ИПМ им.М.В.Келдыша РАН

Абрау-2021 • Труды конференции

А.В. Никешин, В.З. Шнитман

Верификация функций безопасности расширений протокола TLS 1.3

Рекомендуемая форма библиографической ссылки

Никешин А.В., Шнитман В.З. Верификация функций безопасности расширений протокола TLS 1.3 // Научный сервис в сети Интернет: труды XXIII Всероссийской научной конференции (20-23 сентября 2021 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2021. — С. 251-264.
https://doi.org/10.20948/abrau-2021-14
https://keldysh.ru/abrau/2021/theses/14.pdf

Видеозапись выступления

Верификация функций безопасности расширений протокола TLS 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 – широко распространенный криптографический протокол, предназначенный для создания защищенных каналов передачи данных и обеспечивающий необходимую для этого функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. Новая версия протокола TLS 1.3 была представлена в августе 2018 года и имеет ряд существенных отличий, по сравнению с предыдущей версией 1.2. Ряд разработчиков протокола TLS уже включили поддержку последней версии в свои реализации. Данные обстоятельства делают актуальным проведение исследований в области верификации и безопасности реализаций новой версии протокола TLS. 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. Текущая работа является частью проекта верификации протокола TLS 1.3 и охватывает часть дополнительной функциональности и необязательных расширений протокола. 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 с помощью передачи некорректных данных. В поток обмена протокола, создаваемый в соответствии со спецификацией, вносятся некоторые изменения: либо изменяются значения полей сообщений, сформированных на основе разработанной модели протокола, либо изменяется порядок сообщений в потоке обмена. 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 соответствии с заданной программой. На данный момент были обнаружены несколько отклонений реализаций от спецификации.

Представленный подход доказал свою эффективность в нескольких наших 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 1.3 extensions

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 cryptographic protocol version 1.3. TLS is a widely used cryptographic protocol designed to create secure data transmission channels and provides the necessary functionality for this: confidentiality of the transmitted data, data integrity, and authentication of the parties. The new version 1.3 of the TLS protocol was introduced in August 2018 and has a number of significant differences compared to the previous version 1.2. A number of TLS developers have already included support for the latest version in their implementations. These circumstances make it relevant to do research in the field of verification and security of the new TLS protocol implementations. 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, using UniTESK technology and mutation testing methods. The current work is part of the TLS 1.3 protocol verification project and covers some of the additional functionality and optional protocol extensions. 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 performing a transition, the specified impact is passed to the implementation under test, after which the implementation's reactions are recorded and a verdict is automatically made on the compliance of the observed behavior with the specification. Mutational testing methods are used to detect non-standard behavior of the system under test by transmitting incorrect data. Some changes are made to the protocol exchange flow created in accordance with the specification: either the values of the message fields formed on the basis of the developed protocol model are changed, or the order of messages in the exchange flow is changed. The protocol model allows one to make changes to the data flow at any stage of the network exchange, which allows the test scenario to pass through all the significant states of the protocol and in each such state to test the implementation in accordance with the specified program. So far, several implementations have been found to deviate from the specification. The presented approach has proven effective in several of our

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 – широко распространенный криптографический протокол, предназначенный для создания защищенных каналов передачи данных и обеспечивающий необходимую для этого функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. Новая версия протокола TLS 1.3 была представлена в августе 2018 года и имеет ряд существенных отличий, по сравнению с предыдущей версией 1.2, при этом сохранилась обратная совместимость с ранними версиями. [1,2].

Ряд разработчиков протокола TLS уже включили поддержку последней версии в свои реализации. Данные обстоятельства делают актуальным проведение исследований в области верификации и безопасности реализаций новой версии протокола TLS. 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 данной работе используются наработанные нами методики по тестированию сетевых протоколов: автоматизированное тестирование на соответствие формальным спецификациям и методы мутации данных, доказавшие свою эффективность в наших предыдущих проектах при тестировании сетевых протоколов, обеспечив обнаружение различных отклонений от спецификации и других ошибок.

