Автоматизация тестирования соответствия реализаций стандарту протокола безопасности транспортного уровня TLS

Алексей Никешин, Николай Пакулин, Виктор Шнитман

Институт системного программирования РАН,
Москва, Россия
{alexn,npak,vzs}@ispras.ru
http://www.ispras.ru/

Аннотация Протокол TLS используется для защиты обменов между клиентом и сервером в различных прикладных сценариях: при обмене данными между браузером и веб-сервером, передаче электронной почты, создании виртуальных сетей, голосовой и видеосвязи и т.п. На настоящий момент широко используются более десятка различных реализаций TLS. Обеспечение совместимости реализаций TLS является очень актуальной задачей: несовместимость двух реализаций может привести к различным последствиям, начиная с невозможности установить соединение вплоть до разглашения конфиденциальных данных.
В данной работе мы представляем подход к тестированию соответствия стандарту TLS, основанный на автоматизированном тестировании соответствия формальным спецификациям. Кратко представлены результаты тестирования нескольких общедоступных реализаций TLS. Обсуждаются направления дальнейших исследований.

Keywords: тестирование, верификация, формальные методы, формальные спецификации, Model Based Testing, тестирование с использованием моделей, модели программ, TLS, SSL, UniTESK

1 Введение

Среди всех современных протоколов, предназначенных для защиты передачи информации в открытых сетях, наиболее широко используется протокол Transport Layer Security (TLS). Протокол TLS применяется для защиты обменов между клиентом и сервером в различных прикладных сценариях:

- для защиты WWW-трафика (протокол HTTPS),
- отправка и получение электронной почты (с почтовыми протоколами SMTP, IMAP, POP3),
- при организации виртуальных защищенных сетей (OpenVPN),
- для защиты голосовой и видеосвязи (с SIP и основанными на нем приложениями, например VoIP),

– и других.

Разработчики протокола TLS поставили перед собой следующие цели:

1. Криптографическая безопасность: TLS используется для установления безопасного соединения между двумя участниками.
2. Совместимость: взаимодействие приложений по протоколу TLS не зависит от внутренних особенностей реализаций.
3. Расширяемость: возможность расширения функциональности протокола, в том числе добавления новых алгоритмов шифрования и цифровой подписи.
4. Эффективность криптографических операций.

На настоящий момент широко используются более десятка различных реализаций TLS. По этой причине обеспечение совместимости реализаций TLS является актуальной задачей. Несовместимость двух реализаций может привести к различным последствиям, начиная с невозможности установить соединение до разглашения конфиденциальных данных.

Наивный подход к решению этой задачи заключается в том, чтобы каждую реализацию испытать на возможность взаимодействия с каждой. Очевидны недостатки этого подхода. Необходимо собрать все реализации TLS и провести попарные сценарии обменов данными. Так как для протоколов безопасности сценарии отказов не менее важны, чем сценарии нормальной передачи данных, необходимо разработать большое количество сценариев для каждой пары, причем может оказаться, что отдельные сценарии невозможно реализовать для тех или иных пар. Кроме того, такие испытания плохо поддаются автоматизации, многие сценарии приходится выполнять вручную.

В данной работе используется подход к оценке совместимости реализаций протокола TLS, основанный на следующей гипотезе: протокол сконструирован таким образом, что если две реализации соответствуют спецификации протокола, то они совместимы. Указанная гипотеза принимается как постулат и в рамках данной работы не проверяется.

Для оценки соответствия реализации спецификации протокола в данной работе используется тестирование. Несмотря на популярность протокола TLS и ценность информации, которая защищается этим протоколом, открытых тестовых наборов для тестирования соответствия TLS нет. Существуют тестовые наборы, разработанные для отдельных реализаций, но они направлены прежде всего на функциональное тестирование подсистем конкретной реализации, а не на проверку соответствия наблюдаемого поведения реализации TLS стандарту протокола.

В данной работе мы рассматриваем вопросы формализации требований к протоколу TLS и тестирования соответствия реализаций стандарту протокола. Также в статье представлены направления дальнейших работ по развитию тестового набора для протокола TLS.

