ИПМ им.М.В.Келдыша РАН

Абрау-2020 • Труды конференции

А.В. Никешин, В.З. Шнитман

Верификация функций безопасности протокола TLS версии 1.3

Рекомендуемая форма библиографической ссылки

Никешин А.В., Шнитман В.З. Верификация функций безопасности протокола TLS версии 1.3 // Научный сервис в сети Интернет: труды XXII Всероссийской научной конференции (21-25 сентября 2020 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2020. — С. 515-526.
https://doi.org/10.20948/abrau-2020-22
https://keldysh.ru/abrau/2020/theses/22.pdf

Видеозапись выступления

Верификация функций безопасности протокола TLS версии 1.3

А.В. Никешин, В.З. Шнитман

Институт системного программирования им. В.П. Иванникова РАН

Аннотация. В данной работе представлен опыт верификации реализаций сервера криптографического протокола TLS версии 1.3. TLS является одним из наиболее востребованных криптографических протоколов, предназначенный для создания защищенных каналов передачи данных – одним из основных механизмов защиты информации в современных сетях. Протокол обеспечивает необходимую для своих задач функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. В новой версии протокола TLS 1.3 была существенно переработана архитектура, устранив ряд недостатков предыдущих версий, выявленных как при разработке реализаций, так и в процессе их эксплуатации. В работе использовался новый тестовый набор для верификации реализаций протокола TLS 1.3 на соответствие спецификациям Интернет, разработанный на основе спецификации RFC8446 с использованием технологии UniTESK и методов мутационного тестирования. Для тестирования реализаций на соответствие формальным спецификациям применяется технология UniTESK, предоставляющая средства автоматизации тестирования на основе использования конечных автоматов. Состояния тестируемой системы задают состояния автомата, а тестовые воздействия – переходы этого автомата. При выполнении перехода заданное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. Мутационные методы тестирования используются для обнаружения нестандартного поведения тестируемой системы (завершение из-за фатальной ошибки, "подвисание", ошибки доступа к памяти) с помощью передачи некорректных данных, такие ситуации часто остаются за рамками требований спецификаций. В сообщения, сформированные на основе разработанной модели протокола, вносятся какие-либо изменения. Модель протокола позволяет вносить изменения в поток данных на любом этапе сетевого обмена, что позволяет тестовому сценарию проходить через все значимые состояния протокола и в каждом таком состоянии проводить тестирование реализации в соответствие с заданной программой. Представленный подход доказал свою эффективность в наших предыдущих проектах при тестировании сетевых протоколов, обеспечив обнаружение различных отклонений от спецификации и других ошибок.

Ключевые слова: безопасность, TSL, TSLv1.3, протоколы, тестирование, оценка устойчивости, Интернет, стандарты, формальные методы спецификации

Verification of security properties of the TLS protocol version 1.3

A.V. Nikeshin, V.Z. Shnitman

Ivannikov Institute for System Programming of the Russian Academy of Sciences

Abstract. This paper presents the experience of verifying server implementations of the TLS protocol version 1.3. TLS is one of the most popular cryptographic protocols designed to create secure data transmission channels – one of the main mechanisms for protecting information in modern networks. The protocol provides the necessary functionality for its tasks: confidentiality of transmitted data, data integrity, and authentication of the parties. In the new version 1.3 of the TLS architecture was significantly redesigned, eliminating a number of shortcomings of previous versions that were identified both during the development of implementations and during their operation. We used a new test suite for verifying implementations of the TLS 1.3 for compliance with Internet specifications, developed on the basis of the RFC8446 specification, using UniTESK technology and mutation testing methods. To test implementations for compliance with formal specifications, UniTESK technology is used, which provides testing automation tools based on the use of finite state machines. The states of the system under test define the states of the state machine, and the test effects are the transitions of this machine. When a transition is made, the specified effect is passed to the implementation under test, after which the implementation's reactions are registered and a verdict is automatically given on whether the observed behavior matches the specification. Mutation methods are used to detect non-standard behavior of the system under test (termination due to a fatal error, "suspension", memory access errors) by transmitting incorrect data, such situations often remain outside the requirements of the specifications. Any changes are made to messages generated based on the developed protocol model. The protocol model allows one to make changes to the data flow at any stage of network exchange, which allows the test scenario to pass through all significant protocol states and in each such state to test the implementation in accordance with the specified program. This approach has proven effective in our previous projects when testing network protocols, providing detection of various deviations from the specification and other errors.

