and one 256-bit key for HMAC-SHA-256 [RFC4634].

The ticket is structured as follows:

```
struct {
    opaque key_name[16];
    opaque iv[16];
    opaque encrypted_state<0..2^16-1>;
    opaque mac[32];
} ticket;
```

Here, key_name serves to identify a particular set of keys used to protect the ticket. It enables the server to easily recognize tickets it has issued. The key_name should be randomly generated to avoid collisions between servers. One possibility is to generate new random keys and key_name every time the server is started.

The actual state information in encrypted_state is encrypted using 128-bit AES in CBC mode with the given IV. The Message Authentication Code (MAC) is calculated using HMAC-SHA-256 over key_name (16 octets) and IV (16 octets), followed by the length of the encrypted_state field (2 octets) and its contents (variable length).

The RFC doesn't describe how key management should be performed and merely emits recommendations. This has been found to be problematic in OpenSSL's case: **Keys are never rotated and always 128bits, regardless of the cipher negociated.**

5.5. Ticket Protection Key Management

A full description of the management of the keys used to protect the ticket is beyond the scope of this document. A list of RECOMMENDED practices is given below.

- The keys should be generated securely following the randomness recommendations in [RFC4086].
- The keys and cryptographic protection algorithms should be at least 128 bits in strength. Some ciphersuites and applications may require cryptographic protection greater than 128 bits in strength.
- The keys should not be used for any purpose other than generating and verifying tickets.
- The keys should be changed regularly.
- The keys should be changed if the ticket format or cryptographic protection algorithms change.

5.6. Ticket Lifetime

The TLS server controls the lifetime of the ticket. Servers determine the acceptable lifetime based on the operational and security requirements of the environments in which they are deployed. The ticket lifetime may be longer than the 24-hour lifetime recommended in [RFC4346]. TLS clients may be given a hint of the lifetime of the ticket. Since the lifetime of a ticket may be unspecified, a client has its own local policy that determines when it discards tickets.

Conclusion

The security implications associated with using the default settings of OpenSSL can be summarized as follows:

- 128 bit of security is all you get (at best), regardless of the cipher which has been negociated
- The Perfect Forward Secrecy interval is likely to be more than expected – the program's lifetime in most cases (as opposed to hours like best practices would recommend)

Matta is planning on releasing a tool allowing for forensic recovery of the key allowing to decrypt session tickets, which in turn, once decrypted will reveal the Master Key allowing to derive the session-keys which can be used to decrypt the cyphertext to recover the plaintext.

References (illustrations)

http://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077.html

http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html

http://zombe.es/post/4078724716/openssl-cipher-selection

https://tools.ietf.org/rfc/rfc5077.txt