Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 28, 2016

D. Migault
K. Ma
Ericsson
R. Rich
Akamai
S. Mishra
Verizon Communications
O. Gonzales de Dios
Telefonica

May 27, 2016

LURK TLS/DTLS Use Cases
draft-mglt-lurk-tls-use-cases-01

Abstract

TLS as been designed to setup and authenticate transport layer between a TLS Client and a TLS Server. In most cases, the TLS Server both terminates the TLS Connection and owns the authentication credentials necessary to authenticate the TLS Connection.

This document provides use cases where these two functions are split into different entities, i.e. the TLS Connection is terminated on an Edge Server, while authentication credentials are generated by a Key Server, that owns the Private Key.

Table of Contents

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

TLS has been designed for end-to-end security between a TLS Server and a TLS Client. As TLS is widely used to provide an authenticated channel between applications, the following models assumes that applications end points and connectivity end point are combined. In that case, authentication of the connection end point and authentication of the application end point could be combined and assimilated as a single authentication.

Such assumption for the TLS model may not be true especially in the current web architecture where application content is not anymore associated with the connection end point. For example, Content Delivery Network are in charge of delivering content they are not necessarily owning.

This document provides use case where authentication of of the TLS Server involves multiple parties or entities as opposed to a single entity in the standard TLS model.

3. Terminology

TLS Client: The TLS Client designates the initiator of the TLS session. The terminology is the one of [RFC5246]. The current document considers that the TLS Client and the application initiating the session are hosted on the same host. If not they are hosted on the same administrative domain with a trust relation between the TLS Client and the application. In other words, the client endpoint is considered to be a single entity as described initially in [RFC5246].

TLS Server: The TLS Server designates the endpoint of a TLS session initiated by the TLS Client. In the standard TLS, the TLS Server, is both the terminating point of the TLS Connection and the entity authenticated by the TLS Client. This document considers that the TLS Server be split into the Edge Server terminating the TLS Connection and the Key Server providing the necessary capabilities so the TLS Client proceed to the authentication.

Private Key : is the cryptographic credential used to authenticate the TLS Server by the TLS Client. The purpose of the document is to enable the Private Key to be hosted in the Key Server outside the TLS Connection terminating point.

TLS Connection : The authenticated TLS Connection between the TSL Client and the TLS Server. In this document, the TLS

Connection terminates on the Edge Server and authenticates the Private Key hosted on the Key Server.

Key Server: The server hosting the Private Key. The Key Server provides an interface that enable cryptographic operations to be performed remotely by the Edge Servers.

Edge Server: The Edge Server designates a node that handles traffic for a Content Provider. A TLS Client initiates a TLS session to authenticate a Content provider, but may be in fact served by a Edge Server that may belong to a different administrative domain.

Content Owner: The owner of the content. This is the entity requested by the application of the TLS Client.

Content Provider: The entity responsible to deliver the content. In some cases, the Content Provider is the managing the Content Delivery Network.

Content Delivery Network (CDN): designates a organization in charge of managing delivery of a content on behalf of a Content Provider. In most cases, the CDN is a different organization than the Content Provider.

4. Architecture Overview

Figure 1 provides an overview of the architecture considered by the different uses cases exposed in this document.

The TLS Client initiates a TLS connection with the Edge Server (1) which is expected to be authenticated by its Private Key. The Edge Server terminates the TLS connection of the TLS Client, but does not own the Private Key. Instead, the Private Key is owned by the Key Server, which performed all cryptographic operations on behalf of the Edge Server. Upon request from the Edge Server, the Key Server provides the authentication credentials to the Edge Server (2). These authentication credentials depends on the authentication methods agreed between the TLS Client and the Edge Server as well as the capabilities of the Key Server. The Authentication Credentials returned by the Key Server enables the Edge Server to complete the TLS Handshake and the TLS Client to authenticate the Edge Server as a key owner of the Private Key (3).

Such architecture consists in splitting the standard TLS Server into two functions: the Edge Server that terminates the TLS Connection and the Key Server that is hosting the Private Key and performs all necessary cryptographic operations for the authentication.