Keywords: security, TSL, TSLv1.3, protocols, testing, verification, evaluate robustness, Internet, standards, formal specifications

1. Введение

TLS является сегодня одним из наиболее востребованных криптографических протоколов, предназначенный для создания защищенных каналов передачи данных и обеспечивающий необходимую для этого функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. В процессе использования протокола неизбежно выявляются его недостатки, появляются новые угрозы безопасности, что приводит к дальнейшему развитию протокола. В августе 2018 года была представлена спецификация новой версии протокола TLS 1.3, в которой, по сравнению с предыдущей версией 1.2, была существенно переработана архитектура [1,2].

Архитектура TLS делится на два уровня обработки сообщений:

– Протокол рукопожатия (Handshake protocol) является основной частью архитектуры TLS. Он выполняет аутентификацию взаимодействующих сторон, согласовывает версию протокола, криптографические режимы и параметры безопасности, устанавливает общий ключевой материал. Встроенные механизмы защиты противодействуют вмешательству третьей стороны и понижению уровня безопасности согласованного криптографического набора.

– Протокол записей (Record protocol), используется для инкапсуляции протоколов более высокого уровня, в том числе протокола рукопожатия. Он использует параметры, установленные протоколом рукопожатия для защиты передаваемых между партнерами данных. Весь трафик делится на последовательность записей, каждая из которых независимо защищается с помощью согласованных криптографических ключей. Протокол записей, работает поверх надежного транспортного протокола (как правило, TCP).

Неудачное рукопожатие или другая ошибка протокола вызывает завершение соединения, которому может предшествовать сообщение уведомления.

TLS не зависит от протоколов прикладного уровня, для которых он является прозрачным, спецификация TLS не определяет схему их взаимодействия. Решение о том, как инициировать TLS-диалог и интерпретировать результаты его работы, оставляется на усмотрение разработчиков протоколов верхнего уровня.

Ниже представлено основное полное рукопожатие TLSv1.3:

```
C→S:    ClientHello                    (Key Exchange phase)
        + key_share*
        + signature_algorithms*
        + psk_key_exchange_modes*
        + pre_shared_key*
C←S:    ServerHello
```

```
                + key_share*
                + pre_shared_key*
               {EncryptedExtensions}          (Server Parameters phase)
               {CertificateRequest*}
               {Certificate*}                  (Authentication phase)
               {CertificateVerify*}
               {Finished}
               [Application Data*]
C→S:
               {Certificate*}
               {CertificateVerify*}
               {Finished}

[Application Data]  <-------> [Application Data]
```

* Обозначает необязательные или зависящие от ситуации сообщения/расширения, которые посылаются не всегда.

{} Обозначает сообщения, защищенные с помощью сеансовых ключей для обмена рукопожатия.

[] Обозначает сообщения, защищенные с помощью сеансовых ключей для прикладных данных.

Назначение сообщений и расширений подробно описаны в спецификации.

Сообщения протокола рукопожатия сгруппированы в три блока:

– Обмен ключами (Key Exchange): Сообщения ClientHello, ServerHello. Устанавливает ключевой материал и выбирает криптографические параметры.

– Параметры сервера (Server Parameters): Сообщения EncryptedExtensions, CertificateRequest. Устанавливает дополнительные параметры рукопожатия (например, запрос аутентификации клиента, поддержка протокола прикладного уровня и др.).