2 Обзор протокола TLS

В середине 1990-х годов в компании Netscape велись разработки средств защиты передачи данных, которые обеспечивали бы аутентификацию и конфиденциальность, и одновременно предоставляли программный интерфейс, близкий к популярному интерфейсу сокетов. Получившийся протокол получил название Secure Sockets Layer, SSL. Первая версия SSL никогда не была опубликована за пределами Netscape. Вторая версия протокола, SSL 2.0, была опубликована в 1995 году, но вследствие существенных уязвимостей в Netscape немедленно приступили к полной переделке протокола. Новая версия протокола получила название SSL 3.0 и была выпущена в 1996 году [1]. Протокол SSL 3.0 оказался настолько удачным, что в конце 1990-х годов его поддерживали все основные разработчики браузеров и веб-серверов, включая Microsoft и сообщество разработчиков открытых программ.

Спецификация SSL 3.0 стала основой для интернет-стандарта Transport Layer Security (TLS). Первая версия спецификации увидела свет в 1999 году [2]. Несмотря на то, что различия между протоколами невелики, TLS и SSL 3.0 несовместимы. Так как на момент публикации спецификации TLS 1.1 протокол SSL уже был широко распространен, в TLS предусмотрен механизм, позволяющий реализациям TLS общаться с реализациями как SSL 3.0, так и SSL 2.0. Однако, в силу того, что протокол SSL 2.0 уязвим атакам “грубой силы” (brute force), реализациям TLS запрещено использовать (MUST NOT) SSL 2.0 [5]. По мере развития вычислительных средств и появления новых атак протокол TLS дважды модернизировался: в 2004-м году была опубликована версия протокола TLS 1.1 [3], а в 2008-м году TLS 1.2 [4]. В настоящее время наиболее распространенными версиями протоколов SSL и TLS являются SSL 3.0 и TLS 1.0 : более 99% веб-сайтов поддерживают их [6]. 13.4% сайтов поддерживают TLS 1.1 и 15.8% – TLS 1.2.

Протокол TLS состоит из двух уровней: TLS Record Protocol (протокол записей) и служебных протоколов.

На нижнем уровне находится протокол TLS Record, работающий поверх транспортного протокола TCP. Протокол TLS Record обеспечивает конфиденциальность и надежность соединений. Для защиты данных используются алгоритмы симметричного шифрования, ключи для которых уникальны для каждой сессии и создаются на основе секрета, согласованного другими протоколами (например, протоколом Handshake). Необходимо отметить, что протокол TLS Record также может использоваться без шифрования. Целостность сообщений обеспечивается путем вычисления и включения в сообщения кода аутентификации (MAC). Для этого используются криптографические алгоритмы на основе хэш-функций (такие как SHA-1, MD5 и др.). Данный протокол может использоваться без вычисления MAC.

Протокол TLS Record используется для инкапсуляции протоколов TLS более высокого уровня: handshake, оповещений, смены состояния, передачи данных. При отправке TLS Record фрагментирует и сжимает сообщения служебных протоколов, вычисляет код аутентификации и шифрует. При получении операции осуществляются в обратном порядке.

Протокол Handshake предназначен для взаимной аутентификации клиента и сервера и согласования параметров криптографической защиты (выбор конкретных криптографических алгоритмов и ключей) перед началом передачи данных. Для аутентификации используются криптографические алгоритмы с открытым ключом (такие как RSA, DSS и др.). Протокол Handshake также может использоваться без аутентификации. Однако обычно она необходима, по крайней мере, для сервера.

Одно из преимуществ TLS состоит в том, что он независим от протоколов прикладного уровня. Для протоколов более высокого уровня использование TLS является прозрачным, однако спецификация TLS не определяет схему их взаимодействия. Решение о том, как инициировать TLS-диалог и как интерпретировать сертификаты аутентификации, оставляется на усмотрение разработчиков протоколов, которые работают поверх TLS.

