The following paper was originally published in the
Proceedings of the Second USENIX Workshop on Electronic Commerce
Oakland, California, November 1996

Analysis of the SSL 3.0 Protocol

David Wagner, University of California, Berkeley
Bruce Schneier, Counterpane Systems

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Analysis of the SSL 3.0 protocol

David Wagner
University of California, Berkeley
daw@cs.berkeley.edu

Bruce Schneier
Counterpane Systems
schneier@counterpane.com

Abstract

The SSL protocol is intended to provide a practical, application-layer, widely applicable connection-oriented mechanism for Internet client/server communications security. This note gives a detailed technical analysis of the cryptographic strength of the SSL 3.0 protocol. A number of minor flaws in the protocol and several new active attacks on SSL are presented; however, these can be easily corrected without overhauling the basic structure of the protocol. We conclude that, while there are still a few technical wrinkles to iron out, on the whole SSL 3.0 is a valuable contribution towards practical communications security.

1 Introduction

The recent explosive growth of the Internet and the World Wide Web has brought with it a need to securely protect sensitive communications sent over this open network. The SSL 2.0 protocol has become a de facto standard for cryptographic protection of Web `http` traffic. But SSL 2.0 has several limitations—both in cryptographic security and in functionality—so the protocol has been upgraded, with significant enhancements, to SSL 3.0. This new version of SSL will soon see widespread deployment. The IETF Transport Layer Security working group is also using SSL 3.0 as a base for their standards efforts. In short, SSL 3.0 aims to provide Internet client/server applications with a practical, widely-applicable connection-oriented communications security mechanism.

This note analyzes the SSL 3.0 specification [FKK96], with a strong focus on its cryptographic security. We assume familiarity with the SSL 3.0 specification. Explanations of some of the cryptographic concepts can be found in [Sch96].

The paper is organized as follows. Section 2 briefly gives some background on SSL 3.0 and its predecessor SSL 2.0. Sections 3 and 4 explore several possible attacks on the SSL protocol and offer some technical discussion on the cryptographic protection afforded by SSL 3.0; this material is divided into two parts, with the SSL record layer analyzed in Section 3 and the SSL key-exchange protocol considered in Section 4. Finally, Section 5 concludes with a high-level view of the SSL protocol's strengths and weaknesses.

2 Background

SSL is divided into two layers, with each layer using services provided by a lower layer and providing functionality to higher layers. The SSL record layer provides confidentiality, authenticity, and replay protection over a connection-oriented reliable transport protocol such as TCP. Layered above the record layer is the SSL handshake protocol, a key-exchange protocol which initializes and synchronizes cryptographic state at the two endpoints. After the key-exchange protocol completes, sensitive application data can be sent via the SSL record layer.

SSL 2.0 had many security weaknesses which SSL 3.0 aims to fix. We briefly describe a short list of the flaws in SSL 2.0 which we have noticed. In export-weakened modes, SSL 2.0 unnecessarily weakens the authentication keys to 40 bits. SSL 2.0 uses a weak MAC construction, although post-encryption seems to stop attacks. SSL 2.0 feeds padding bytes into the MAC in block cipher modes, but leaves the padding-length field unauthenticated, which may potentially allow active attackers to delete bytes from the end of messages. There is a ciphersuite rollback attack, where an active attacker edits the list of ciphersuite preferences in the hello messages to invisibly force both endpoints to use a weaker form of encryption than they otherwise would choose; this serious flaw limits SSL 2.0's strength to "least common denominator" security when active attacks are a threat. Oth-

ers have also discovered some of these weaknesses: Dan Simon independently pointed out the ciphersuite rollback attack, Paul Kocher has addressed these concerns [Koc96], and the PCT 1.0 protocol [PCT95] discussed and countered some (though not all) of these flaws.

3 The record layer

This section considers the cryptographic strength of the record layer protocol, and assumes that the key-exchange protocol has securely set up session state, keys, and security parameters. Of course, a secure key-exchange protocol is vital to the security of application data, but an examination of attacks on the SSL key-exchange protocol is postponed until the next section.

The SSL record layer addresses fairly standard problems that have received much attention in the cryptographic and security literature [KV83], so it is reasonable to hope that SSL 3.0 provides fairly solid protection in this respect. As we shall see, this is not far from the truth. We consider confidentiality and integrity protection in turn.

3.1 Confidentiality: eavesdropping

The SSL protocol encrypts all application-layer data with a cipher and short-term session key negotiated by the handshake protocol. A wide variety of strong algorithms used in standard modes is available to suit local preferences; reasonable applications should be able to find an encryption algorithm meeting the required level of security, US export laws permitting. Key-management is handled well: short-term session keys are generated by hashing random per-connection salts and a strong shared secret. Independent keys are used for each direction of a connection as well as for each different instance of a connection. SSL will provide a lot of known plaintext to the eavesdropper, but there seems to be no better alternative; since the encryption algorithm is required to be strong against known-plaintext attacks anyway, this should not be problematic.