– Аутентификация (Authentication): Сообщения Certificate, CertificateVerify, Finished. Аутентифицирует сервер (и, при неоходимости, клиента), подтверждает созданные сеансовые ключи и обеспечивает целостность всего обмена рукопожатия.

2. Основные особенности протокола TLS версии 1.3

Все сообщения, отправляемые после фазы обмена ключами (т.е. после сообщения ServerHello) зашифрованы соответствующими ключами. Это одна из особенностей данной версии протокола TLS – шифровать как можно больше передаваемых данных.

Сообщение EncryptedExtensions содержит расширения, ответные для соответствующих расширений ClientHello, которые не требуются для

создания криптографических параметров. Такие расширения ранее отправлялись в открытом виде.

После формирования сообщения ServerHello сервер получает возможность вычислить все необходимые сеансовые ключи. Это позволяет начать отправку прикладных данных до приема сообщений аутентификации клиента. Однако необходимо понимать, что в этой точке любые данные посылаются не аутентифицированному клиенту.

Набор сообщений протокола рукопожатия был переработан, чтобы стать более согласованным и ликвидировать излишние сообщения, такие как ChangeCipherSpec (за исключением случаев обеспечения обратной совместимости с промежуточными сетевыми устройствами (middlebox)).

Механизм согласования версий перемещен в расширения, что должно улучшить совместимость с существующими серверами, которые часто неправильно реализуют согласование версий. Также изменен механизм защиты от понижения номера версии.

Изменена концепция набора шифров, чтобы разделить механизмы аутентификации и обмена ключами от алгоритмов защиты данных. Список поддерживаемых алгоритмов переработан и избавлен от всех устаревших и ненадежных алгоритмов. Оставшиеся алгоритмы шифрования являются алгоритмами аутентифицированного шифрования со связанными данными (Authenticated Encryption with Associated Data, AEAD).

Функции вычисления ключей были переделаны. Новая конструкция упрощает специалистам анализ криптографических механизмов благодаря улучшенным свойствам разделения ключей.

Переработан механизм возобновление сессии. Теперь для этого используется общий ключ, согласованный в предыдущем обмене рукопожатия.

Добавлен новый механизм передачи данных 0-RTT (zero round-trip time), позволяющий несколько сократить количество обменов данными между партнерами.

Также стоит отметить, что в более ранних спецификациях TLS определенные части было трудно понять, что приводило к возникновению проблем с функциональной совместимостью реализаций и безопасностью. В данной спецификации более четко проработаны вопросы поведения реализаций в различных ситуациях.

3. Данные 0-RTT

Когда клиент и сервер обладают общим ключом PSK (полученным либо внешними средствами, либо посредством предыдущего обмена рукопожатия), TLS 1.3 позволяет клиенту отправить некоторые данные во время первого рейса (так называемые "ранние данные", early data). Клиент использует PSK для аутентификации сервера и для шифрования ранних данных.

```
C→S:    ClientHello
        + early_data
        + key_share*
        + psk_key_exchange_modes
        + pre_shared_key
        (Application Data*)
C←S:    ServerHello
        + pre_shared_key
        + key_share*
        {EncryptedExtensions}
        + early_data*
        {Finished}
        [Application Data*]
C→S:
        (EndOfEarlyData)
        {Finished}

    [Application Data]  <-------->  [Application Data]
```

Таким образом, данные 0-RTT просто добавляются к рукопожатию 1-RTT в первом рейсе в сообщение ClientHello. Оставшаяся часть рукопожатия использует такие же сообщения, что и механизм возобновления сессии.

Как отмечено в спецификации, свойства безопасности для данных 0-RTT слабее, чем свойства безопасности для других данных TLS:

– Эти данные не обладают свойством прогрессирующей секретности, поскольку зашифрованы ключами, полученными с использованием только предлагаемого ключа PSK.