Важнейшей составляющей TLS является состояние соединения TLS. В нем содержится набор параметров для работы протокола TLS Records: криптографические алгоритмы сжатия, шифрования и вычисления MAC, а также соответствующие ключи. Существуют четыре состояния соединения: текущие состояния чтения и записи и следующие (pending) состояния чтения и записи. Для любой обработки данных используются установки текущего состояния. Параметры следующих состояний устанавливаются через протокол Handshake. Протокол Изменения состояния (Change Cipher Spec) заменяет текущее состояние следующим, а следующее состояние инициализируется пустым состоянием. Начальное текущее состояние всегда определяется без использования шифрования, сжатия и вычисления MAC.

Реализации TLS используют протокол оповещений (Alerts) для сигнализации ошибочных ситуаций. Оповещения делятся на предупреждающие и критические. Критическое сообщение приводит к немедленному закрытию соединения. При этом другие соединения данной сессии могут быть продолжены, но новые соединения в рамках данной сессии создавать запрещено. Сообщения оповещения зашифрованы и сжаты, как определено в текущем состоянии соединения.

Протокол TLS допускает расширения для добавления новой функциональности. Для согласования расширений используются сообщения *ClientHello* и *ServerHello* протокола Handshake. Каждое расширение содержит тип расширения и данные, формат которых зависит от конкретного расширения. Первые расширения (как и сами механизмы расширений) описаны в спецификации RFC 4366.

3 Тестирование TLS

В настоящее время существуют более десяти реализаций протокола TLS. Безусловно, все они тестировалась в процессе разработки. Однако, как правило, тестирование реализаций разработчиками продуктов имеет следующие отличительные черты:

– реализации тестируются по принципу “белого ящика”, при этом используются внутренние особенности реализации;
– тестовые наборы для проприетарных реализаций недоступны для использования вне компании-разработчика;
– тестирование нацелено на выявление ошибок в “недрах” реализации, соответственно, тестовые наборы не содержат специально выделенного подмножества тестов соответствия стандарту.

Тестовый набор для самой популярной свободной реализации SSL/TLS – проект OpenSSL – содержит большой набор тестов для проверки различных аспектов поведения библиотеки openssl. Однако, эти тесты непереносимы на другие реализации, так как существенно используют особенности реализации OpenSSL, и, кроме того, не предназначены для тестирования соответствия стандарту.

С некоторой натяжкой можно сказать, что в OpenSSL частично реализовано тестирование совместимости. А именно, есть возможность запустить приложение, реализующее клиентскую сторону TLS, в режиме тестирования. При запуске необходимо указать адрес и порт, на которых находится тестируемая реализация сервера. Результатом выполнения тестов будет вердикт о совместимости целевой реализации с OpenSSL.

Необходимо отметить, что при этом нет возможности как-либо соотнести результат работы тестов со стандартом. Результат говорит только о совместимости с OpenSSL, но ничего не говорит о том, какие пункты стандарта проверялись. При этом нельзя считать OpenSSL эталоном, т.к. не исследовалось, как данная реализация сама по себе соотносится со стандартом.

Тестовый набор для реализации TLS в проекте открытой реализации языка Java OpenJDK насчитывает около 140 классов. Из них для тестирования соответствия стандарту предназначены только 4, остальные проверяют различные компоненты реализации TLS в Java Secure Socket Extension.

В работе [7] описана верификация самого протокола TLS на исполнимой модели. Модель разрабатывалась на функциональном языке F# и представляет собой упрощенную реализацию TLS, которая, тем не менее, была способна взаимодействовать с полноценными (mainstream) реализациями TLS. Разработанная исполнимая модель была верифицирована на соответствие требованиям безопасности, другими словами, верифицировался сам протокол TLS. Авторы упомянули, что идет разработка соответствующего тестирующего программного обеспечения для протокола TLS на базе разработанной модели. Но подробности не приведены, других работ этих авторов на тему тестирования реализаций TLS обнаружить не удалось.

Как уже говорилось во введении, тестирование соответствия стандарту является одним из основных механизмов обеспечения совместимости между реализациями протоколов. Фактически, TLS является единственным из всех основных протоколов стека TCP/IP, для которого нет тестового набора для тестирования соответствия стандарту.