Предыдущая работа была посвящена верификации базовой функциональности протокола TLSv1.3 [3]. Текущая работа является ее продолжением и охватывает часть дополнительной функциональности и необязательных расширений протокола. На данный момент были обнаружены несколько отклонений реализаций от спецификации.

2. Расширенная функциональность протокола TLS версии 1.3

Основной частью архитектуры TLS является протокол рукопожатия (Handshake protocol). Он выполняет аутентификацию взаимодействующих сторон, согласовывает версию протокола, криптографические режимы и параметры безопасности, устанавливает общий ключевой материал. Различные механизмы защиты противодействуют вмешательству третьей стороны и понижению уровня безопасности согласованного криптографического набора. Основные особенности версии протокола 1.3 представлены в другой нашей статье [3].

Для наглядности напомним 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.

Кроме базового полного рукопожатия, TLS как расширяемый протокол предоставляет дополнительные, необязательные сервисы безопасности, согласуемые, как правило, через механизм расширений. Ниже рассматривается часть таких сервисов, указанных в спецификации TLS.

Режим PSK

TLS 1.3 позволяет использовать сокращенный режим рукопожатия с использованием заранее распределенных ключей (**pre-shared key, PSK**) [2]. Данных режим использует меньше сообщений для обмена и меньше трудоемких вычислений с ключами и может быть полезен в небольших системах, в которых проще сконфигурировать PSK на каждом узле, чем разворачивать инфраструктуру для работы с сертификатами. Также

стороны могут иметь свой механизм установления общего секретного ключа.

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

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

Режим PSK может использоваться в двух вариантах, определяемых расширением psk_key_exchange_modes: PSK и PSK-(EC)DHE. Последний дополняет процедуру формирования сеансовых ключей из PSK алгоритмом Диффи-Хеллмана (используется расширение key_share), что увеличивает как надежность итоговых ключей (обеспечивая прямую секретность, forward secrecy), так и сложность дополнительных криптографических вычислений.

Возобновление сеанса

Рассматриваемая версия протокола, в отличие от предыдущей версии, не выделяет отдельно режим возобновления сеанса. Вместо этого предлагается механизм согласования общего секрета PSK. Для этого используются так называемые удостоверения (tickets) и сообщение NewSessionTicket, которое отправляется сервером клиенту в любое время после завершения сеанса рукопожатия, и может использоваться в следующих соединениях. Сообщений NewSessionTicket может быть несколько, каждое содержит одно удостоверение (ticket), которое с одной стороны используется для формирования нового ключа PSK, с другой – является уникальным идентификатором этого ключа. Также при создании PSK используется соответствующий секрет из предыдущего сеанса рукопожатия, связывая ключевые материалы разных сеансов и обеспечивая дополнительную безопасность соединений.

Данные 0-RTT

На основе режима рукопожатия PSK реализован новый для протокола TLS способ отправки так называемых "ранних данных" (early

data). При наличии общего ключа PSK, TLS 1.3 позволяет клиенту отправить некоторые данные во время первого рейса. Клиент использует PSK одновременно 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.

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, поскольку данные 0-RTT не зависят от нового ServerHello (использующего случайное значение, Random). При этом, внутри одного соединения данные 0-RTT дублироваться не могут, поскольку данные 0-RTT и 1-RTT защищаются разными ключами.

Обновление сеансовых ключей

Для повышения безопасности передачи данных, рекомендуется периодически обновлять ключевой материал. Во время обмена рукопожатия создается базовый ключ соединения (master secret), из которого получаются первоначальные сеансовые ключи. Сообщение KeyUpdate сообщает партнеру, что отправитель создал новые сеансовые ключи, и последующие сообщения зашифрованы новыми ключами.

Расширение Cookie

Если в сообщении ClientHello недостаточно данных для продолжения рукопожатия, сервер может отправить сообщение HelloRetryRequest, требующее прислать исправленное ClientHello с указанным ключевым материалом. При этом сервер может включить в HelloRetryRequest необязательное расширение cookie [2]. Клиент должен скопировать это расширение в своем повторном ClientHello.