– Нет защиты от повторного воспроизведения между разными соединениями. Защита от повторного воспроизведения для обычных данных обеспечивается использованием случайного значения (Random) сервера, но данные 0-RTT не зависят от ServerHello. При этом, внутри одного соединения данные 0-RTT дублироваться не могут, поскольку данные 0-RTT и 1-RTT защищаются разными ключами.

4. Верификация протокола

В процессе тестирования сетевых протоколов решаются несколько важных задач: проверяется функциональная совместимость различных реализаций, проверяется соответствие реализации требованиям спецификации и устойчивость реализации к нестандартным воздействиям.

В наших проектах используются наработанные нами методики по тестированию сетевых протоколов: автоматизированное тестирование на соответствие формальным спецификациям и методы мутации данных.

В текущих экспериментах используется модель протокола TLS версии 1.3, разработанная нами на основе спецификаций RFC и описывающая сложную схему функционирования протокола.

Для тестирования реализаций на соответствие формальным спецификациям применяется технология UniTESK, предоставляющая средства автоматизации тестирования на основе использования конечных автоматов [3]. Состояния тестируемой системы задают состояния автомата, а тестовые воздействия – переходы этого автомата. При выполнении перехода заданное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. В UniTESK алгоритм обхода конечного автомата реализован как внутренний компонент и не зависит от протокола и тестируемой системы.

Мутационные методы тестирования используются для обнаружения нестандартного поведения тестируемой системы (завершение из-за фатальной ошибки, "подвисание", ошибки доступа к памяти). Как правило, подобные ситуации не рассматриваются в спецификациях. В сообщения, сформированные на основе разработанной модели протокола, вносятся какие-либо изменения. Модель протокола позволяет менять данные на любом этапе обмена, что позволяет тестовому сценарию проходить через все значимые состояния протокола и в каждом таком состоянии проводить тестирование реализации в соответствие с заданной программой.

5. Тестовый стенд

Для тестирования реализаций сервера протокола TLS используются два сетевых узла. На одном узле функционирует модельная реализация под управлением UniTESK, выполняется основной поток управления тестовыми сценариями, обход тестового автомата и верификация наблюдаемых реакций. На другом узле функционирует тестируемая реализация. Тестовые сообщения протокола, сформированные модельной реализацией, передаются тестируемой системе, после чего регистрируются реакции тестируемого узла.

В качестве реализаций сервера TLSv1.3 были выбраны:

– реализация TLS в виртуальной машине Java, JDK-14 (Java Development Kit) [4],

– реализация TLS библиотеки openssl-1.1.1f [5].

Обе реализации являются частью широко используемых библиотек с открытым исходным кодом, имеют развитую функциональность, в том числе подробную журнализацию событий. Такой выбор, на наш взгляд, оптимален на начальном этапе проекта как для тестирования реализаций, так и, что не менее важно, для более качественной отладки самого тестового набора.

6. Результаты тестирования

Выполняемый нами проект начат в январе 2020 года. Протокол TLSv1.3 позволяет добавлять новую функциональность за счет использования механизма расширений. В нашей работе мы использовали только те расширения, которые необходимы для основной работы протокола. Необязательные расширения, такие как «early data» (0-RTT), в данной работе не использовались. На данный момент в рамках технологии UniTESK (с использованием инструмента JavaTesK [6])

– разработана модель основной функциональности протокола TLS версии 1.3,

– разработаны спецификации и медиаторы для протокола TLS 1.3,

– разработан набор тестов, покрывающий часть требований спецификации.

Найдены несколько отклонений реализаций от спецификаций.