Практическая значимость разработки такого тестового набора несомненна. С другой стороны, разработка тестового набора традиционным образом,

как набора отдельных тестовых программ, каждая из которых проверяет определенное требование, является трудоемкой задачей. Прежде всего, тесты должны проверять большое количество вариантов поведения протокола TLS в зависимости от выбора параметров TLS соединения – криптографических алгоритмов, алгоритма сжатия, схемы построения общего секрета.

В данной работе представлен подход к автоматизации разработки такого тестового набора. Подход основан на методе тестирования с использованием моделей.

3.1 Используемый подход

В данной работе используется подход к автоматизации тестирования, основанный на технологии автоматизированного тестирования UniTESK [8].

Технология UniTESK предоставляет средства автоматизации тестирования соответствия формальным спецификациям. Требования, представленные в тексте стандарта, изложены на английском языке и представляют собой неформальный текст, описывающий желаемое поведение системы на естественном языке. Для того, чтобы автоматизировано построить тесты для протокола, необходимо перевести его спецификацию в вид, пригодный для решений этой задачи. В данной работе в этой роли выступают формальные спецификации, в которых требования задаются как модель – набор булевских предикатов и императивных конструкций, заданных средствами исполнимого формализма. Для упрощения разработки и анализа модели в данной работе в качестве формализма использовалось расширение языка Java[9]. Указанный подход уже применялся авторами в аналогичных проектах [10,11,12,13].

В UniTESK тест представляет собой конечный автомат. С каждым переходом автомата сопоставлено определённое тестовое воздействие. При выполнении перехода это воздействие подаётся на тестируемую реализацию, регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. Вердикт выносится на основании модели, которая проверяет, допустима ли зарегистрированная реакция в текущем состоянии тестируемой реализации. Так как тестирование проводится по схеме “черного ящика”, то есть внутреннее состояние реализации недоступно тесту, то модель также обновляет модельное состояние в соответствии с требованиями стандарта. Тем самым модель выступает как эталонная реализация протокола.

Спецификация автомата теста на конкретном формализме называется тестовым сценарием. В данной работе для записи тестовых сценариев используется то же расширение языка Java[9], что и для разработки моделей. Тестовые воздействия, реализующие дуги в автомате теста, представляют собой методы на языке Java, в которых тестовые воздействия оказываются на объект модели протокола. Эти методы разрабатываются вручную, но обход осуществляется в полностью автоматическом режиме. Алгоритм обхода не зависит от протокола, тестируемой реализации или конкретного теста; он

реализован как библиотечный компонент UniTESK, разработчикам тестов не требуется его как либо адаптировать под целевой протокол.

Последовательность переходов автомата теста определяет тестовую последовательность. В силу различий в настройках конкретных реализаций и различий в поддержке необязательных функций в реализациях тестовые последовательности, получаемые при прогоне тестов на разных реализациях, могут не совпадать друг с другом.

Построение тестовых воздействий и верификация полученных ответов реализации опирается на модель протокола. Спецификация протокола не требует, чтобы реализации протокола использовали в точности те структуры данных, которые введены в стандарте для описания протокола. Требуется только соответствие внешне наблюдаемого поведения требованиям. Однако в модели мы не гонимся за производительностью, поэтому набор внутренних переменных модели состоит из тех же самых переменных, что указаны в стандарте: параметры сессии TLS и по четыре состояния чтения-записи на сессию.

Помимо состояний в модель протокола входят спецификационные методы – исполнимые модели стимулов и реакций TLS сервера. Каждый спецификационный метод содержит предусловие, в котором проверяется правильность структуры тестового сообщения и его своевременность, и на основании этого делается вывод о том, должен ли на него быть ответ, сообщение об ошибке или реализация должна его проигнорировать.

Параметрами спецификационных методов служат сообщения TLS в модельном представлении, то есть объекты Java. Наборы полей соответствующих классов следуют структуре сообщений, описанных в стандарте TLS. В состав тестового набора входят кодеки (codecs) для преобразования сообщений из объектов в байтовые массивы для отправки в целевую систему, и для обратного преобразования полученных сообщений.