3.2 Confidentiality: traffic analysis

When the standard attacks fail, a cryptanalyst will turn to more obscure ones. Though often maligned, traffic analysis is another passive attack worth considering. Traffic analysis aims to recover confidential information about protection sessions by examining unencrypted packet fields and unprotected packet attributes. For example, by examining the unencrypted IP source and destination addresses (and even TCP ports), or examining the volume of network traffic flow, a traffic analyst can determine what parties are interacting, what type of services are in use, and even sometimes recover information about business or personal relationships. In practice, users typically consider the threat of this kind of coarse-grained tracking to be relatively harmless, so SSL does not attempt to stop this kind of traffic analysis. Ignoring coarse-grained traffic analysis seems like a reasonable design decision.

However, there are some more subtle threats posed by traffic analysis in the SSL architecture. Bennet Yee has noted that examination of ciphertext lengths can reveal information about URL requests in SSL- or SSL-encrypted Web traffic [Yee96]. When a Web browser connects to a Web server via an encrypted transport such as SSL, the GET request containing the URL is transmitted in encrypted form. Exactly which Web page was downloaded by the browser is clearly considered confidential information—and for good reason, as knowledge of the URL is often enough for an adversary to obtain the entire Web page downloaded—yet traffic analysis can recover the identity of the Web server, the length of the URL requested, and the length of the `html` data returned by the Web server. This leak could often allow an eavesdropper to discover what Web page was accessed. (Note that Web search engine technology is certainly advanced enough to catalogue the data openly available on a Web server and find all URLs of a given length on a given server which return a given amount of `html` data.)

This vulnerability is present because the ciphertext length reveals the plaintext length.[1] SSL includes support for random padding for the block cipher modes, but not for the stream cipher modes. We believe that SSL should at the minimum support the usage of random-length padding for all cipher modes, and should also strongly consider requiring it for certain applications.

[1] This is strictly speaking only true of stream ciphers, but they are currently the common case. With block ciphers, plaintexts are padded out to the next 8-byte boundary, so one can only recover a close estimate of the plaintext length.

3.3 Confidentiality: active attacks

It is important that SSL securely protect confidential data even against active attacks. Of course, the underlying encryption algorithm should be secure against adaptive chosen-plaintext/chosen-ciphertext attacks, but this is not enough on its own. Recent research motivated by the IETF ipsec (IP security) working group has revealed that sophisticated active attacks on a record layer can breach a system's confidentiality even when the underlying cipher is strong [Bel96]. It appears that the SSL 3.0 record layer resists these powerful attacks; it is worth discussing in some depth why they are foiled.

One important active attack on ipsec is Bellovin's cut-and-paste attack [Bel96]. Recall that, to achieve confidentiality, link encryption is not enough—the receiving endpoint must also guard the sensitive data from inadvertent disclosure. The cut-and-paste attack exploits the principle that most endpoint applications will treat inbound encrypted data differently depending on the context, protecting it more assiduously when it appears in some forms than in others.[2] The cut-and-paste attack also takes advantage of a basic property of the cipher-block chaining mode: it recovers from errors within one block, so transplanting a few consecutive ciphertext blocks between locations within a ciphertext stream results in a corresponding transfer of plaintext blocks, except for a one-block error at the beginning of the splice. In more detail, Bellovin's cut-and-paste attack cuts an encrypted ciphertext from some packet containing sensitive data, and splices it into the ciphertext of another packet which is carefully chosen so that the receiving endpoint will be likely to inadvertently leak its plaintext after decryption. For example, if cut-and-paste attacks on the SSL record layer were feasible, they could be used to compromise site security: a cut-and-paste attack on a SSL server-to-client Web page transfer could splice ciphertext from a sensitive part of that `html` transfer into the hostname portion of a URL included elsewhere in the transferred Web page, so that when a user clicks on the booby-trapped URL link his browser would interpret the decryption of the spliced sensitive ciphertext as a hostname and send a DNS domain name lookup for it in the clear, ready for capture by the eavesdropping attacker. Cut-and-paste attacks, in short, enlist the unsuspecting receiver to decrypt and inadvertently leak sensitive data for them.

SSL 3.0 stops cut-and-paste attacks. One partial defense against cut-and-paste attacks is to use independent session keys for each different context. This prevents cutting and pasting between different connections, different directions of a connection, etc. SSL already uses independent keys for each direction of each incarnation of each connection. Still, cutting and pasting within one direction of a transfer is not prevented by this mechanism. The most comprehensive defense against cut-and-paste attacks is to use strong authentication on all encrypted packets to prevent enemy modification of the ciphertext data. The SSL record layer does employ this defense, so cut-and-paste attacks are completely foiled. For a more complete exposition on cut-and-paste attacks, see Bellovin's paper [Bel96].

