Отчет о дополнительных расширениях для TLS

Дасько Ф.Б.

18 ноября 2022 г.

Аннотация

1 Введение

Данная статья рассматривает обновление стандарта `СТБ 34.101.65 - 2014` (Информационные технологии и безопасность ПРОТОКОЛ ЗАЩИТЫ ТРАНСПОРТНОГО УРОВНЯ (TLS)).

Данный стандарт принят в 2014 году и описывает применение протокола TLS 1.2. За последние 8 лет были найдены новые подходы и оптимизации данного протокола. В основном реализуемы они при помощи расширений (`extensions`).

Данный подход позволяет с небольшими изменениями реализации протокола перейти к более обновленной и более надежной версии.

Эти 8 лет протокол TLS не стоял на месте. Некоторые части протокола обновлялись и оптимизировались, в других обнаруживались уязвимости и они подпадали под запрет использования.

В связи с этим возникла необходимость внести коррективы или пояснения в Белорусский стандарт описывающий TLS.

2 Методология

Опишем подходы при помощи которых был проведен поиск наиболее важных обновлений для TLS 1.2.

1. Наиболее полное описание протокола TLS и его обновлений содержат в себе документы RFC (Request for Comments). Найти полный список можно по ссылке тут.
2. Для данного вопроса интерес представляют лишь RFC вышедшие после августа 2008 года. Тогда вышел RFC объявляющий TLS 1.2. При этом RFC, очевидно, должен быть посвящен TLS. Таких RFC 147.
3. При этом наибольший интерес представляют RFC имеющие статус «proposed standart» таких RFC 96.
4. В данный список из 96 RFC попали так же посвященные DTLS (Datagram Transport Layer Security), посвященные TLS иных версий (например RFC 8997), другим протоколам действующих поверх TLS (например RFC 9103) и обновления TLS применяемые лишь в некоторых случаях (например RFC 7817).
5. В отдельную категорию выделим RFC посвященные криптонаборам. В отличии от предыдущих категорий данные RFC следует учитывать при использовании TLS, но не требуют изменения в стандарте. Приведем тут список.

Введены	**Запрещены**	**Имя**
5288		AES GCM
5289		ECC Suites with SHA-256/384 and AES GCM
5487		PSK Cipher Suites with SHA-256/384 and AES GCM
5932		Camellia Cipher Suites
6655		AES-CCM Cipher Suites
7905		ChaCha20-Poly1305 Cipher Suites
8422		ECC Cipher Suites Versions
8442		ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites
	7465	RC4 Cipher Suites
	9155	MD5 and SHA-1 Signature Hashes

6. Таким образом можно отбросить множество излишних RFC оставив лишь 31.

номер RFC	**название**
5425	Transport Layer Security (TLS) Transport Mapping for Syslog
5705	Keying Material Exporters for Transport Layer Security (TLS)
5746	Transport Layer Security (TLS) Renegotiation Indication Extension
5878	Transport Layer Security (TLS) Authorization Extensions
5929	Channel Bindings for TLS
5953	Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)
6066	Transport Layer Security (TLS) Extensions: Extension Definitions
6546	Transport of Real-time Inter-network Defense (RID) Messages over HTTP/TLS
6698	The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA
6961	The Transport Layer Security (TLS) Multiple Certificate Status Request Extension
7250	Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
7301	Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension
7366	Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
7507	TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks
7590	Use of Transport Layer Security (TLS) in the Extensible Messaging and Presence Protocol (XMPP)
7627	Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension
7633	X.509v3 Transport Layer Security (TLS) Feature Extension
7673	Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records
7685	A Transport Layer Security (TLS) ClientHello Padding Extension
7817	Updated TLS Server Identity Check Procedure for Email-Related Protocols
7919	Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS
7924	Transport Layer Security (TLS) Cached Information Extension
7925	TLS / DTLS Profiles for the Internet of Things
8122	Connection-Oriented Media Transport over the TLS Protocol in SDP
8310	Usage Profiles for DNS over TLS and DNS over DTLS
8314	Cleartext Considered Obsolete: Use of TLS for Email Submission and Access
8447	IANA Registry Updates for TLS and DTLS
8449	Record Size Limit Extension for TLS
8460	SMTP TLS Reporting
8472	TLS Extension for Token Binding Protocol Negotiation
8737	ACME TLS Application-Layer Protocol Negotiation (ALPN) Challenge Extension
8842	SDP Offer/Answer Considerations for DTLS and TLS
8879	TLS Certificate Compression
9149	TLS Ticket Requests
9261	Exported Authenticators in TLS

7. Исследовав данные RFC подробнее существенными для обновления стандарта приняты следующие RFC : 5746, 6066, 6961, 7366, 7627, 7633.

