Introduction

SSL/TLS is a deceptively simple technology. It is easy to deploy, and it just works . . . except that it does not, really. The first part is true—SSL is easy to deploy—but it turns out that it is not easy to deploy correctly. To ensure that SSL provides the necessary security, users must put more effort into properly configuring their servers.

In 2009, we began our work on SSL Labs because we wanted to understand how SSL was used and to remedy the lack of easy-to-use SSL tools and documentation. We have achieved some of our goals through our global surveys of SSL usage, as well as the online assessment tool, but the lack of documentation is still evident. This document is a first step toward addressing that problem.

Our aim here is to provide clear and concise instructions to help overworked administrators and programmers spend the minimum time possible to obtain a secure site or web application. In pursuit of clarity, we sacrifice completeness, foregoing certain advanced topics. The focus is on advice that is practical and easy to understand. For those interested in advanced topics, we provide references at the end of the guide.

Section 1: Private Key and Certificate

The quality of the protection provided by SSL relies on the private key, which lays down the foundation for the security, and the certificate, which communicates the identity of the server to its visitors.

1.1 Use 2048-bit private keys

Aim to use 2048-bit private keys for all your servers. Keys of this length are secure and will stay secure for a very long time. Your existing 1024-bit keys can stay in place, but you should plan to upgrade them the next time the certificates are up for renewal, or within the next two years.

1.2 Protect private keys

Treat your private keys as an important asset, restricting access to the smallest possible group of employees while still keeping the arrangements practical. Recommended policies include the following:

• Password-protect encryption keys to prevent keys from being compromised when they are stored in backup systems.
• If there is a compromise, revoke old certificates and generate new keys to use with new certificates.
• Renew certificates every year—always with new private keys.

1.3 **Ensure sufficient domain name coverage**

Ensure that your certificates cover all the names you wish to use with a site. For example, your main name is www.example.com, but you may also have www.example.net configured. Your goal is to avoid invalid certificate warnings, which will confuse your users and weaken their trust.

Even when there is only one name, remember that you cannot control how your users arrive at the site or how others link to the site. In most cases, you should ensure that the certificate works with and without the www prefix (e.g., for both example.com and www.example.com). The rule of thumb is this: a secure web server should have a certificate that is valid for every DNS name configured to point to it.

Wildcard certificates are generally best avoided. Although they are not any less secure from a strict technical point of view, the way in which they are typically handled (especially in larger organizations) makes them less secure in practice.

1.4 **Obtain certificates from a reliable Certificate Authority**

Select a Certificate Authority (CA) that is reliable, that is, one that is serious about its certificate business and about security. Consider the following criteria while selecting your CA:

• **Substantial market share**: A CA that meets this criterion will not likely have all its certificates easily recalled, which was the case with some smaller ones in the past.

• **Business focus**: CAs whose activities constitute a substantial part of their business have everything to lose if something goes terribly wrong, and they will not neglect their certificate business by chasing potentially more lucrative opportunities elsewhere.

• **Services offered**: The CA should provide support for CRL and OCSP revocation as well as allow you to reissue your certificates online and as many times as necessary.

• **Freedom of deployment**: No certificate licensing limitations (i.e., you can deploy your certificates on as many servers as you like).

• **Certificate management options**: If you need a large number of certificates, choose a business with a good management user interface that enables you to manage all of your certificates as well as to delegate management to multiple user accounts.

• **Technical support**: Choose a business that will give you good support if you need it.

**Section 2: Configuration**

With correct SSL server configuration, you ensure that your credentials are properly presented to the site’s visitors, that only secure cryptographic primitives are used, and that all known weaknesses are mitigated.

2.1 **Ensure that the certificate chain is valid**

In most deployments, the server certificate alone is insufficient; two or more certificates are needed to establish a complete chain of trust. A common problem is configuring the server certificate correctly but
forgetting to include other required certificates. Further, although these other certificates are typically valid for longer periods of time, they too expire, and when they do, they invalidate the entire chain. An invalid certificate chain renders the actual server certificate invalid. In practice, this problem is sometimes difficult to diagnose because some browsers can reconstruct a complete chain, and some can’t. Testing with a tool that is designed to highlight configuration flaws is the only way to be sure.

2.2 **Use only secure protocols**

There are five protocols in the SSL/TLS family: SSL v2, SSL v3, TLS v1.0, TLS v1.1, and TLS v1.2. Of these:

- **SSL v2 is insecure and must not be used.**
- **SSL v3 and TLS v1.0 largely still hold up**: we do not know of major security flaws when they are used for protocols other than HTTP. When used with HTTP, they can be made secure with careful configuration.
- **TLS v1.1 and v1.2 are without known security issues.** Unfortunately, many server and client platforms do not support these newer protocol versions.