JDK-14:

- сообщение ChangeCipherPlain принимается после ClientFinished (такие сообщения разрешается принимать только до ClientFinished),
- сообщение от клиента с пользовательскими данными (тип Application) между сообщениями ServerFinished и ClientFinished принимается и игнорируется, хотя должно быть сообщения об ошибке (сообщения с типом Application клиент может отправлять только после ClientFinished, при этом для шифрования ClientFinished и Application используются разные ключи),
- сервер допускает несколько циклов сообщений ClientHello - ServerHelloRetry. (Сообщение ServerHelloRetry используется, если ClientHello предлагает приемлемые алгоритмы, но не хватает каких-либо данных. Спецификация TLS разрешает только одно сообщение ServerHelloRetry для каждого соединения),
- после обмена Рукопожатия (Handshake) сервер игнорирует тип входящих сообщений Протокола записей (Record protocol). Данный тип не несет никакой нагрузки поскольку все сообщения зашифрованы, но в целях маскировки должен устанавливаться в значение для пользовательских данных (Application),
- странное поведение сервера: лишние данные после сообщения ClientHello остаются в буфере входящих сообщений сервера и в последующем рассматриваются как начальные байты следующего сообщения. При этом ClientHello не зашифровано,

а остальные входящие сообщения должны быть зашифрованы. Данное поведение можно рассматривать как потенциальную уязвимость.

openssl-1.1.1f:

- не отвечает на сообщение KeyUpdate с флагом 1, хотя для входящих от клиента сообщений создает новые ключи. (Сообщение KeyUpdate уведомляет получателя, что отправитель изменил ключи для своих исходящих сообщений и следующие его сообщения защищаются новыми ключами. Флаг 1 указывает, что получатель должен сделать тоже самое: отправить ответное сообщение KeyUpdate и создать новые ключи для своих исходящих сообщений).

Разработка тестового набора и тестирование продолжается.

7. Заключение

В данной работе представлен опыт верификации реализаций сервера криптографического протокола TLS версии 1.3. TLS является одним из наиболее востребованных криптографических протоколов, предназначенный для создания защищенных каналов передачи данных – одним из основных механизмов защиты информации в современных сетях. Протокол обеспечивает необходимую для своих задач функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. В новой версии протокола TLS 1.3 была существенно переработана архитектура, устранив ряд недостатков предыдущих версий, выявленных как при разработке реализаций, так и в процессе их эксплуатации. В работе использовался новый тестовый набор для верификации реализаций протокола TLS 1.3 на соответствие спецификациям Интернет, разработанный на основе спецификации RFC 8446 с использованием технологии UniTESK и методов мутационного тестирования. Для тестирования реализаций на соответствие формальным спецификациям применяется технология UniTESK, предоставляющая средства автоматизации тестирования на основе использования конечных автоматов. Состояния тестируемой системы задают состояния автомата, а тестовые воздействия – переходы этого автомата. При выполнении перехода заданное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. Мутационные методы тестирования используются для обнаружения нестандартного поведения тестируемой системы (завершение из-за фатальной ошибки, "подвисание", ошибки доступа к памяти) с помощью передачи некорректных данных, такие ситуации часто остаются

за рамками требований спецификаций. В сообщения, сформированные на основе разработанной модели протокола, вносятся какие-либо изменения. Модель протокола позволяет вносить изменения в поток данных на любом этапе сетевого обмена, что позволяет тестовому сценарию проходить через все значимые состояния протокола и в каждом таком состоянии проводить тестирование реализации в соответствие с заданной программой.

Представленный подход доказал свою эффективность в наших предыдущих проектах при тестировании сетевых протоколов, обеспечив обнаружение различных отклонений от спецификации и других ошибок [7-12].

Проект выполняется при поддержке РФФИ, проект № 20-07-00493 «Верификация функций безопасности и оценка устойчивости к атакам реализаций протокола TLS версии 1.3».