В отличие от большинства телекоммуникационных протоколов, кодеки для TLS не являются контекстно-независимыми. Для корректного кодирования и декодирования сообщений кодек должен иметь доступ к текущему состоянию приема и отправки, прежде всего к ключам шифрования и криптографической контрольной суммы.

Другая особенность кодеков для TLS отражает тот факт, что протокол TLS двухуровневый. Сообщения служебных протоколов (например, TLS Handshake) и данные прикладных протоколов преобразуются в двоичный вид и фрагментируются, а полученные фрагменты обрабатываются протоколом TLS Records – упаковываются и шифруются. Фактически, именно в кодеках реализуется модель протокола TLS Records и его взаимодействие со служебными протоколами.

3.2 Тестовый набор

Тестовый стенд состоит из двух сетевых узлов: инструментального и целевого. На целевом узле функционирует тестируемая реализация. На инструментальном узле исполняется обход тестового автомата и верификация

наблюдаемых реакций. Инструментальный и целевой узлы могут располагаться в разных сегментах сети.

Стимулами в разработанном тестовом наборе являются сообщения от инструментального узла, а реакциями - сообщения со стороны тестируемого узла. Основная часть требований спецификации TLS проверяется в модели протокола. Часть требований проверяется в декодерах сообщений – прежде всего, требования к структуре сообщений и форматам данных.

Процесс тестирования начинается с создания TCP соединения к целевому узлу и инициализации начальных значение переменных модели. Все обмены сообщениями происходят по созданному соединению. Из модельного представления тестового сообщения TLS строится реализационное, которое и отправляется в сеть. Сообщения TLS, полученные из установленного соединения, декодируются, и строятся модельные представления этих сообщений. Последовательность блоков данных в полученных сообщениях рассматривается как последовательность реакций целевой системы.

В модели полученные сообщения проверяются на соответствие требованиям спецификации. Проверка разделена на несколько стадий. Сначала проверяется допустимость такого сообщения от реализации и его своевременность, затем - структура самого сообщения (присутствующие поля и их значения должны соответствовать текущему обмену). После проверки всех требований, результат передается тестовому сценарию, где в зависимости от плана сценария, принимается решение о продолжении или завершении информационного обмена. В случае выявления нарушения требований принимается решение о критичности ошибки и возможности отправки следующих запросов.

Первыми по установленному TCP соединению передаются сообщения протокола TLS Handshake. В случае успешного завершения обменов по этому протоколу в модели создается новая TLS-сессия и устанавливаются параметры следующих состояний чтения-записи (pending read-write states). Кроме того, модель поддерживает сокращенный вариант обмена, в котором используется уже существующая TLS сессия.

При создании TLS сессии клиент и сервер согласовывают различные параметры: применяемые алгоритмы сжатия, шифрования и цифровой подписи, схемы генерации разделяемых секретов, методы аутентификации. Для TLS Handshake разработаны несколько тестовых сценариев, в которых перебираются различные сочетания криптографических алгоритмов и соответствующих им схем построения секретов. Тест для Handshake разделен на несколько автоматов в силу особенности алгоритма обхода в UniTESK – длина тестовой последовательности (а значит, и время выполнения теста) быстро растут по мере увеличения числа вариантов тестовых воздействий (дуг в автомате). Для получения приемлемого времени работы теста иногда имеет смысл разделить тестовый автомат на несколько, которые в совокупности дают то же покрытие требований, что и “большой” тест, но исполняются за меньшее время.

Протокол TLS Handshake содержит львиную долю функциональности протокола TLS. Функции остальных служебных протоколов гораздо проще, поэтому для них не требуется большое количество сценариев. Протокол смены состояния и протокол передачи прикладных данных тестируются в одном сценарии, так как невозможно передавать данные через соединение TLS, не осуществив смену состояния. Протокол оповещений покрыт в текущей реализации тестового набора не полностью. Часть из ошибочных ситуаций, предусмотренных стандартом протокола, не тестируются.