The best practice is to use TLS v1.0 as your main protocol (making sure the BEAST attack is mitigated in configuration, as explained in subsequent sections) and TLS v1.1 and v1.2 if they are supported by your server platform. That way, the clients that support newer protocols will select them, and those that don’t will fall back to TLS v1.0.

You should always use the most recent versions of the protocol for security and the oldest (yet still secure) versions for interoperability with your customer base.

2.3 **Use only secure cipher suites**

To communicate securely, you must first ascertain that you are communicating directly with the desired party (and not through someone else who will eavesdrop), as well as exchanging data securely. In SSL/TLS, cipher sites are used to define how secure communication takes place. They are composed from varying building blocks with the idea of achieving security through diversity. If one of the building blocks is found to be weak or insecure, you can always rely on another building block that is supported.

Your goal should be thus to use only suites that provide authentication and encryption of 128 bits or stronger. Everything else must be avoided:

- **Anonymous Diffie-Hellman (ADH) suites** do not provide authentication.
- **NULL cipher suites** provide no encryption.
- **Export key exchange suites** use authentication that can easily be broken.
- **Suites with weak ciphers** (typically of 40 and 56 bits) use encryption that can easily be broken.

2.4 **Control cipher suite selection**

In SSL v3 and later versions, clients submit a list of cipher suites that they support, and servers choose one from the list to establish a secure communication channel. Not all servers do this well, however—some will select the first supported suite from the list. Having servers select the right cipher suite is critical for security (see Section 2.6).

2.5 **Disable client-initiated renegotiation**

In SSL/TLS, renegotiation allows parties to stop exchanging data for a moment and to renegotiate
how the communication is secured. There are some cases in which renegotiation needs to be initiated by the server, but there is no clear need for clients to do so. In addition, allowing clients to initiate renegotiation makes it easier for them to perform Denial of Service attacks.

2.6 Mitigate known problems

Nothing is perfectly secure, and at any given time there may be issues with the security stack. It is good practice to keep an eye on what happens in the security world and to adapt to situations as necessary. At the very least, you should apply vendor patches as soon as they become available.

At this time, two issues require your attention:

- **Disable insecure renegotiation.** In 2009, the renegotiation feature was found to be insecure. Most vendors have issued patches by now or, at the very least, provided workarounds for the problem.

- **Prioritize RC4 to mitigate the BEAST attack.** The 2011 BEAST attack is a practical attack based on a protocol problem that was discovered in 2004. Despite having been addressed in TLS v1.1 in 2006, the problem is still relevant because most clients and servers do not support newer protocol versions. Practical mitigation requires that your servers speak only RC4 when using TLS v1.0 or SSL v3.

### Section 3: Performance

Security is our main focus in this guide, but we must also pay attention to performance: a secure service that does not satisfy performance criteria will no doubt be dropped. However, because SSL configuration does not usually have a significant overall performance impact, we are limiting the discussion in this section to the common configuration problems that result in serious performance degradation.

3.1 Do not use private keys that are longer than necessary

The cryptographic handshake, which is used to establish secure connections, is an operation whose cost is highly influenced by private key size. Using a key that is too short is insecure, but using a key that is too long will result in too much security and result in slow operation. Currently, you can use 1024- and 2048-bit keys (with caveats, as explained in Section 1.1), but anything more than that is a waste of CPU power and will impair user experience.

3.2 Ensure that session resumption works

Session resumption is a performance-optimization technique that makes it possible to save the results of costly cryptographic operations and to reuse them for a period of time. A disabled or nonfunctional session resumption mechanism may introduce a significant performance penalty.

3.3 Use persistent connections (HTTP)

These days, most of the overhead of SSL comes not from the CPU-hungry cryptographic operations but from network latency. An SSL handshake is performed after the TCP handshake completes; it requires a further exchange of packets. To minimize the cost of latency, you enable HTTP persistence (keep-alives), allowing your users to submit many HTTP requests over a single TCP connection.

3.4 Enable caching of public resources (HTTP)
When communicating over SSL, browsers assume that all traffic is sensitive. They will typically use the memory to cache certain resources, but once you close the browser, all the content may be lost. To get a performance boost and enable long-term caching of some resources, mark public resources (e.g., images) as public by attaching the Cache-Control: public response header to them.

Section 4: Application Design and Implementation (HTTP)

The HTTP protocol and the surrounding platform for web application delivery continued to evolve rapidly after SSL was born. As a result of that evolution, the platform now contains features that can be used to defeat encryption. In this section we list those features, as well as ways to use them securely.