Далее приведем указания и разбор дополнений вызванных данными RFC.

3 Дополнение 1 SCSV

3.1 Проблематика

В стандарте `СТБ 34.101.65` (в пункте 8.7.2) вводится расширение renegatiation_info, оно используется для связывания соединения, в котором выполняется переустановка связи, с предыдущим соединением.

Однако некоторые реализации SSLv3 и TLS 1.0 неправильно завершают рукопожатие в таком случае. Поскольку спецификации TLS 1.1 и ранее требуют игнорировать данные после ClientHello (т.е. расширения), если они их не понимают. Это означает, что клиенты, предлагающие расширение "renegotiation_info могут столкнуться с ошибками в процессе handshake.

3.2 Решение

Из соображений улучшения обрастной совместимости в `RFC 5746` (пункт 3.3) в сообщении `client_hello` вводится возможность использования специального значения Signaling Cipher Suite Value (SCSV) «TLS_EMPTY_RENEGOTIATION_INFO_SCSV» с кодовой точкой 0x00, 0xFF.

Этот SCSV не является настоящим криптонабором и не может быть согласован. Вместо этого он имеет ту же семантику, что и у пустого расширения "renegotiation_info". Поскольку реализации SSLv3 и TLS надежно игнорируют неизвестные криптонаборы, SCSV может быть безопасно отправлен на любой сервер. SCSV также может быть включен в CLIENT-HELLO, обратно совместимый с SSLv2.

3.3 Вывод

Данный механизм полезен, однако существеннен лишь для обратной совместимости с устаревшими серверами. Его использование допускает полное замещение расширения renegotiation_info на сигнальный криптонабор TLS_EMPTY_RENEGOTIATION_INFO_SCSV. Кроме того, допустимо и совметсное их использование.

4 Планованое EMS

4.1 Проблематика EMS

В стандарте `СТБ 34.101.65` указывается что существует атака "triple handshake". Её суть состоит в том что активный злоумышленник может организовать атаку "человек по середине"(mitm) за счет того что задаст параметры соединения таким образом что `master_secret` в обоих сеансах совпадет (полное описание атаки в RFC 7627 пункт 1.). Однако при этом стандарт не определяет механизмов противодействия данной атаке.

4.2 Концепт решения

Решение противодействующее данной атаке предоставляет `RFC 7626` при помощи расширения `extended_master_secret`. Суть его состоит в изменении вычисления `master_secret` таким образом что значение становится зависимых от данных переданных ранее, что не позволит злоумышленнику задать два сеанса с одинаковыми `master_secret`.

4.3 Реальное решение

Решение: Когда происходит полное рукопожатие TLS, мы определяем `session_hash = Hash(handshake_messages)`

Где "handshake_messages"предствляет собой конкатенацию всех отправленных и полученных структур в сообщениях начиная с `ClientHello` вплоть до `ClientKeyExchange`

включительно, в том числе типы и длины полей этих структур. (при этом, для рукопожатий возобновления соединения вычисления не потребуются, так как они не приводят к созданию новой сессии.)

Ранее `master_secret` определялся как `master_secret = PRF(pre_master_secret, "master secret ClientHello.random + ServerHello.random) [0..47];`

В данном расширении он определяется как:

`master_secret = PRF(pre_master_secret, "extended master secret session_hash) [0..47];`

При этом "extended master secret"представляет собой то же что и "master secret".

Расширение `extended_master_secret` имеет кодировку в шеснадцатиричном виде 00 17 00 00 (hex).

Данное расширение весьма существенно и должно постоянно применяться на практике.

Для полного протокола handshake:

- Клиент поддерживающий данное расширение ДОЛЖЕН отправить расширение "extended_master_secret"в сообщении ClientHello.

  Сервер получивший расширение "extended_master_secret"и поддерживающий данное расширение ДОЛЖЕН включить расширение "extended_master_secret"в его serverHello.