3.3 Апробация

Тестовый набор был апробировн на примере тестирования нескольких открытых серверных реализаций TLS. В роли TLS-сервера реализация не генерирует запросы, а лишь поддерживает информационный обмен, инициированный другим узлом. Стимулами являются тестовые сообщения от инструментального узла.

Для тестирования на соответствие стандарту были выбраны следующие реализации:

- Почтовый сервер Postfix 2.9.3 с открытой реализацией протокола TLS openssl.1.0.1c под управлением операционной системы Red Hat Enterprise Linux 5.5;
- Реализация TLS в виртуальной машине Java 1.7.0_05 (Java Secure Socket Extension, JSSE);
- Тестовый сервер TLS интернет ресурса https://www.mikestoolbox.net.

В первой реализации были выявлены 8 отклонений от спецификации TLS. В реализации JSSE выявлены 5 отклонений от спецификации. В третьем сервере обнаружены 3 отклонения от реализации. Несмотря на некоторые особенности и нарушения, реализации в целом соответствуют спецификации, за исключением JSSE. Эта реализация нарушает критичное требование проверки версии протокола TLS в сообщении $ClientKeyExchange$ с использованием алгоритма RSA, а также игнорирует пустое сообщение $Certificate$ (стандарт требует послать сообщение $Error$). Подробный разбор части результатов тестирования можно найти в [14].

4 Направления дальнейшего развития

Разработанный тестовый набор для протокола TLS покрывает существенную часть функциональных требований, изложенных в стандартах протокола. Однако для протоколов безопасности есть ещё одна важная задача – тестирование устойчивости. Протокол TLS используется для защиты общедоступных серверов, поэтому реализация TLS должна быть “готова” к приему произвольных сообщений.

Спецификация TLS определяет более 40 ошибочных ситуаций, однако их список не является исчерпывающим. Помимо тестирования на выделенные в спецификации особые случаи необходимо тестировать устойчивость к ошибкам в сообщениях, не указанным в спецификации протокола.

На практике тестирование устойчивости к некорректным сообщениям осуществляется преимущественно посредством случайного тестирования (fuzz testing, random testing). При случайном тестировании генерируются более-менее произвольные последовательности байтов в качестве целых пакетов или отдельных полей.

Построение случайных сообщений легко реализовать. Существует большое число исследовательских и коммерческих генераторов случайных сообщений, например IxDefend компании Ixia, Mu Dynamix одноименной компании, EMRET компании Microsoft, Dranzer организации CERT и т.д. Однако вероятность построения сообщения, которое способно преодолеть хотя бы часть механизмов валидации (шифрование и контрольные суммы TLS Records, сжатие, согласованные значения полей сообщений TLS Handshake) настолько мала, что случайное тестирование, дающее сколько-нибудь значимую долю покрытия кода реализации, займет неприемлемо длительное время.

Мы предполагаем дополнить имеющийся тестовый набор средствами мутационного тестирования (mutation testing), при котором ошибочные сообщения строятся как искажения корректных сообщений. При мутационном тестировании необходимо сначала построить корректное сообщение, а затем трансформировать («мутировать») его в некорректное. Такое преобразование сообщений называется оператором мутации. При мутационном тестировании число тестовых воздействий существенно меньше, чем при случайном тестировании, а покрытие реализации выше. Однако применение мутационного тестирования к TLS связано с нетривиальной задачей построения корректного сообщения, так как содержимое сообщения связано с внутренним состоянием протокола: сообщение должно быть корректно зашифровано и подписано, в противном случае оно будет отброшено на ранних этапах обработки.

Применение мутационного тестирования нацелено на выявление нарушений функционирования из-за переполнений буфера, разрушения стека, индексирования за пределами массива. Мы предполагаем использовать мутации совместно с тестированием на основе моделей: благодаря наличию модели можно целенаправленно привести реализацию в необходимое состояние и подготовить разумные данные для корректного сообщения. Кроме того, модель нужна для корректной работы кодека TSL Records.