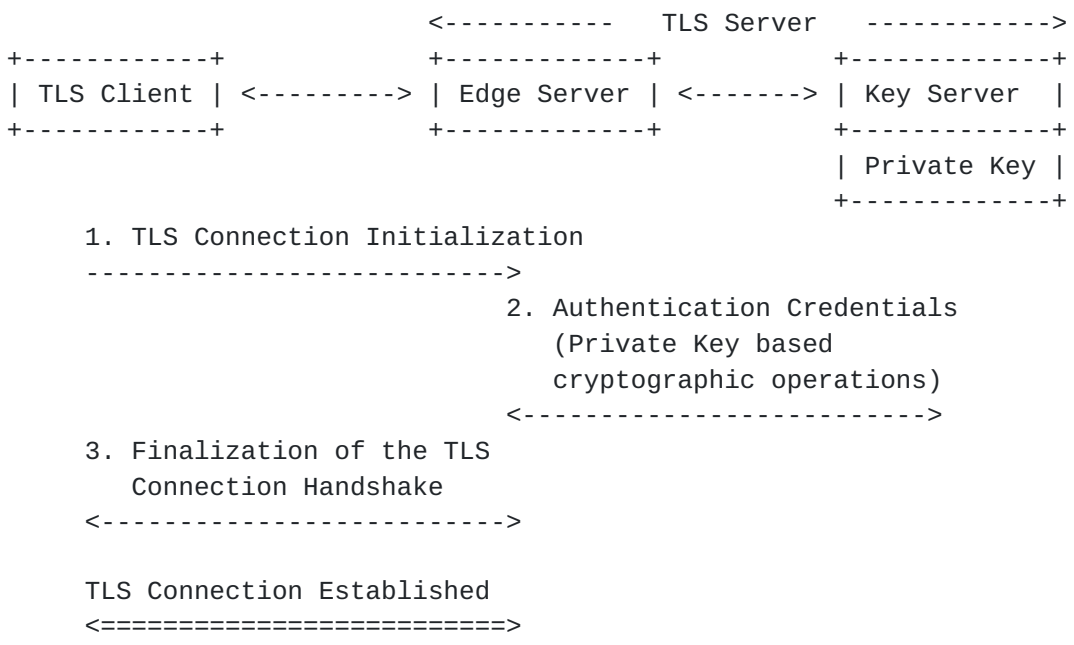

Figure 1: TLS Session Key Interface Architecture

5. Use cases

5.1. Containers and Virtual Machines Use Cases

In virtual environment application servers may run within virtual machines or containers. When TLS is used to provide an authenticated and encrypted communication channel to the application, it is currently common that the container or the virtual machine hosts the Private Key used for the authentication. Hosting multiple copies of the Private Key through the cloud increases the risk of leaking the Private Key.

For example, virtualization and persistent storage of virtual machines or containers image over different places in the cloud may result in multiple copies of the Private Key through the cloud. In addition, operating system level virtualization is a virtualization method with a very low overhead. On the other hand, isolation is performed at the process and kernel level, which may provide a smaller isolation compared to the one provided by the traditional hypervisors. With lighter isolation, containers avoid storing private and highly sensitive data such as a Private Key.

As a result, in order to prevent the leak of the Private Key used for the TLS authentication, cloud provider may prefer storing the Private Key in a centralized Key Server that is remotely access by the different instances of the virtual machines or containers.

In this scenario, the Key Server as well as the containers or virtual machine are expected to be hosted on the same Cloud. As a result, the latency between the Key Server and the containers or the virtual machine and the Key Server remains limited and acceptable for the TLS Client. In addition, the containers or virtual machines communicating with the Key Server may be different applications running on different operating systems.

5.2. Content Provider Use Case

A Content Provider may use multiple Edge Servers, that are directly exposed on the Internet. Edge Servers presents a high risk of exposure as they are subject for example to misconfiguratons as well as all vulnerabilities running on the Edge Server. This includes, os vulnerabilities to web applications vulnerabilities. as illustrated for example by the Heartbleed attack [HEART]. More specifically, the Heartbleed attack uses a weakness of a software implementation to retrieve the private key used by the TLS server. Such attack would not for example has been so successful if the private key was not stored on the Edge Server. In addition, a Cloud Provider may run different implementations of web servers, or OS in order to make its infrastructure or service less vulnerable to a single vulnerability. On the other hand, the diversity of implementations increases the risk of a leakage of the Private Key.

Note that if the Private Key is shared between multiple Edge Servers, a leakage occurring at one Edge Server affect the service.

A Content Provider, may prefer to store the critical information in a more contained place the Key Server, accessed only by all the authenticated Edge Servers.

In this scenario, the Key Server is accessed by a limited number of Edge Servers which are authenticated. An Edge Server may present a vulnerability, it will not have access to the Private Key. It eventually may use the identity of the Edge Server to perform cryptographic operations with that Private Key, and means should be provided to limit the usability of such use.

In this scenario, the latency between the Edge Server and the Key Server depends on the distribution of the Edge Servers. When Edge Servers are far away from the Key Server, the time to set TLS Connection may be impacted by this architecture. In case, such overhead impact the quality of service of the TLS Client, the Content Provider may use multiple Key Servers in order to reduce the latency of the communication between the Edge Server and the Key Server. This scenario assumes that we are within a single administrative domain, so the Private Key remains inside the boundaries of such

domain. In addition, the use of the Key Server prevents a direct access to the Private Key.

5.3. Content Owner / Content Provider Use Case

It is common that applications - like a web browser for example - use TLS to authenticate a Content Owner designated by a web URL and build a secure channel with that Content Owner.

When the Content Owner is not able to support all the TLS Client requests or would like to optimize the delivery of its content, it may decide to take advantage of a third party delivery service designated a Content Delivery Network (CDN) also designated as the Content Provider. This third party is able to locate the Edge Servers closer to the TLS Clients in various different geographical locations.