Как указано в спецификации, cookie преследует две основные цели:

- Заставляет клиента продемонстрировать достижимость на сетевом адресе (обеспечивая защиту от DoS-атак). Это прежде всего полезно для транспорта, не ориентированного на соединения.
- Позволяет серверу не сохранять состояние сеанса. Поскольку первое сообщение ClientHello (точнее его хеш) участвует в криптографических вычислениях, сервер должен где-то его хранить. Вместо этого сервер может выгрузить эти данные клиенту, включив их в расширение cookie, а клиент вернет его в ответном сообщении.

Расширение Signature algorithms cert

Чтобы указать, какие алгоритмы подписи могут использоваться в цифровой подписи, TLS 1.3 предоставляет два расширения [2]. Расширение "signature_algorithms_cert" применяется к подписям в сертификатах, а расширение "signature_algorithms" применяется к подписям в сообщениях CertificateVerify. В предыдущей версии TLS 1.2 используется только расширение "signature_algorithms". Расширение "signature_algorithms_cert" было добавлено в новой версии протокола, чтобы позволить реализациям, поддерживающим разные наборы алгоритмов для сертификатов и для самого TLS, явно просигнализировать о своих возможностях. Ключи сертификатов должны соответствовать алгоритмам подписи, с которыми они используются. Если расширение отсутствует "signature_algorithms_cert", то предполагается, что реализации используют одну и ту же политику в обоих случаях и расширение "signature_algorithms" применяется также к подписям в сертификатах (что соответствует поведению TLS 1.2).

Расширение Server name

Могут возникнуть ситуации, когда по одному сетевому адресу размещено несколько "виртуальных" серверов, для каждого из которых используется свой сертификат. Данное расширение позволяет клиенту указать имя сервера, с которым он связывается. Для сервера получение этого расширения носит рекомендательный характер. Сервер может

использовать информацию, содержащуюся в расширении, для выбора нужного сертификата, учитывая также и другие настройки политики безопасности. В этом случае серверу следует включить это расширение в соответствующее ответное сообщение. Хотя данное расширение представлено в отдельной более ранней спецификации, его поддержка и использование является обязательным для реализаций TLS 1.3 [4].

Расширение Max fragment length / Record size limit

Если клиент хочет использовать сообщения меньшего размера, чем предусмотрено спецификацией (2^14 байт), то он может воспользоваться расширением "max_fragment_length" [4]. Однако у него есть ряд существенных недостатков:

- Данное расширение предлагает к применению всего несколько фиксированных значений, при этом максимальное значение (2^12 байт) сильно меньше предлагаемого спецификацией TLS (2^14 байт).
- Добавление новых значений не предусмотрено.
- Сервер не может запросить более низкое значение, чем то, которое предложил клиент. Это серьезная проблема, если сервер более ограничен.
- Расширение плохо подходит для случаев, когда возможности клиента и сервера неодинаковы, т.е. если, например, конечная точка хочет отправлять записи большего размера, чем те, которые она получает.

Чтобы обойти эти ограничения было предложено другое расширение для TLS - "record size limit" [5]. Оно позволяет задать произвольное максимальное значение сообщений (не превышающее значение, определенное согласованной партнерами версией протокола) для каждого направления передачи данных.

Расширение Post handshake auth

Расширение "post_handshake_auth" добавлено в TLS 1.3 и используется для указания того, что клиент хочет выполнить аутентификацию после рукопожатия [2]. В этом случае сервер может запросить аутентификацию клиента в любой момент времени после завершения рукопожатия путем посылки сообщения CertificateRequest. Клиент должен ответить соответствующими сообщениями аутентификации.

C←S: [CertificateRequest]
C→S:
[Certificate]
[CertificateVerify*]
[Finished]

Если у клиента нет соответствующих сертификатов, то отправляется пустое сообщение Certificate, а сообщение CertificateVerify не используется.

