End Of Life for TLS 1.0 and 1.1 Support

Table of Contents:

Introduction

This document presents short guidance and formal notification on rapidly identifying and removing Transport Layer Security (TLS) protocol version 1.0 dependencies in Usercentrics services and systems.

TLS 1.0 is a security protocol first defined in 1999 for establishing encrypted channels over computer networks. While no longer the default security protocol in use by modern operating systems, TLS 1.0 is still supported for backwards compatibility. Evolving regulatory requirements as well as new security vulnerabilities in TLS 1.0 provide corporations with the incentive to disable TLS 1.0 entirely.

Usercentrics recommends to get ahead of this issue by removing TLS 1.0 dependencies in the environments and disabling TLS 1.0 at the operating system level where possible.

The goal of this document is to provide recommendations which can help remove technical blockers to disable TLS 1.0 while at the same time increasing visibility into the impact of this change to your own customers. Completing such investigations can help reduce the business impact of the next security vulnerability in TLS 1.0. For the purposes of this document, references to the deprecation of TLS 1.0 also include TLS 1.1.

Support for Deprecation

A quick way to determine what TLS version will be requested by various clients when connecting to our online services is by referring to the Handshake Simulation at Qualys SSL Labs. This simulation covers client OS/browser combinations across manufacturers. See **Appendix A** at the end of this document for a detailed example showing the TLS protocol versions negotiated by various simulated client OS/browser combinations when connecting to www.usercentrics.com.

If not already complete, it is highly recommended to conduct an inventory of operating systems used by your enterprise, customers and partners (the latter two via outreach/communication or at least HTTP User-Agent string collection). This inventory can be further supplemented by traffic analysis at your enterprise network edge. In such a situation, traffic analysis will yield the TLS versions successfully negotiated by customers/partners connecting to your services, but the traffic itself will remain encrypted.

TLS security considerations

Why are we deprecating TLS 1.0 and 1.1?

TLS 1.0 and 1.1 are out-of-date protocols that do not support modern cryptographic algorithms, and they contain security vulnerabilities that may be exploited by attackers. The Internet Engineering Task Force is also planning to officially deprecate both protocols. In addition, the vast majority of encrypted Internet traffic is now over TLS 1.2, which was introduced over a decade ago.

The older TLS versions are riddled with security vulnerabilities. As such, these protocols are updated over time to patch out these vulnerabilities and keep users safe. TLS 1.0 came out in 1999 and has had many issues with heartbleed, POODLE, CRIME, etc. That said, it's been a long time coming for companies to drop their support of 1.0 and 1.1. When it comes to TLS deprecation, many other tech companies have chosen to sunset these old protocols as well. In March 2020, all four major internet browser providers ended their support of TLS 1.0 and 1.1 – which was a major push in the right direction for better security.

When will this change happen?

TLS 1.0 and TLS 1.1 deprecation will take place at the **end of January, 2021**. After the date of deprecation, you will not be able to connect to Usercentrics services using browsers or applications not compatible with TLS 1.2. Usercentrics encourages users to quickly abandon older versions of TLS to avoid exposure to security vulnerabilities.

How does TLS affect you?

Usercentrics services are web-based and can only be engaged through a secure network connection. TLS helps ensure a secure and reliable connection between your browser or server and Usercentrics web services, which includes anything that uses Usercentrics's API, such as REST, App SDK, Mobile SDK, and more.
As technology evolves, security standards are upgraded to ensure higher levels of privacy and data integrity. However, older applications are not updated to include the latest standards. As the acceptable level of security rises, these older, less secure applications are left behind.
To be able to connect to Usercentrics web services, update your browsers and application frameworks to a version that supports TLS 1.2.

How does TLS affect end-users?

Usercentrics web services will be served to your visitors through TLS 1.2 secured connections. Any browser updated since late 2013 (except Chrome, updated since 2017) will be TLS 1.2 compliant; further, Apple, Google, Microsoft, and Mozilla have all announced their plan to completely disable TLS 1.0 & 1.1 support by the first half of 2020, so we expect very minimal impact to visitors. If visitors report a loss of connectivity to Usercentrics web services as a result of this change, they will need to update to a compatible browser version.

Will this affect users that have up-to-date devices?

