[Skip to main content](https://lms.alnafi.com/xblock/block-v1:alnafi+DCCS102+2025_DCCS+type@vertical+block@174f8b54437148f48395f5dc844ff1bc?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#main)

**Ref**[**https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide**](https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide)

**SSL Server Rating Guide**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The Secure Sockets Layer (SSL) protocol is a standard for encrypted network communication.                    We feel that there is surprisingly little attention paid to how SSL is configured,                    given its widespread usage. SSL is relatively easy to use, but it does have its traps. This                    guide aims to establish a straightforward assessment methodology, allowing administrators                    to assess SSL server configuration confidently without the need to become SSL                    experts.    Complete Guide:     * [SSL Server Rating Guide](https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide)   **SSL Server Rating Guide**                naumanshah03 edited this page on Feb 4, 2020        ·        [36 revisions](https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide/_history)    **Version 2009q (31 January 2020)**  The Secure Sockets Layer (SSL) protocol is a standard for encrypted  network communication. We feel that there is surprisingly little  attention paid to how SSL is configured, given its widespread usage. SSL is relatively easy to use, but it does have its traps. This guide aims  to establish a straightforward assessment methodology, allowing  administrators to assess SSL server configuration confidently without  the need to become SSL experts.  **Methodology Overview**  Our approach consists of four steps:   1. We first look at a certificate to verify that it is valid and trusted. 2. We inspect server configuration in three categories: 3. Protocol support 4. Key exchange support 5. Cipher support 6. We combine the category scores into an overall score (expressed as a number between 0 and 100). A zero in any category will push the overall score to zero. Then, a letter grade is calculated, using the table  below. 7. We then apply a series of rules (documented in the Changes section)  to handle some aspects of server configuration that cannot be expressed  via numerical scoring. Most rules will reduce the grade (to A-, B, C, D, E, or F) if they encounter an unwanted feature. Some rules will  increase the grade (to A+), to reward exceptional configurations. 8. In certain situations we avoid the standard A-F grades if we think  we've encountered a situation that's out of scope. That's the case with  the M grade (certificate name mismatch) and the T grade (site  certificate is not trusted). When there is no certificate trust, the  actual security grade doesn't matter because active network attackers  can subvert connection security.   **Table 1. Letter grade translation**   | **Numerical Score** | **Grade** | | --- | --- | | score >= 80 | A | | score >= 65 | B | | score >= 50 | C | | score >= 35 | D | | score >= 20 | E | | score < 20 | F |   Our methodology was initially designed to be simple and  straightforward, but has, unfortunately, gotten more complicated over  time. This document has not been fully updated to reflect the changes.  In the next major version, we will start afresh, aiming to go back to  the original simplicity.  **What This Guide Does Not Cover**  Our immediate goal is to focus on those configuration problems whose  presence can be determined remotely and without manual assessment. It is only a fully automated approach that makes it possible to perform a  large-scale assessment of SSL configuration practices. Our aim is to  scan all SSL servers on the public Internet. In focusing on automation, we have decided not to look for certain  problems. We will list those problems in this guide, and hopefully find  ways to enhance our automation to include them in a future version of  this guide. Some of those problems are listed here:  **Certificate quality** Three certificate types are  currently in use: domain-validated, organization-validated and  extended-validation (EV) certificates. This guide requires a certificate to be correct, but does not go beyond this basic requirement. The  domain-validated and organization-validated certificates are generally  treated in the same way by the current generation of browser software,  and thus offer similar assurance to users. EV certificates are treated  significantly better and, generally, they are recommended for high-value web sites. Without a reliable way to determine the purpose of a web  site, however, there is little that this guide can do to assess whether a certificate used on an arbitrary site is suitable for the purpose of  the site.  **Session hijacking issues in web applications** There  are several ways in which web applications can subvert SSL and make it  less effective. For example, session cookies that are not marked as  secure can be retrieved by a determined attacker, leading to session  hijacking and thus application compromise. Such problems are not the  fault of SSL, but they do affect its practical applications  nevertheless. Detecting web application–specific problems is non-trivial to perform in an automated fashion, and this version of the guide does  not attempt to do it. We leave this problem for the consideration in the future. In the meantime, to remove any doubt that might exist about the seriousness of the above-mentioned issues, we will state that any  application that incorrectly implements session token propagation should be awarded a zero score.  **What Should My Score Be?**  We don’t know. In order to tell you whether you’ve configured your  SSL server correctly, we would need to know what your site does. Because different web sites have different needs, it is not possible for us to  choose any one configuration and say that it works for everyone. But we  can do two things. First, we can give you some general configuration  advice and tell you what you should never do. Second, we can give you  some general guidance using examples of what other web sites do. If  that’s what you are interested in, skip to the end of this document for  more information.  **Is SSL Enough?**  No. A non-trivial web site cannot be secure if it does not implement  SSL, but SSL is not enough. SSL deals with only one aspect of security,  and that is the security of the communication channel between a web site and its users. SSL does not and cannot address a number of possible  security issues that may exist on a web site. View SSL as a foundation  on which to build, but the foundation alone is not enough.  **Acknowledgments**  The first version of this guide was written by Ivan Ristic, and  subsequently enhanced by the contributions from the following  individuals, listed in alphabetical order: Christian Bockermann,  Christian Folini, Robert Hansen, Ofer Shezaf and Colin Watson.  **Certificate Inspection**  Server certificate is often the weakest point of an SSL server  configuration. A certificate that is not trusted (i.e., is not  ultimately signed by a well-known certificate authority) fails to  prevent man-in-the-middle (MITM) attacks and renders SSL effectively  useless. A certificate that is incorrect in some other way (e.g., a  certificate that has expired) erodes trust and, in the long term,  jeopardizes the security of the Internet as a whole.  For these reasons, any of the following certificate issues immediately result in a zero score:   * Domain name mismatch * Certificate not yet valid * Certificate expired * Use of a self-signed certificate * Use of a certificate that is not trusted (unknown CA or some other validation error) * Use of a revoked certificate * Insecure certificate signature (MD2 or MD5) * Insecure key   **Note**  Some organizations create their own (private) CA certificates, a  practice that is entirely legitimate, provided such CA certificates are  distributed, in a safe manner (e.g., through the use of customized  browsers) to all those who need it. Without the access to such  certificates we may not be able to verify that a site we are inspecting  has a trusted certificate, but we believe that such sites will be  relatively rare. Such issues can be considered on a case-by-case basis.  **Scoring**  SSL is a complex hybrid protocol with support for many features  across several phases of operation. To account for the complexity, we  rate the configuration of an SSL server in three categories, as  displayed in . We calculate the final score as a combination of the  scores in the individual categories, as described in the “Methodology  Overview” section.  **Table 2. Criteria categories**   | **Category** | **Score** | | --- | --- | | Protocol support | 30% | | Key exchange | 30% | | Cipher strength | 40% |   The sections that follow explain the rating system for each of the categories.  **Protocol Support**  First, we look at the protocols supported by an SSL server. For  example, both SSL 2.0 and SSL 3.0 have known weaknesses. Because a  server can support several protocols, we use the following algorithm to  arrive to the final score:   1. Start with the score of the best protocol. 2. Add the score of the worst protocol. 3. Divide the total by 2.   **Table 3. Protocol support rating guide**   | **Protocol** | **Score** | | --- | --- | | SSL 2.0 | 0% | | SSL 3.0 | 80% | | TLS 1.0 | 90% | | TLS 1.1 | 95% | | TLS 1.2 | 100% |   **Key Exchange**  The key exchange phase serves two functions. One is to perform  authentication, allowing at least one party to verify the identity of  the other party. The other is to ensure the safe generation and exchange of the secret keys that will be used during the remainder of the  session. The weaknesses in the key exchange phase affect the session in  two ways:   * Key exchange without authentication allows an active attacker to  perform a MITM attack, gaining access to the complete communication  channel. * Most servers also rely on public cryptography for the key exchange.  Thus. the stronger the server’s private key, the more difficult it is to break the key exchange phase. A weak key, or an exchange procedure that uses only a part of the key (the so-called exportable key exchanges),  can result in a weak key exchange phase that makes the per-session  secret keys easier to compromise. Some servers use key exchange  mechanisms that do not depend on the private key (the key is still used  for authentication). Two popular algorithms are the ephemeral  Diffie-Hellman key exchange (DHE) and its Elliptic Crypto variation  ECDHE. If a separate key exchange mechanism is used, the overall  strength will depend on its strength and the strength of the private  key.   **Table 4. Key exchange rating guide**   | **Key exchange aspect** | **Score** | | --- | --- | | Weak key (Debian OpenSSL flaw) | 0% | | Anonymous key exchange (no authentication) | 0% | | Key or DH parameter strength < 512 bits | 20% | | Exportable key exchange (limited to 512 bits) | 40% | | Key or DH parameter strength < 1024 bits (e.g., 512) | 40% | | Key or DH parameter strength < 2048 bits (e.g., 1024) | 80% | | Key or DH parameter strength < 4096 bits (e.g., 2048) | 90% | | Key or DH parameter strength >= 4096 bits (e.g., 4096) | 100% |   **Note**  For suites that rely on DHE or ECDHE key exchange, the strength of DH parameters is taken into account when determining the strength of the  handshake as a whole. Many servers that support DHE use DH parameters  that provide 1024 bits of security. On such servers, the strength of the key exchange will never go above 1024 bits, even if the private key is  stronger (usually 2048 bits).  **Cipher Strength**  To break a communication session, an attacker can attempt to break  the symmetric cipher used for the bulk of the communication. A stronger  cipher allows for stronger encryption and thus increases the effort  needed to break it. Because a server can support ciphers of varying  strengths, we arrived at a scoring system that penalizes the use of weak ciphers. To calculate the score for this category, we follow this  algorithm:   1. Start with the score of the strongest cipher. 2. Add the score of the weakest cipher. 3. Divide the total by 2.   **Table 5. Cipher strength rating guide**   | **Cipher strength** | **Score** | | --- | --- | | 0 bits (no encryption) | 0% | | < 128 bits (e.g., 40, 56) | 20% | | < 256 bits (e.g., 128, 168) | 80% | | >= 256 bits (e.g., 256) | 100% |   **SSL Configuration Advice**  The configuration advice from the original document has been  superseded by a standalone document with comprehensive coverage of this  topic. You can download the SSL/TLS Deployment Best Practices document  from the SSL Labs web site.  **Changes**  We are planning to release a completely new version of the rating  guide in Q1 2017, building on what we have learned from the current  version. In the meantime, we are making small revisions in order to  react to the threats as they come and go.  **Changes in 2009c (7 February 2013)**  Changes to the grading criteria:   * SSL 2.0 is not allowed (F). * Insecure renegotiation is not allowed (F). * Vulnerability to the BEAST attack caps the grade at B. * Vulnerability to the CRIME attack caps the grade at B. * The test results no longer show the numerical score (0-100) because  we have realized that the letter grade (A-F) is more useful. In addition, we've taken the opportunity to remove the old configuration advice, directing the readers to our SSL/TLS Deployment Best Practices  document instead.   **Changes in 2009d (9 September 2013)**   * No longer require server-side mitigation for the BEAST attack.   **Changes in 2009e (21 January 2014)**   * Support for TLS 1.2 is now required to get the A grade. Without, the grade is capped a B. * If vulnerable to the Heartbleed attack, it will be given F. * If vulnerable to the OpenSSL CVE-2014-0224 vulnerability, it will be given F. * Keys below 2048 bits (e.g., 1024) are now considered weak, and the grade capped at B. * Keys under 1024 bits are now considered insecure (F). * This version introduces warnings as part of rating criteria. In most cases, warnings are about issues that do not yet affect the grade, but  likely will in the future. Server administrators are advised to correct  the warnings as soon as possible. * Warning: RC4 is used with TLS 1.1 or newer protocol. Because RC4 is  weak, the only reason to use it is to mitigate the BEAST attack. For  some, BEAST is still a threat. Because TLS 1.1 and newer are not  vulnerable to BEAST, there is no reason to use RC4 with them. * Warning: No support for Forward Security. * Warning: Secure renegotiation is not supported. * New grade A- is introduced for servers with generally good configuration that have one or more warnings. * New grade A+ is introduced for servers with exceptional  configurations. At the moment, this grade is awarded to servers with  good configuration, no warnings, and HTTP Strict Transport Security  support with a max-age of at least 6 months. * MD5 certificate signatures are now considered insecure (F). * Clarified that insecure certificate signatures affect the certificate score. This has always been the case for MD2. * Clarified that the strength of DHE and ECDHE parameters affects key  exchange scoring. This has always been the case, but previous revisions  of the text were not clear about it.   **Changes in 2009f (4 September 2014)**   * Don’t award A+ to servers that use SHA1 certificates.   **Changes in 2009g (15 October 2014)**   * Cap to C if vulnerable to POODLE. * Note: POODLE TLS is treated as a patchable and exploitable vulnerability, which means it gets an F.   **Changes in 2009h (30 October 2014)**   * Don’t award A+ to servers that don’t support TLS\_FALLBACK\_SCSV. * Cap to B if SSL 3 is supported.   **Changes in 2009i (8 December 2014)**   * Cap to B if RC4 is supported. * Cap to B if the chain is incomplete. * Fail servers that have SSL3 as their best protocol. * If using insecure DH parameters (less than 1024 bits) grade will be set to F. * If using export suites grade will be set to F.   **Changes in 2009j (20 May 2015)**   * Cap to B if using weak DH parameters (less than 2048 bits). * Increase CRIME penalty to C (was B). * Cap to C if RC4 is used with TLS 1.1+. * Cap to C if not supporting TLS 1.2.   **Changes in 2009k (14 October 2015)**   * Fail servers that support only RC4 suites.   **Changes in 2009l (4 March 2016)**   * Detect when RSA exponent 1 is used. This is insecure and gets an F. * Hosts that have HPKP issues can't have A+. * Added DROWN testing. Vulnerable servers get an F.   **Changes in 2009m (6 June 2016)**   * If vulnerable to CVE-2016-2107 (Padding oracle in AES-NI CBC MAC check) it will be given F.   **Changes in 2009n (19 January 2017 - 3 April 2017)**   * With this revision we introduced an early warning system about grade changes. We list two dates; one when we started warning about the new  grade and (to be added later), when the new grading becomes effective. * Introduce a penalty (C) for using 3DES (and other ciphers with block sizes of 64 bits) with TLS 1.1+. (Note: For a period of time this  document incorrectly stated that the penalty applies only to TLS 1.2.  Sorry.) * SHA1 certificates are now longer trusted (T). * Introduce an explicit penalty for using cipher suites weaker than  112 bits. This was necessary to address a flaw in the algorithm that  didn't sufficiently penalize these weak suites. * Moved this document to the SSL Labs wiki on GitHub.   **Changes in 2009o (8 May 2017)**   * If vulnerable to the Ticketbleed (CVE-2016-9244), it will be given F. * WoSign/StartCom certificates distrusted, will get 'T' grade.   **Changes in 2009p (1 March 2018 - 6 September 2018)**   * If vulnerable to the Return Of Bleichenbacher's Oracle Threat (ROBOT), it will be given F. * Cap to B if Forward Secrecy is not supported. * Cap to B if Authenticated encryption (AEAD) ciphers not supported. * Symantec certificates issued before June 2016 are distrusted, will get 'T' grade. * All remaining old Symantec PKI certificates will get 'T' grade   **Changes in 2009q (31 January 2020)**   * If vulnerable to the Zombie POODLE, it will be given F. * If vulnerable to the GOLDENDOODLE, it will be given F. * If vulnerable to the Zero Length Padding Oracle (CVE-2019-1559), it will be given F. * If vulnerable to the Sleeping POODLE, it will be given F. * If the server supports TLS 1.0 or TLS 1.1 then grade will be capped to B.   **About SSL**  The Secure Sockets Layer (SSL) protocol is a standard for encrypted  network communication. It was conceived at Netscape in 1994; version 2.0 was the first public release. SSL was later upgraded to 3.0, and, with  further minor improvements, standardized under the name TLS (Transport  Layer Security). TLS v1.2, the most recent version, is defined by RFC  5246.  **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 of  thousands of assessments have been performed using the free online  assessment tool. Other projects run by SSL Labs include periodic  Internet-wide surveys of SSL configuration and SSL Pulse, a monthly scan of about 170,000 most popular SSL-enabled web sites in the world.  **About Qualys**  Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of  cloud-based security and compliance solutions with over 9,300 customers  in more than 100 countries, including a majority of each of the Forbes  Global 100 and Fortune 100. The Qualys Cloud Platform and integrated  suite of solutions help organizations simplify security operations and  lower the cost of compliance by delivering critical security  intelligence on demand and automating the full spectrum of auditing,  compliance and protection for IT systems and web applications. Founded  in 1999, Qualys has established strategic partnerships with leading  managed service providers and consulting organizations including  Accenture, BT, Cognizant Technology Solutions, Deutsche Telekom,  Fujitsu, HCL, HP Enterprise, IBM, Infosys, NTT, Optiv, SecureWorks, Tata Communications, Verizon and Wipro. The company is also a founding  member of the Cloud Security Alliance (CSA). For more information,  please visit [www.qualys.com](http://www.qualys.com/).  Qualys, the Qualys logo and QualysGuard are proprietary trademarks of Qualys, Inc. All other products or names may be trademarks of their  respective companies.      **Pages 23**           * [Home](https://github.com/ssllabs/research/wiki)       * [Assessment Tools](https://github.com/ssllabs/research/wiki/Assessment-Tools)       * [Attack Tools](https://github.com/ssllabs/research/wiki/Attack-Tools)       * [BEAST](https://github.com/ssllabs/research/wiki/BEAST)       * [Configuration tools](https://github.com/ssllabs/research/wiki/Configuration-tools)       * [CRIME](https://github.com/ssllabs/research/wiki/CRIME)       * [Extended Validation Certificates](https://github.com/ssllabs/research/wiki/Extended-Validation-Certificates)       * [FIPS Requirements](https://github.com/ssllabs/research/wiki/FIPS-Requirements)       * [Forward Secrecy](https://github.com/ssllabs/research/wiki/Forward-Secrecy)       * [Interoperability Test Servers](https://github.com/ssllabs/research/wiki/Interoperability-Test-Servers)       * [Long Handshake Intolerance](https://github.com/ssllabs/research/wiki/Long-Handshake-Intolerance)       * [Lucky 13](https://github.com/ssllabs/research/wiki/Lucky-13)       * [Manual Testing](https://github.com/ssllabs/research/wiki/Manual-Testing)       * [Passive SSL Client Fingerprinting](https://github.com/ssllabs/research/wiki/Passive-SSL-Client-Fingerprinting)       * [PCI DSS Requirements](https://github.com/ssllabs/research/wiki/PCI-DSS-Requirements)         Clone this wiki locally |