  Таким образом расширение будет согласовано и применено.

- Если сервер получает clientHello без расширения "extended_master_secret то ему СЛЕДУЕТ разорвать handshake, если он не хочет взаимодействовать с устаревшими клиентами. Если он решит продолжить рукопожатие, ему НЕЛЬЗЯ включать расширение в ServerHello.

- Если клиент получает serverHello без расширения "extended_master_secret, то клиенту СЛЕДУЕТ разорвать handshake, если он не хочет взаимодействовать с устаревшим сервером.

- Если клиент и сервер решат продолжать handshake без данного расширения, то они ДОЛЖНЫ использовать стандартное вычисление master_secret. В таком случае сессия не будет защищена в соответствии с данным расширением.

Для сокращенного протокола handshake (сервер получает сообщение client hello для сокращенного протокола handshake):

- Если исходный сеанс не использовал «extended_master_secret», но сообщение ClientHello содержит расширение, то сервер ДОЛЖЕН разорвать сокращенный протокол handshake, и СЛЕДУЕТ продолжить полным рукопожатием для согласования нового сеанса.

- Если исходный сеанс использовал «extended_master_secret», но новое сообщение ClientHello не содержит его, то сервер ДОЛЖЕН прервать сокращенное рукопожатие.

- Если исходный сеанс не использовал «extended_master_secret» и оно отсутствует в ClientHello, то серверу СЛЕДУЕТ разорвать соединение, иначе оно не будет защищено в соответствии с данным расширением.

- Если новое сообщение ClientHello содержит данное расширение и сервер решает продолжить рукопожатие, тогда сервер ДОЛЖЕН включить расширение «extended_master_
в своем сообщении ServerHello.

Для сокращенного протокола handshake (клиент получает сообщение server hello для сокращенного протокола handshake): сокращенный протокол handshake ДОЛЖЕН быть прерван если новое сообщение не соответствует предыдушей сессии относительно расширения "extended_master_secret". Если клиент и сервер продолжают процедуру сокращенного handshake, то ключи подключения для нового сеанса они получают стандартным образом.

4.4 Вывод

Расширение extended_master_secret критически важно и ДОЛЖНО использоваться для поддержания должного уровня защищенности соединения.

5 Планованое E-t-M

5.1 Проблематика

Каноничный подход MAC-then-ENCRYPT, предложенный как в стандарте `СТБ 34.101.65`, так и в `RFC 5246` объявляющих TLS 1.2, подвергался множеству атак и кроме того обладал некоторыми неудобствами (как минимум то неудобство, что для проверки корректности сообщения его требовалось сначала расшифровать). В связи с чем возникло и рекомендуется применение иного подхода.

5.2 Концепт рещения

В `RFC 7366` вводится расширение `encrypt_then_mac` которое позволяет согласовать применение данной технологии, что увеличивает надежность протокола. Кроме того вводятся новые механизмы защиты от понижения версии. Так после согласования `encrypt_then_mac` пересогласоваться на отказ от него нельзя. Что приводит к некоторым особенностям работы. Так криптонаборы класса `AEAD` воспринимаются как не соответствующие MAC-then-ENCRYPT, что не позволит при пересогласовании перейти к данным криптонаборам.

5.3 Реальное решение

Идет переход при вычислении от encrypt(data || MAC || pad) к encrypt(data || pad) || MAC . Где MAC вычисляется на основании всей зашифрованной части.

Более точно:

```
 MAC(MAC_write_key, seq_num +
TLSCipherText.type +
TLSCipherText.version +
TLSCipherText.length +
IV +
ENC(content + padding + padding_length));
```

а общий пакет данных имеет вид:

```
struct
ContentType type;
ProtocolVersion version;
uint16 length;
GenericBlockCipher fragment;
opaque MAC;
TLSCiphertext;
```

5.4 Вывод

Расширение ENCRYPT-then-MAC является важным расширением существенно влияющим на безопасность соединения. Его применение необходимо, в особенности при некоторых криптонаборах уязвимых при MAC-then-ENCRYPT. Стоит так же учитывать существование таких криптонаборов которые не могут сочетаться с использованием данного расширения (такие как крипторабоpы AEAD). Однако стоит и учитывать что в целом отсутствие данного расширения не обязательно говорит о некоторой уязвимости соединения. Данный вопрос оказывается неотделим от выбора криптонабора.