We don't expect this to impact your users. All modern browsers already support TLS 1.2. Google, Microsoft, Apple, and Mozilla have all either dropped or have announced they will soon drop support for TLS 1.0 and 1.1.

What error message will return to a non-compliant connection?

The exact error messaging returned depends on the browser or application framework being used to connect to Usercentrics web services. Some examples include but are not limited to:

- Unable to connect to the service
- Service not available
- Error in connection

To resolve these errors, the browser or application framework must be updated to a version compatible with TLS 1.2.

What Browsers and OS versions support TLS 1.2?

Browser Support for TLS 1.2

Browser	*TLS 1.2 Supported* *(Not enabled by default)*	*Enabled by default*
Internet Explorer	Version 8	Version 11
Microsoft Edge		All Versions
Google Chrome*		Version 29
Mozilla Firefox	Version 23	Version 27

Opera		Version 17
Apple Safari		Version 7
iOS Safari		Version 5

**For Google Chrome 22 to 37: TLS 1.1 and TLS 1.2 are compatible when running on Windows XP SP3, Vista, or newer (desktop), OS X 10.6 (Snow Leopard) or newer (desktop), or Android 2.3 (Gingerbread) or newer (mobile).*

Mobile Browser Support for TLS 1.2

Browser	*Enabled by default*
Mobile Browser	Compatible Versions
Google Android OS Browser	Android 5.0+
Chrome for Android	V30+
Firefox for mobile	V27+
Opera Mobile	V57+
Apple Safari	IOS 5+

OS Support for TLS 1.2

OS	*TLS 1.2 Supported (Not enabled by default)*	*Enabled by default*
Windows (Desktop)	Windows 8	Windows 7 SP1 and later
Mac OSX		10.9

Application Frameworks

Java	.NET	OpenSSL
Java 8, or later	.NET 4.6, or later	OpenSSL 1.01, or later
Java 7, with TLS 1.2 enabled in app	.NET 4.5, with TLS 1.2 enabled in app	