The short-block attack is another active attack against ipsec which can be found in Bellovin's paper [Bel96]. The short-block attack was originally applied against DES-CBC ipsec-protected TCP data when the final message block contains a short one-byte plaintext and the remainder of it is filled by random padding. One guesses at the unknown plaintext byte by replacing the final ciphertext block with another ciphertext block from a known plaintext/ciphertext pair. Correct guesses can be recognized by the validity of the TCP checksum: an incorrect guess will cause the packet to be silently dropped by the receiver's TCP stack, but the correct guess will cause a recognizable ACK to be returned. Knowledge of the corresponding plaintext for a correctly guessed replacement ciphertext block enables the enemy to recover the unknown plaintext byte. Because the receiving ipsec stack ignores the padding bytes, the short-block attack requires about 2^8 known plaintexts and 2^8 active online trials to recover such an unknown trailing byte. Many distracting technicalities have been significantly simplified; see Bellovin's paper [Bel96] for more details.

There are no obvious short-block attacks on SSL. The SSL record layer format is rather similar to the old vulnerable ipsec layout, so it is admittedly conceivable that a modified version of the attack might work against SSL. In any case, standard SSL-encrypting Web servers probably would not be threatened by a short-block type of attack, since they do not typically encrypt short blocks. (Note, however, that a SSL-encrypting `telnet` client should

[2] In the ipsec world, encrypted data to TCP user ports is not protected by the operating system nearly as strongly as encrypted data to the system TCP login or telnet port. For a SSL-protected Web connection, the client browser will guard the path portion of a URL more carefully than the hostname portion, as the hostname portion may subsequently appear unencrypted in DNS queries and IP source addresses, whereas the path portion of a URL is encrypted via SSL.

demand particularly robust protection against short-block attacks, as each keystroke is typically sent in its own one-byte-long packet.)

In summary, our analysis did not uncover any active attacks on the confidentiality protection of the SSL 3.0 record layer.

3.4 Message authentication

In addition to protecting the confidentiality of application data, SSL cryptographically authenticates sensitive communications. On the Internet, active attacks are getting easier to launch every day. We are aware of at least two commercially available software packages to implement active attacks such as IP spoofing and TCP session hijacking, and they even sport a user-friendly graphical interface. Moreover, the financial incentive for exploiting communications security vulnerabilities is growing rapidly. This calls for strong message authentication.

SSL protects the integrity of application data by using a cryptographic MAC. The SSL designers have chosen to use HMAC, a simple, fast hash-based construction with some strong theoretical evidence for its security [BCK96]. In an area where several initial ad-hoc proposals for MACs have been cryptanalyzed, these provable security results are very attractive. HMAC is rapidly becoming the gold standard of message authentication, and it is an excellent choice for SSL. Barring major unexpected cryptanalytic advances, it seems unlikely that HMAC will be broken in the near future.

We point out that SSL 3.0 uses an older obsolete version of the HMAC construction. SSL should move to the updated current HMAC format when convenient, for maximal security.

On the whole, SSL 3.0 looks very secure against straightforward exhaustive or cryptanalytic attacks on the MAC. SSL 2.0 had a serious design flaw in that it used an insecure MAC—though post-encryption saved this from being a direct vulnerability—but SSL 3.0 has fixed this mistake. The SSL MAC keys contain at least 128 bits of entropy, even in export-weakened modes, which should provide excellent security for both export-weakened and domestic-grade implementations. Independent keys are used for each direction of each connection and for each new incarnation of an connection. The choice of HMAC should stop cryptanalytic attacks. SSL does not provide non-repudiation services, and it seems reasonable to deliberately leave that to special higher-level application-layer protocols.

3.5 Replay attacks

The naive use of a MAC does not necessarily stop an adversary from replaying stale packets. Replay attacks are a legitimate concern, and as they are so easy to protect against, it would be irresponsible to fail to address these threats. SSL protects against replay attacks by including an implicit sequence number in the MACed data. This mechanism also protects against delayed, re-ordered, or deleted data. Sequence numbers are 64 bits long, so wrapping should not be a problem. Sequence numbers are maintained separately for each direction of each connection, and are refreshed upon each new key-exchange, so there are no obvious vulnerabilities.

3.6 The Horton principle

Let's recall the ultimate goal of message authentication. SSL provides message integrity protection just when the data passed up from the receiver's SSL record layer to the protected application exactly matches the data uttered by the sender's protected application to the sender's SSL record layer. This means, approximately, that it is not enough to apply a secure MAC to just application data as it is transmitted over the wire—one must also authenticate any context that the SSL mechanism depends upon to interpret inbound network data. For lack of a better word, let's call this "the Horton principle" (with apologies to Dr. Seuss) of semantic authentication: roughly speaking we want SSL to

> "authenticate what was meant, not what was said."

To phrase it another way,

> Eschew unauthenticated security-critical context.

SSL 2.0 suffered from at least one flaw along these lines: it included padding data but not the length of the padding in the MAC input, so an active attacker could manipulate the cleartext padding length field to compromise message integrity. An analysis checking SSL 2.0's compliance with the Horton principle would have uncovered this flaw; therefore, we undertake an informal analysis of SSL 3.0 following the guidelines of the Horton principle.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.