6 Планованое OSCP

6.1 Проблематика

Кроме вопросов безопасности расширения TLS позволяют решать задачи оптимизации процесса установления защищенного соединения. Так одним из факторов некоторого замедления установления соединения можно выделить запрос сертификата сервера. Оптимизировать данный процесс позволяют протоколы OSCP (Online Certificate Status Protocol) позволяющие получать подтверждение корректности сервера сушественно быстрее.

6.2 Концепт решения

`RFC 6066 пункт 8, 6961` объявляют расширения `status_request` и `status_request_v2` позволяющих пользоваться прокотолом OSCP для получения статуса протокола существенно быстрее нежели стандартный запрос протокола и его стандартная проверка. `status_request_v2` концептуально отличается от `status_request` тем, что позволяет получать статусы сразу всей цепочки сертификатов. Но основная суть расширения сохраняется.

6.3 реальное решение для status_request

- Клиент включает в сообщение client_hello расширение status_request.
- Сервер МОЖЕТ включить в ответ соответствующий статус сертификата вместе со своим сертификатом.

  Если запрашивается OSCP, то серверу СЛЕДУЕТ использовать информацию содержащуюся в расширении для выбора OSCP ответчика и включить request_extensions в OSCP запрос.
- Сервера в ответ возыращают ответ с свой сертификат и сообщение CertificateStatus. Если сервер(а) возвращает(ют) сообщение "CertificateStatus то сервер ДОЛЖЕН включать расширение типа «status_request» с пустым «extension_data» в extended server hello.
- "CertificateStatus"посылается в соответствии с типом "certificate_status"протокола handshake слудующим образом:

```
 struct { CertificateStatusType status_type;
select (status_type) {
case ocsp: OCSPResponse;
} response;
} CertificateStatus;
opaque OCSPResponse<1..2^24 − 1>;
```

6.4 реальное решение для status_request_v2

Действия как клиента так и сервера для расширения status_request_v2 совпадает с поведением при расширении status_request однако несколько отличаются используемые структуры, а именно: struct { CertificateStatusType status_type; uint16 request_length; /* Length of request field in bytes */ select (status_type) { case ocsp: OCSPStatusRequest; case ocsp_multi: OCSPStatusRequest; } request; } CertificateStatusRequestItemV2;

enum { ocsp(1), ocsp_multi(2), (255) } CertificateStatusType;

struct { ResponderID responder_id_list<0..2^16-1>; Extensions request_extensions; } OCSPStatusRequest;

opaque ResponderID<1..2^16-1>; opaque Extensions<0..2^16-1>;

struct { CertificateStatusRequestItemV2 certificate_status_req_list<1..2^16-1>; } CertificateStat

6.5 Вывод

Расширения status_request и status_request_v2 являются важными расширениями ускоряющими установление защищенного соединения через поддержание протоколов OSCP. Их использование не повышает уровень защищенности соединения однако все еще является крайне желательным при применении.

7 планованое расширение x509v3

7.1 проблематика

Существует целый набор различных атак на TLS и из них можно выделить отдельным классом атаки основанные на том что используются версии протокола где некоторые уязвимости не закрыты. И как не меняй текущий стандарт они закрыты не будут так как уже базируются лишь для устаревших версий. Однако и возможность поддерживать обратную совместимость и доступность устаревших серверов необходимо.

7.2 Концепт решения

`RFC 7633` описывает расширение базирующееся на `status_request` и `status_request_v2`. Задача которого обеспечить защиту от атак понижения версии, которые ранее оставались открытыми.

Данный документ предъявляет требования к виду сертификата, который в себе несет информацию о свойствах сервера которому он предоставляется, что не позволит серверу отказывать в расширениях и функциях которыми он обладает. Таким образом реализуется так же защита от отказа в обслуживании, что позволяет определить что действия сервера являются легитимными и он предоставляет на столько полную защиту на сколько это возможно.

7.3 Вывод

Однако, поскольку данные свойства реализуются через сертификаты, а не силами самого протола TLS, то данное расширение (обновление) не влияет на описание самого протокола и не требует внесения изменений в стандарт.