Digital Investigation 16 (2016) S114–S123

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2016 Europe — Proceedings of the Third Annual DFRWS Europe

TLSkex: Harnessing virtual machine introspection for decrypting TLS communication

Benjamin Taubmann[*], Christoph Frädrich, Dominik Dusold, Hans P. Reiser

University of Passau, Innstr. 43, 94032 Passau, Germany

Keywords:
Virtual machine introspection
Transport layer security
Decryption
Malware analysis
Virtualization
Semantic gap

ABSTRACT

Nowadays, many applications by default use encryption of network traffic to achieve a higher level of privacy and confidentiality. One of the most frequently applied cryptographic protocols is Transport Layer Security (TLS). However, also adversaries make use of TLS encryption in order to hide attacks or command & control communication. For detecting and analyzing such threats, making the contents of encrypted communication available to security tools becomes essential. The ideal solution for this problem should offer efficient and stealthy decryption without having a negative impact on over-all security. This paper presents TLSkex (TLS Key EXtractor), an approach to extract the master key of a TLS connection at runtime from the virtual machine's main memory using virtual machine introspection techniques. Afterwards, the master key is used to decrypt the TLS session. In contrast to other solutions, TLSkex neither manipulates the network connection nor the communicating application. Thus, our approach is applicable for malware analysis and intrusion detection in scenarios where applications cannot be modified. Moreover, TLSkex is also able to decrypt TLS sessions that use perfect forward secrecy key exchange algorithms. In this paper, we define a generic approach for TLS key extraction based on virtual machine introspection, present our TLSkex prototype implementation of this approach, and evaluate the prototype.

Introduction

The proliferation of encrypted communication in the Internet has led to an increase in security, but also to an increase in the difficulty of performing forensic investigations and analysis of malicious activity. Today's defacto standard for securing communication is transport layer security (TLS), which is used in a large variety of applications, including e-mail, instant messaging and VoIP communication. A recent NSS Labs report concluded that encryption using TLS "actually reduces security on the corporate network by creating blind spots for corporate security infrastructures" (Pric, 2013). It is likely that an attack against a web-based server uses TLS-based communication channels. Similarly, malware can use TLS channels for protecting information leakage or for communicating with a command & control server. A detailed investigation of such problems can benefit from the ability to decrypt the TLS-encrypted network traffic.

Among existing approaches for TLS decryption, which we discuss in more detail in the Section Related work, active TLS proxies are most likely the most practical approach. Such a proxy acts as a "man-in-the-middle", decrypting and re-encrypting the network traffic. In this paper, we investigate whether TLS-encrypted network traffic can be decrypted if using such a proxy is not feasible. Ideally, TLS decryption should work in a *non-intrusive* and *universal* way. Being non-intrusive implies the following two requirements:

[*] Corresponding author.
E-mail address: bt@sec.uni-passau.de (B. Taubmann).

http://dx.doi.org/10.1016/j.diin.2016.01.014

- *No active manipulation of communication*: The communication should be monitored passively without modifying the contents of the communication (we do not exclude a possible impact on the timing of messages).
- *No modification of application*: The decryption should work without internal modifications to the communicating applications (such as exporting the session key to a file).

Being universal implies the following four requirements:

- *Independence of specific key exchange*: The decryption key extraction should work for any key exchange algorithm.
- *Independence of encryption algorithm*: The decryption and key extraction should work independently of a specific cryptographic algorithm.
- *Independence of client/server role*: It should work for local applications that operate as a server, as well as for local client applications that connect to a remote server.
- *Independence of the implemenation*: The key extraction should work for every TLS implementation.

This paper proposes a novel approach for TLS decryption based on virtual machine introspection (VMI) and presents details of TLSkex, a prototype implementation of this approach. The proposed approach makes the following contributions. First, it proposes a method for inspecting network traffic to detect TLS connections and extracting essential information out of it. Second, it defines a strategy for minimizing the size of memory snapshots that potentially contain TLS session keys. Third, it proposes and compares various heuristics to minimize the time required for a brute-force search of cryptographic session keys in memory snapshots. Finally, it presents an evaluation of the prototype implementation and discusses benefits and limitations.

The remainder of this paper is organized as follows. In Section TLS internals we present some background about the TLS protocol. The basic concepts of TLSkex are discussed in Section Conceptual approach. The implementation details are presented in Section TLSkex implementation and evaluated in Section Evaluation. In Section Related work we compare our approach to related work in the fields of TLS decryption, cryptographic key extraction of main memory and VMI. We finally conclude our elaborations in Section Conclusion.

TLS internals

TLS is the successor of secure sockets layer (SSL). These cryptographic protocols provide a secure communication channel.

TLS uses cryptographic certificates based on asymmetric cryptography for server authentication and – optionally – client authentication. This means that all decryption attempts based on active man-in-the-middle intercepts with fake certificates can be detected, unless the interceptor has access to the private keys of the original endpoints.

After endpoint authentication, TLS negotiates a symmetric session key (master secret). This negotiation can be done with RSA encryption, or with the Diffie-Hellman (DH) or elliptic curve Diffie-Hellman (ECDH) algorithm. Nowadays, DH or ECDH should be the preferred way to negotiate a session key. In contrast to RSA, they have the advantage that even if an attacker obtains the private RSA key it is not possible to decrypt already captured connections as it is not possible to extract the session key from the network traffic. This property is called perfect forward secrecy (PFS).

Finally, the communication between the endpoints is protected with symmetric encryption and message authentication based on symmetric keys derived from the master secret. TLS does not rely on a single fixed cryptographic algorithm; instead it provides a flexible framework that can support a large variety of encryption algorithms. At the beginning of a TLS session, the endpoints negotiate which algorithms and parameters to use.

TLS records

Internally, TLS is composed of several sub-protocols. The lower protocol layer is the TLS record protocol, which is used to exchange control and data messages between the communication partners. Each message of the record protocol contains the content type of a record, the TLS version, the length of a data fragment, and the data fragment (compressed, integrity protected and encrypted using the negotiated algorithms). At this layer, only the data fragment is encrypted, whereas all other fields (record type, TLS version and length) are exchanged in plain text.

Key negotiation and derivation

The key negotiation process is used to exchange the cryptographic parameters of the upcoming encrypted network session. The client initiates the protocol by sending a TLS record with the content type Client Hello (CH) and the server responds with a Server Hello (SH). These messages do not only define which algorithms to use, but are also used to exchange a client and server random value. Afterwards, the client and server define a premaster secret, for example by using the DH algorithm. In the initial handshake of TLS, no encryption is used, and thus the client and server random value are exchanged in plain text. If keys are renegotiated on an already encrypted TLS channel, the parameters will be encrypted with the still active configuration.

The premaster secret, client random, and server random are used to calculate the master secret of a connection. The master secret is used together with the server and client random to compute the derived keys using a pseudo random function (PRF). Typically, they comprise a MAC secret key used to verify the integrity of a TLS record (*write_MAC_key*), the data encryption key used to encrypt the payload of a TLS record (*write_key*), and an initialization vector (*write_IV*).

In the TLS protocol, a dedicated message is used to signal the transition to new cryptographic parameters. After completing the computation of the master secret, each communication partner sends a Change Cipher Spec