Сервер может отправить несколько сообщений CertificateRequest, как в разное время, так и последовательно (например, если требуется доступ к нескольким сервисам). При этом ответы могут приходить в произвольной последовательности (для разделения запросов используются соответствующие уникальные идентификаторы).

Такая функциональность протокола может также использоваться и для режима рукопожатия PSK, во время которого сертификаты не используются, но зато после завершения рукопожатия можно запросить сертификат клиента (если ранее клиент включил расширение "post_handshake_auth" в сообщение ClientHello).

3. 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 используется технология UniTESK, предоставляющая средства автоматизации тестирования на основе использования конечных автоматов [6]. 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 поток обмена протокола, создаваемый в соответствии со спецификацией, вносятся некоторые изменения: либо в сообщениях, сформированных на основе разработанной модели протокола, изменяются значения полей, либо

изменяется порядок сообщений в потоке обмена. 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 соответствии с заданной программой.

4. 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 используются:

– реализация TLS в виртуальной машине Java, JDK-14 (Java Development Kit) [7],

– реализация TLS библиотеки openssl-1.1.1f [8],

– реализация TLS в Windows 11 Pro 21H2 22000-100 (IIS).

Данные реализации являются частью широко используемых библиотек с открытым исходным кодом, имеют развитую функциональность и обеспечивают хорошую журнализацию событий.

5. Результаты тестирования

Протокол TLSv1.3 позволяет добавлять новую функциональность за счет использования механизма расширений. В предыдущей части проекта акцент был сделан использовании базовой функциональности протокола с минимальным количеством обязательных расширений. В данном работе используются дополнительные, описанные выше, возможности протокола. На данный момент в рамках технологии UniTESK (с использованием инструмента JavaTesK [9]) получены результаты:

– расширена модель функциональности протокола TLS версии 1.3,

– разработан новый набор тестов, покрывающий часть требований спецификации.

Найдены несколько отклонений реализаций от спецификации, касающиеся обработки расширения cookie. Напомним, что в ответ на сообщение ClientHello, в котором отсутствует необходимый ключевой материал для выбранных сервером алгоритмов, сервер может отправить сообщение HelloRetryRequest, требующее предоставить необходимые данные. После чего клиент должен прислать исправленное ClientHello,

которое полностью повторяет первое, но с требуемым ключевым материалом. При этом, если сервер включил в HelloRetryRequest необязательное расширение cookie, клиент должен скопировать это расширение в свое второе ClientHello.

JDK-14:

- Реализация использует расширение cookie в сообщении HelloRetryRequest. При этом реализация успешно принимает ответное ClientHello с другим набором криптографических алгоритмов (поле TLSCipherSuite), если клиент не включил в него расширение cookie (если cookie присутствует, то реализация требует точного повторения всех полей, как и положено по спецификации).
- Сообщение HelloRetryRequest содержит алгоритм, для которого требуется прислать ключевой материал. Если клиент включил во второе ClientHello расширение cookie, то он может отправить ключевой материал для другого алгоритма (отличного от того, что запросил сервер). Если сервер поддерживает этот новый алгоритм, то сообщение принимается.

Таким образом, в ответ на HelloRetryRequest клиент может поменять в новом ClientHello предлагаемые алгоритмы. Однако такое поведение сервера может быть следствием того, что сервер не сохраняет первоначальное состояние сеанса (спецификация допускает работу сервера без сохранения состояния).

openssl-1.1.1f:

- Реализация не использует расширение cookie в сообщении HelloRetryRequest (что не запрещается), но при этом игнорирует присутствие и содержание данного расширения в ответном сообщении ClientHello.
- Реализация игнорирует присутствие и содержание данного расширения в начальном сообщении ClientHello (расширение cookie используется клиентом только в повторном ClientHello в ответ на получение HelloRetryRequest).