4.1 Encrypt 100% of your web site traffic

The fact that encryption is optional is probably one of the biggest security problems today. We see the following problems:

• No SSL on sites that need it
• Sites that have SSL but that do not enforce it
• Sites that mix SSL and non-SSL content, sometimes even within the same page
• Sites with programming errors that subvert SSL

Although many of these problems can be mitigated if you know exactly what you’re doing, at the end of the day the only way to reliably protect web site communication is to enforce encryption throughout—with no exceptions.

4.2 Ensure that cookies are secured

To be properly secure, a web site requires all its cookies to be marked as secure, too. Failure to secure cookies makes it possible for an active man-in-the-middle (MITM) attacker to tease some information out through clever tricks, even on web sites that are 100% encrypted.

4.3 Ensure that mixed content is not used

Mixed-content pages are those that are transmitted over SSL but include resources (e.g., JavaScript files, images, CSS files) that are not transmitted over SSL. Such pages are not secure. An active MITM attacker can piggyback on a single unprotected JavaScript resource, for example, and hijack the entire user session.

4.4 Enable HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS) is an SSL safety net: technology designed to ensure that security remains intact even in the case of configuration problems and implementation errors. To activate HSTS protection, you set a single response header in your websites. After that, browsers that support HSTS (at this time, Chrome and Firefox) will respect your instructions.

The goal of HSTS is simple: after activation, do not allow insecure communication with your website. It achieves this goal by automatically converting all plain-text links to secure ones. As a bonus, it will also disable click-through SSL certificate warnings. (SSL certificate warnings are an indicator of an active
MITM attack. Studies have shown that most users click through these warnings, so it is in your best interest to never allow them.)

4.5 Disable caching of sensitive content

The goal of this recommendation is to ensure that sensitive content is communicated to only the intended parties and that it is treated as sensitive. Although proxies do not see encrypted traffic and cannot share content among users, the use of cloud-based application delivery platforms is increasing, which is why you need to be very careful when specifying what is public and what is not.

4.6 Ensure that there are no other vulnerabilities

This item is a reminder that SSL does not equal security. SSL is designed to address only one aspect of security – confidentiality and integrity of the communication between you and your users—but there are many other threats that you need to deal with. In most cases, that means ensuring that your website does not have other weaknesses.

4.7 Understand and acknowledge third-party trust

Web sites often use third-party services activated via JavaScript code downloaded from another server. A good example of such a service is Google Analytics, which is used on large parts of the Internet. Such inclusion of third-party code creates an implicit trust connection that effectively gives the other party full control over your web site. The third party may not be malicious, but large providers of such services are increasingly seen as targets. The reasoning is simple: if a large provider is compromised, the attacker is automatically given access to all the sites that depend on the service.

If you follow the advice from Section 4.3, at least your third-party links will be encrypted. However, learn what services your sites use, and either remove them—and replace them with safer alternatives—or accept the risk of their continued use.

Section 5: Validation

With many configuration parameters available for tweaking, it is difficult to know in advance what impact certain changes will have. Further, changes are sometimes made accidentally; software upgrades can introduce changes silently. For that reason, we advise that you use a comprehensive SSL/TLS assessment tool initially to verify your configuration to ensure that you start out secure, and then periodically to ensure that you stay secure. For public web sites, the free online assessment tool on the SSL Labs web site (see the References section at the end of this guide) is hard to beat.

Advanced Topics

The following advanced topics are outside the scope of our guide. They require a deeper understanding of SSL/TLS and Public Key Infrastructure (PKI), and they are still being debated by experts.

The advanced topics are:

- **Public key pinning** (or, to give its full name, the Public Key Pinning Extension for HTTP) is designed to give
website operators the means to restrict which certificate authorities can issue certificates for their servers. This standard is currently being developed by the Web Security Working Group.

- **Forward secrecy** is a protocol feature that enables secure conversations that are not dependent on the server’s private key. With cipher suites that do not allow for forward secrecy, someone who can recover a server’s private key can decrypt all earlier encrypted conversations, assuming they have them recorded.

## References

For further information, please refer to the following resources on the SSL Labs web site and the Qualys Security Labs blog:

- SSL Labs: SSL/TLS Assessment Tool
  [https://www.ssllabs.com/ssldb/](https://www.ssllabs.com/ssldb/)
- SSL Labs: SSL Server Rating Guide
- TLS Renegotiation and DoS Attacks
- BEAST Attack Mitigation
- SSL and TLS Authentication Gap Discovered

## Acknowledgments

Thanks to Marsh Ray (PhoneFactor), Nasko Oskov (Google), Adrian F. Dimcev, and Ryan Hurst (GlobalSign) for their valuable feedback and help in crafting this document.

## About SSL Labs

SSL Labs is Qualys’s research effort to understand SSL/TLS and PKI as well as to provide tools and documentation to assist with assessment and configuration. Since 2009, when SSL Labs was launched, hundreds and thousands of websites have checked and improved their SSL configuration using the free online assessment tool.