5 Заключение

Тестирование соответствия спецификации или стандарту является в настоящее время основным механизмом обеспечения совместимости между различными реализациями: если две реализации корректно реализуют протокол, то

они обязательно смогут взаимодействовать. Такой подход используется для протоколов всех уровней в стеке TCP/IP – существуют тесты для канального уровня (Ethernet), сетевого (IPv4 и IPv6), транспортного (TCP и UDP), многих протоколов прикладного уровня (DNS, DHCP, SMTP/POP/IMAP и т.п.)

Однако в сфере TLS сложилась парадоксальная ситуация. Протокол TLS используется на сотнях тысяч серверов по всему миру, сотни миллионов людей пользуются им ежедневно для защиты соединений с веб-серверами, отправки и получения электронной почты, существуют более десятка реализаций, но при этом нет ни одного достаточно содержательного тестового набора, с помощью которого можно было бы удостовериться в соответствии реализации стандарту TLS.

В данной работе представлен подход к разработке тестового набора для тестирования соответствия реализации протокола TLS на соответствие стандарту. Подход использует технологию UniTESK к автоматизации тестирования: тестовая последовательность строится динамически как обход некоторого автомата теста, для построения тестовых воздействий и вынесения вердикта о корректности наблюдаемого поведения реализации используется модель протокола.

Был разработан тестовый набор; он покрывает большинство требований, изложенных в стандарте протокола TLS[4]. Тестовый набор апробирован на трех общедоступных реализациях TLS/SSL, во всех реализациях обнаружены отклонения от спецификации, в одном случае нарушение расценивается как критическое.

В качестве дальнейшего направления работ рассматривается расширение тестового набора тестами на некорректные сообщения. Такие тесты предполагается автоматизировать с использованием методов мутационного тестирования.

Список литературы

1. A. Freier, P. Karlton, P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. August 2011.
2. IETF RFC 2246. Dierks, T. and C. Allen, "The TLS Protocol Version 1.0 January 1999.
3. IETF RFC 4346. Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1 April 2006.
4. IETF RFC 5246. Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2 August 2008.
5. IETF RFC 6176. Turner S. and Polk, T. "Prohibiting Secure Sockets Layer (SSL) Version 2.0 March 2011.
6. Проект SSL Pulse консорциума Trustworthy Internet Movement, https://www.trustworthyinternet.org/ssl-pulse/
7. K. Bhargavan, C. Fournet, R. Corin, E. Zalinescu. Cryptographically verified implementations for TLS. Proceedings of the 15th ACM conference on Computer and communications security. ACM New York, NY, USA 2008. ISBN: 978-1-59593-810-7.

8. Bourdonov I., Kossatchev A., Kuliamin V., Petrenko A. UniTesK Test Suite Architecture. Proceedings of FME, LNCS 2391. Springer-Verlag, 2002. P. 77-88
9. Bourdonov I.B., Demakov A.V., Jarov A.A., Kossatchev A.S., Kuliamin V.V., Petrenko A.K. and Zelenov S.V. Java Specification Extension for Automated Test Development. Proceedings of PSI-2001. Novosibirsk, Russia July 2-6 2001, LNCS 2244:301-307. Springer-Verlag, 2001.
10. Н.В. Пакулин. Формализация стандартов и тестовых наборов протоколов Интернета. Автореферат диссертации на соискание учёной степени кандидата физико-математических наук. Москва, 2006.
11. Н.В. Пакулин, А.В. Хорошилов "Разработка формальных моделей и тестирование соответствия для систем с асинхронными интерфейсами и телекоммуникационных протоколов Журнал "Программирование"№ 5, 2007 г., ISSN 0132-3474, с. 1-29.
12. А.В. Никешин, Н.В. Пакулин, В.З. Шнитман "Разработка тестового набора для верификации реализаций протокола безопасности IPsec v2 Труды Института системного программирования РАН, т. 18, 2010, стр. 151-182.
13. А.В. Никешин, Н.В. Пакулин, В.З. Шнитман "Верификация функций безопасности протокола IPsec v2 Журнал "Программирование"№ 1, 2011, стр. 36-56.
14. А.В. Никешин, Н.В. Пакулин, В.З. Шнитман. Разработка тестового набора для верификации реализаций протокола безопасности TLS. Труды Института системного программирования РАН, том 23, 2012 г., стр. 387-404.