Internet Information Services, Windows 11:

- Сервер отвечает на сообщение ClientHello с единственным расширением SupportedVersions<3.4> (спецификация требует присутствия в ClientHello еще нескольких обязательных расширений помимо SupportedVersions, таким образом серверу следовало бы ответить ошибкой и разорвать соединение).

- Сервер принимает ClientHello с дубликатами расширений (спецификация допускает присутствие только по одному расширению каждого типа).
- Сервер принимает от клиента сообщения Application и RecordLayerUndefinedTypeMessage до получения ClientFinished (т.е. до завершения рукопожатия, эти сообщения уже зашифрованы сеансовыми ключами). Сервер игнорирует эти сообщения (хотя они нарушают допустимый порядок сообщений рукопожатия) и продолжает сеанс вместо разрыва соединения. RecordLayerUndefinedTypeMessage – тестовый тип сообщения, в котором в заголовке RecordLayer поле Type выставлено в произвольное не зарегистрированное значение; Application – сообщение с пользовательскими данными.

6. Заключение

В данной работе представлен опыт верификации реализаций сервера криптографического протокола TLS версии 1.3. TLS – широко распространенный криптографический протокол, предназначенный для создания защищенных каналов передачи данных и обеспечивающий необходимую для этого функциональность: конфиденциальность передаваемых данных, целостность данных, аутентификацию сторон. Новая версия протокола TLS 1.3 была представлена в августе 2018 года и имеет ряд существенных отличий, по сравнению с предыдущей версией 1.2. Ряд разработчиков протокола TLS уже включили поддержку последней версии в свои реализации. Данные обстоятельства делают актуальным проведение исследований в области верификации и безопасности реализаций новой версии протокола TLS. 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. Текущая работа является продолжением нашего проекта верификации протокола TLS 1.3 и охватывает часть дополнительной функциональности и необязательных расширений протокола. 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 поток обмена протокола, создаваемый в соответствии со спецификацией, вносятся некоторые изменения: либо изменяются значения полей в сообщениях, сформированных на основе разработанной модели протокола, либо изменяется порядок сообщений в потоке обмена. 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 соответствии с заданной программой.

На данный момент были обнаружены несколько отклонений реализаций от спецификации. Представленный подход доказал свою эффективность в наших предыдущих проектах при тестировании сетевых протоколов, обеспечив обнаружение различных отклонений от спецификации и других ошибок [10-12].

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. 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 .
4. Eastlake D. Transport Layer Security (TLS) Extensions: Extension Definitions. January 2011. IETF RFC 6066. — https://tools.ietf.org/html/rfc6066 .
5. Thomson M. Record Size Limit Extension for TLS. August 2018. IETF RFC 8449. — https://tools.ietf.org/html/rfc8449 .
6. 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 .
7. Java Development Kit 14.0.1 GA. — URL: https://jdk.java.net/14/ .
8. OpenSSL Project. — https://www.openssl.org/ .

9. JavaTESK. — http://www.unitesk.ru/content/category/5/25/60/ .
10. 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.
11. 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.
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. Nikeshin A.V., Shnitman V.Z. Verification of security properties of the TLS protocol version 1.3 // Nauchnyi servis v seti Internet: trudy XXII Vserossiiskoi nauchnoi konferentsii (21–25 sentiabria 2020 g., online). — M.: IPM im. M.V.Keldysha, 2020. — P. 515–526. — https://doi.org/10.20948/abrau-2020-22 .
4. Eastlake D. Transport Layer Security (TLS) Extensions: Extension Definitions. January 2011. IETF RFC 6066. — https://tools.ietf.org/html/rfc6066 .
5. Thomson M. Record Size Limit Extension for TLS. August 2018. IETF RFC 8449. — https://tools.ietf.org/html/rfc8449 .
6. 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
7. Java Development Kit 14.0.1 GA. — https://jdk.java.net/14/
8. OpenSSL Project. — https://www.openssl.org/
9. JavaTESK. — http://www.unitesk.ru/content/category/5/25/60/
10. 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
11. 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
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