The Content Owner may still want to be authenticated by TLS Client while not terminating the TLS Connection of the TLS Client. In addition, while the Content Owner is provides the Content Provider the content to deliver it may not accept to provide its Private Key to the Content Provider. In fact, the Private Key used to authenticate the Content Provider may present more value than the content itself. For example, the content may be accessed by devices or clients configured with the public authentication credential. In such cases, the leak of the Private Key and the renewal of the Private Key would require to configure all these devices. Such additional configuration are likely to affect the image of the Content Provider as well as result in some interruption of the service. The content, on the other hand may have only an ephemeral value and the Content Owner, may accept the risks of leakage and provide the Content Provider the content in cleartext. Alternatively, the content may also be encrypted with DRM, so its access remains restricted to authorized users only.

In this scenario, the Content Provider and the Content Owner are different administrative entities. As a result, the Key Server and the Edge Servers may be located in different networks and the communication between the Edge Server and the Key Server may introduce some delays. Such delay may be acceptable for the TLS Client, especially for long term TLS connections. Otherwise, the Content Owner, may provide the Key Server to the Content Provider. This use case is then very similar to the one described in Section 5.2. Note that providing the Key Server to the Content Provider in a hardware security module for example, still prevent the Content Provider to access the Private Key while providing its usage.

In this scenario, the Content Owner is likely to involve multiple Content Providers. In addition, the agreement between the Content Provider and the Content Owner may take various forms. The Content provider may provide for example, an infrastructure, or a delivery service. As a result, the Content Owner may not control the application or TLS library interacting with the Key Server.

5.4. Content Distribution Network Interconnection Use Case

In the case of Content Distribution Network Interconnection (CDNI) [RFC6707], it may also that the company with which the Content Owner has contracted may further delegate delivery to another CDN with which the Content Owner has no official business relationship. Even if the Content Provider trusts the upstream CDN, and perhaps has strong legal contracts in place, it has no control over, and possibly no legal recourse against, the further downstream CDNs.

In this case, similarly to Section 5.3 the delegating CDN may provide the content but not the Private Key. If the delegating CDN hosts the Key Server it needs to to provide an access to the Key Server. On the other hand, the delegating CDN may not even host the Key Server, in which case, it may proxy the communications to the upstream CDN or the Content Owner. Other architecture may also enable a direct access to the Key Server by the delegated CDN.

In this scenario, different CDN are interacting, and the access to teh Key Server may result in substantial additional latencies. This additional latency should not affect the quality of service of the delivery service. In addition, the motivations for providing content to the delegated CDN without providing the Private Key are similar to those of Section 5.3.

6. Requirements

The requirements listed in this section are limited to the LURK protocol, that is the exchanges between the Edge Server and the Key Server.

6.1. LURK Requirements

This section provides the requirements associated to the protocol used by the Edge Server and the Key Server. In the remaining section, this protocol is designated as LURK.

Multiple implementations of Edge Server, located in different administrative domains must be able to interact with multiple implementations of Key Servers also located in multiple

administrative domains. In addition, the scope of LURK is closely related to TLS standardized at the IETF.

R1: LURK MUST be standardized at the IETF

LURK is limited to the Edge Server and the Key Server, so it is expected to be transparent to the TLS Client. In addition, in order to be deployed in the short term, any modification on the TLS Client should be avoided.

R2: LURK MUST NOT impact the TLS Client.

LURK is associated to TLS related operations performed by the Key Server on behalf of the Edge Server. On the other hand, interactions between the Edge Server and the Key Server also consists enable control-plan like operations such as reachability, capabilities discovery.

R3: LURK MUST provide control plane-like facilities such as reachability, keep-alive, and capability discovery.

6.2. Key Server Requirements

The Key Server holds the Private Key, and interacts with the Edge Servers.

R4: The Key Server MUST be able to provide the necessary authentication credential so the TLS Client and the Edge Server set an authenticate TLS Connection with the Private Key.

R5: The Key Server MUST NOT leak any information associated to the Private Key. In particular the Key Server MUST NOT provide a generic singing/encryption oracle.

R6: The Key Server SHOULD NOT perform any operation outside the authentication of a TLS Connection.

R7: The Key Server MUST provide confidential information to the Edge Sever over an authenticated and encrypted channel.

6.3. Edge Server Requirements

R8: The Edge Server SHOULD be provisioned with the public authentication credentials, and so public certificate provisioning is outside of LURK.

7. Security Considerations

8. IANA Considerations

There are no IANA considerations in this document.

9. Acknowledgements

Thanks are due for insightful feedback on this document to Robert Skog, Hans Spaak, Salvatore Loreto, John Mattsson, Alexei Tumarkin, Yaron Sheffer, Eric Burger, Stephen Farrell, Richard Brunner, Stephane Dault, Dan Kahn Gillmor, Joe Hildebrand, Thomas Fossati, Kelsey Cairns.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

10.2. Informative References

[HEART] Codenomicon, "The Heartbleed Bug", <http://heartbleed.com/>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <http://www.rfc-editor.org/info/rfc6707>.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Kevin Ma J
Ericsson
43 Nagog Park
Acton, MA 01720
USA

Phone: +1 978-844-5100
Email: kevin.j.ma@ericsson.com

Rich Salz
Akamai

Email: rsalz@akamai.com

Sanjay Mishra
Verizon Communications

Email: sanjay.mishra@verizon.com

Oscar Gonzales de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com