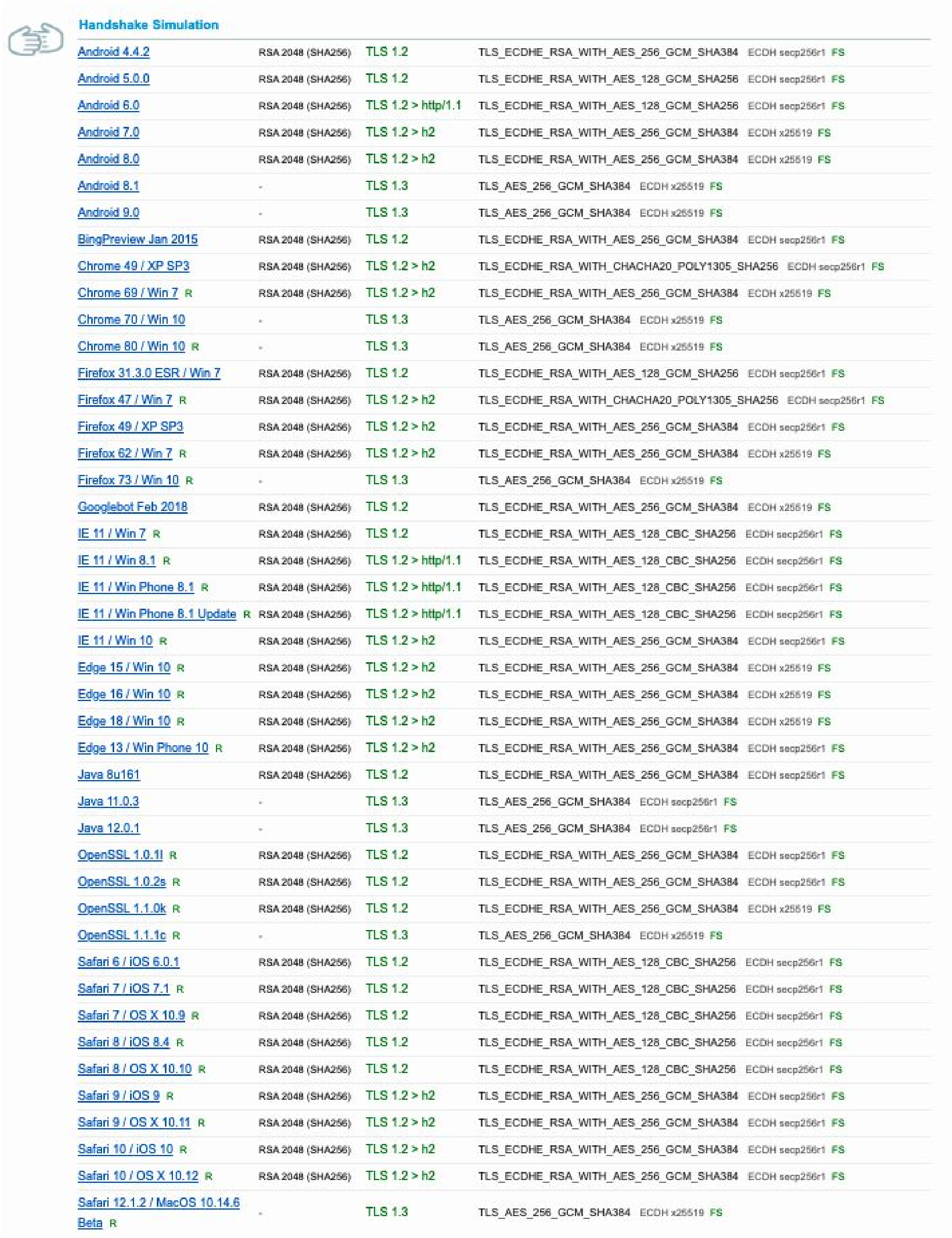
Conclusion

Removing TLS 1.0 dependencies is a complicated issue to drive end to end. Usercentics alongside the industry leaders are taking action on this today to ensure that their services are more secure by default, from their OS components and development frameworks up to the applications/services built on top of them. Following the recommendations made in this document will help your enterprise chart the right course and know what challenges to expect. It will also help your own customers become more prepared for the transition.

References

1. IETF TLS 1.0 deprecation
2. NIST TLS Guidelines
3. TLS 1.0 RFC 2246
4. TLS 1.3 Draft
5. Man-in-the-Middle
6. TLS Protocol Compatibility
7. SSL Pulse
8. SSL Browser check
9. SSL Server check

Appendix A: Handshake Simulation for various clients connecting to www.usercentrics.com, courtesy SSLLabs.com



Handshake Simulation

Android 4.4.2	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Android 5.0.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1	FS
Android 6.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1	FS
Android 7.0	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Android 8.0	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Android 8.1	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
Android 9.0	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
BingPreview Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Chrome 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ECDH secp256r1	FS
Chrome 69 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Chrome 70 / Win 10	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
Chrome 80 / Win 10 R	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
Firefox 31.3.0 ESR / Win 7	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1	FS
Firefox 47 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ECDH secp256r1	FS
Firefox 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Firefox 62 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Firefox 73 / Win 10 R	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
Googlebot Feb 2018	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
IE 11 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
IE 11 / Win 8.1 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
IE 11 / Win Phone 8.1 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
IE 11 / Win Phone 8.1 Update R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
IE 11 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Edge 15 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Edge 16 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Edge 18 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
Edge 13 / Win Phone 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Java 8u161	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Java 11.0.3	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH secp256r1	FS
Java 12.0.1	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH secp256r1	FS
OpenSSL 1.0.1l R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
OpenSSL 1.0.2s R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
OpenSSL 1.1.0k R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH x25519	FS
OpenSSL 1.1.1c R	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS
Safari 6 / iOS 6.0.1	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
Safari 7 / iOS 7.1 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
Safari 7 / OS X 10.9 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
Safari 8 / iOS 8.4 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
Safari 8 / OS X 10.10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDH secp256r1	FS
Safari 9 / iOS 9 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 9 / OS X 10.11 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 10 / iOS 10 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 10 / OS X 10.12 R	RSA 2048 (SHA256)	TLS 1.2 > h2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1	FS
Safari 12.1.2 / MacOS 10.14.6 Beta R	-	TLS 1.3	TLS_AES_256_GCM_SHA384	ECDH x25519	FS