Where can HTTPS Connections be Broken ?

Tao Wan

Huawei Canada

January 8, 2016

1 Introduction

HTTPS is intended to establish an end to end secure channel between a browser and a website (namely origin website thereafter), providing origin authentication, data confidentiality, and data integrity. Billion of users depend on HTTPS on a daily basis for email, online banking, and online shopping, among other activities, expecting their communications to be private between themselves and the origin websites. However, the reality is far from satisfactory.

HTTPS connection is often broken into two or more segments with or without end user's awareness, introducing one or more middle mans into a communication intended to be end-to-end secure. Such segmentation of HTTPS could be by design (i.e., agreed by an end user or the origin website) or by accident (i.e., unilateral action by a third party without being agreed by either end user or the origin website).

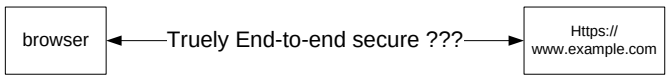

Figure 1: Is HTTPS truely end to end secure ?

We plan to publish a series of articles discussing security risks faced by HTTPS, particularly where HTTPS connections can be broken, how they can be broken, and what security problems could arise from broken HTTPS connections. In this article, we discuss where an end-to-end HTTPS connection can be broken.

2 Where to break HTTPS

There are three places where an HTTPS connection can be broken (see Figure 2). **First**, it can be broken on the client side , i.e., inside the device from which HTTPS connection is originated or inside the network connecting the client

device to the Internet. Examples of client side networks include home network, office network, and campus network, among others. **Second**, it can be broken in the middle, i.e., inside the Internet over which an HTTPS connection is established. **Third**, it can be broken on the server side, i.e., inside the facility hosting the origin website.

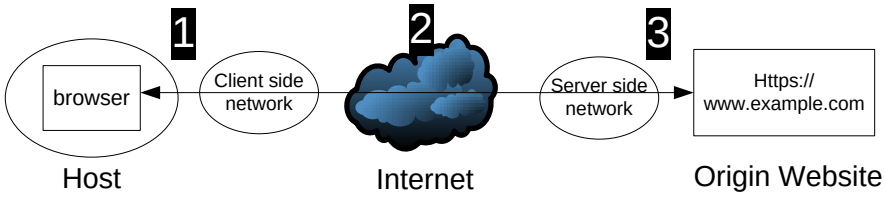

Figure 2: Where HTTPS can be broken?

2.1 On the Client Side

On the client side, HTTPS connections can be broken inside a device by certain software installed on the client such as anti-virus software (e.g., Kaspersky) and parental control tools. Those client side tools function as a man-in-the-middle (MITM) between the browser and the origin website to provide security services to the end user such as content filtering and access control.

While intended to provide more security, client side HTTPS proxy could sometime make end users more vulnerable to MITM attacks. For example, a client side HTTPS proxy often installs its own self-signed root CA certificate into a browser's trusted certificate store, allowing the proxy to sign certificate for any website. When functioning as a proxy communicating with an origin website, the proxy does not trust its own root CA, preventing a malicious party in possession of the private key associated with this root CA to launch MITM attacks. However, after stopping functioning as a proxy (e.g., after expired or uninstalled), some tools may not remove their own root CA certificates from the trust store, leaving users vulnerable to MITM attacks, particularly when a pre-configured static root CA key pair is used by all software installations. An NDSS 2016 paper [1] provides a comprehensive study of this topic, which we will discuss in the near future.

HTTPS connections could also be broken by an HTTPS proxy located on the client side network. For example, an HTTPS proxy could be installed inside a home network to protect children from accessing harmful websites. One can also use such proxy (in addition to DNS manipulation e.g., via /etc/hosts) to regular other family members from accessing certain websites, e.g., accessing certain e-commerce websites during the single day (November 11th) to avoid excessive spending. Many enterprises also require the use of HTTPS proxy to regulate Internet access and protect corporate intellectual properties.

In those cases, HTTPS is broken into two segments, client to proxy, and proxy to the origin website. Client side segmentation of HTTPS is usually

by design, and may or may not be known to end users. However, it is likely unknown to the origin website unless additional analysis is conducted by the origin website.

2.2 In the Middle

An HTTPS connection often traverses over the Internet to reach the origin website, and can also be broken in the middle way. We consider three scenarios, where HTTPS connections could be broken in the middle, acknowledging other cases are possible.

First, an HTTPS connection could be broken by a transparent HTTPS proxy, e.g., deployed by an ISP for caching purpose. Transparent proxies are commonly deployed inside ISP networks to accelerate content access. An transparent proxy is usually unknown to both the end user and the origin website, since neither is required to do any configuration. Note that an origin website may learn the existence of a transparent proxy by further analysis, e.g., by checking for certain HTTP headers such as X-forwarded-For header or Via header.

Second, an HTTPS connection could be broken by a malicious third party in control of the HTTPS path to attack end users. Such attacks are not by imagination, and likely unknown to both end users and the origin website.

Third, an HTTPS could terminate at a CDN server, instead of the origin website. Many websites use CDN services for both speed acceleration and security services such as DDoS mitigation and web application firewall (WAF). A CDN MITM is by design of the origin website but usually unaware by end users. When HTTPS meets CDN, a significant number of security problems could arise. This subject has been well studied in [3]. Please see [2] for a summary by Prof. Duan Haixin.

2.3 On the Server Side

Some websites, particular with high load, often deploy hardware based TLS acceleration to reduce latency incurred by TLS handshaking and content encryption and decryption. Such TLS accelerator is usually deployed in front of a server farm co-located with the accelerator within the same physical facility. It performs both TLS acceleration and load balancing, among other tasks. In this case, HTTPS terminates at the accelerator, and HTTP is often used between the accelerator and an origin website. An TLS accelerator MITM is introduced by design, but unknown to end users.

3 Conclusion

We discuss a number of scenarios where HTTPS connections could be broken (see Table 1 for a summary). Note that several scenarios could happen to a single HTTPS connection at the same time, and each scenario may occur

HTTPS MITM		Known to End Users	Known to Origin Websites
Client Side	Host Based	maybe	No
	Network Based	No	No
Network Side	ISP	No	Maybe
	CDN	No	Yes
	Others	No	No
Server Side		No	Yes

Table 1: A Summary of HTTPS Proxy Locations

multiple times, breaking an HTTPS connection into three or more segments. For example, an HTTPS connection may be broken once by a client side proxy, and then several more times by cascading CDNs in the middle.

To summarize, we would like to reiterate that HTTPS cannot always provide true end to end secure communication. Highly sensitive information such as user credentials and payment information might need message level protection to prevent them from being disclosed to middle mans, either legitimate or malicious. In next article, we will discuss technical details of how to break HTTPS connections. So stay tuned.

References

[1] Xavier de Carné de Carnavalet and Mohammad Mannan. Killed by Proxy: Analyzing Client-end TLS Interception Software. In *Network and Distributed System Security Symposium (NDSS 2016)*, San Diego, CA, USA, 2016. Internet Society.

[2] Haixin Duan. How NSA Intercepts Google's Encrypted Traffic - When HTTPS Meets CDN. http://www.inforsec.org/wp/?p=230, December 2015.

[3] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. When https meets cdn: A case of authentication in delegated service. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 2014.