Литература

1. Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. IETF RFC 5246. — https://tools.ietf.org/html/rfc5246 .
2. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. August 2018. IETF RFC 8446. — https://tools.ietf.org/html/rfc8446 .
3. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture // Proceedings of FME 2002. LNCS 2391. — P. 77–88, Springer-Verlag, 2002 .
4. Java Development Kit 14.0.1 GA. — URL: https://jdk.java.net/14/ .
5. OpenSSL Project. — https://www.openssl.org/ .
6. JavaTESK. — http://www.unitesk.ru/content/category/5/25/60/ .
7. Никешин А.В., Пакулин Н.В., Шнитман В.З. Разработка тестового набора для верификации реализаций протокола безопасности TLS // Труды ИСП РАН, том 23, 2012. — С. 387–404.
8. Никешин А.В., Пакулин Н.В., Шнитман В.З. Тестирование реализаций клиента протокола TLS // Труды ИСП РАН, том 27, вып. 2, 2015. — С. 145–160.
9. Никешин А.В., Пакулин Н.В., Шнитман В.З. Подходы к разработке тестового набора для тестирования реализаций протокола EAP и его методов // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19–24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В. Келдыша, 2016. — С. 290–297. — https://doi.org/10.20948/abrau-2016-24 .
10. Никешин А.В., Шнитман В.З. Верификация протокола EAP и его методов в беспроводных сетях // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2017. — С. 369–376. — https://doi.org/10.20948/abrau-2017-43 .

11. Никешин А.В., Шнитман В.З. Верификация туннельных методов протокола аутентификации EAP // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17-22 сентября 2018 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2018. — С. 406–416. — https://doi.org/10.20948/abrau-2018-17 .
12. Никешин А.В., Шнитман В.З. Тестирование соответствия реализаций протокола EAP и его методов спецификациям Интернета // Труды ИСП РАН, том 30, вып. 6, 2018. — С. 89–104. — https://doi.org/10.15514/ISPRAS-2018-30(6)-5 .

References

1. Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. IETF RFC 5246. — https://tools.ietf.org/html/rfc5246
2. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. August 2018. IETF RFC 8446. — https://tools.ietf.org/html/rfc8446
3. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture // Proceedings of FME 2002. LNCS 2391. — P. 77–88, Springer-Verlag, 2002
4. Java Development Kit 14.0.1 GA. — https://jdk.java.net/14/
5. OpenSSL Project. — https://www.openssl.org/
6. JavaTESK. — http://www.unitesk.ru/content/category/5/25/60/
7. Nikeshin A.V., Pakulin N.V., Shnitman V.Z. Razrabotka testovogo nabora dlya verifikatsii realizatsiy protokola bezopasnosti TLS // Trudy ISP RAN /Proc. ISP RAS, Vol. 23, 2012. — P. 387–404
8. Nikeshin A.V., Pakulin N.V., Shnitman V.Z. TLS Clients Testing // Trudy ISP RAN /Proc. ISP RAS, vol. 27, issue 2, 2015. — P. 145–160
9. Nikeshin A.V., Pakulin N.V., Shnitman V.Z. Approaches to the development of a test suite for testing implementations of EAP and its methods // Nauchnyi servis v seti Internet: trudy XVIII Vserossiiskoi nauchnoi konferentsii (19-24 sentiabria 2016 g., g. Novorossiisk). — M.: IPM im. M.V.Keldysha, 2016. — P. 290–297. — https://doi.org/10.20948/abrau-2016-24
10. Nikeshin A.V., Shnitman V.Z. Verification of EAP and its methods in wireless networks // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18-23 sentiabria 2017 g., g. Novorossiisk). — M.: IPM im. M.V.Keldysha, 2017. — P. 369–376. — https://doi.org/10.20948/abrau-2017-43
11. Nikeshin, A.V., Shnitman, V.Z. The verification of tunnel methods of the Extensible Authentication Protocol (EAP) // (2018) CEUR Workshop Proceedings, 2260. — P. 406–416. — http://ceur-ws.org/Vol-2260/17_406-416.pdf
12. Nikeshin A.V., Shnitman V.Z. Conformance testing of Extensible Authentication Protocol implementations // Trudy ISP RAN/Proc. ISP RAS,

vol. 30, issue 6, 2018. — P. 89–104 (in Russian). — https://doi.org/10.15514/ISPRAS-2018-30(6)-5