МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

ПРРКС

“Дослідження особливостей реалізації криптографічних механізмів протоколу SSL/TLS”

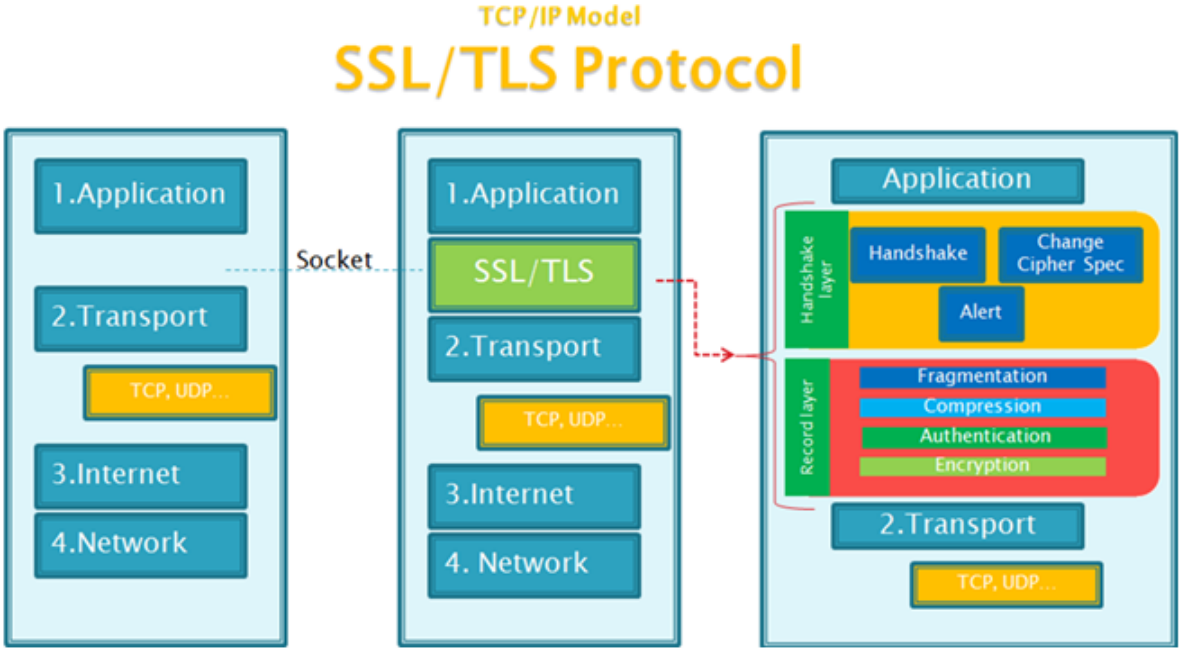
Виконали:

Студенти групи ФІ-22мн
Бондаренко Андрій
Яценко Артем

Київ – 2023

Вступ

Бум Інтернету, веб-технологій об’єднує весь світ під єдиним дахом. Передача інформації через електронні шляхи призводить до того, що безпека є важливим аспектом, з яким потрібно мати справу. В IP-мережі протокол SSL/TLS працює на верхньому рівні транспортного рівня для захисту трафіку додатків і забезпечує наскрізний безпечний зв’язок. Діра в безпеці в цих протоколах робить канал зв’язку вразливим для прослуховування та зміни інформації пізніше. Розглянемо архітектури SSL і TLS і наведемо перелік атак на SSL/TLS. Розглянемо фактори, що впливають на ці атаки.



Структура протокола SSL та підпротоколів з визначенням функцій

Сьогодні в діловому світі Всесвітня павутина (WWW) є зразком для наслідування кожної дії. Оскільки попит зростає, це вимагає перетворення веб-сервісів на безпечні веб-сервіси. Протоколи Secure Socket Layer (SSL)/Transport Layer Security (TLS) використовуються для надання надійних послуг через протокол транспортного рівня. SSL пройшов кілька оновлень, таких як SSLv1.0, SSLv2.0 і SSLv3.0 тощо. SSLv3.1 в основному називається TLSv1.0, який забезпечує зворотну сумісність із попередньою версією SSL. Протокол SSL працює на двох рівнях служб де перший – це SSL-з’єднання, а другий – SSL-сеанс. Підключення SSL працює на

транспортному рівні для встановлення зв'язків між клієнтами та серверами. Однорангові асоціації дозволяють створювати сеанси, що є ефемерним. Кожен сеанс SSL пов'язаний з одним з'єднанням SSL. Протокол встановлення зв'язку SSL/TLS використовується для створення сеансу шляхом обміну парою параметрів (наприклад, випадкове число, ідентифікатор сеансу, набір шифрів, методи стиснення тощо). Кожен сеанс підтримується переважно двома станами. Стан сеансу має справу з рядом параметрів, таких як ідентифікатор сеансу, сертифікат X509, методи стиснення, специфікація шифру, головний секрет тощо. Параметр стану з'єднання включає в себе секретні MAC-адреси надсилання сервером і клієнтом, вектори ініціалізації, порядкові номери тощо.

SSL — це суміш чотирьох протоколів, які забезпечують безпеку протоколам верхнього рівня, таким як HTTP, FTP і будь-якому протоколу прикладного рівня. Вони розподіляються на два шари (див. Малюнок 1).

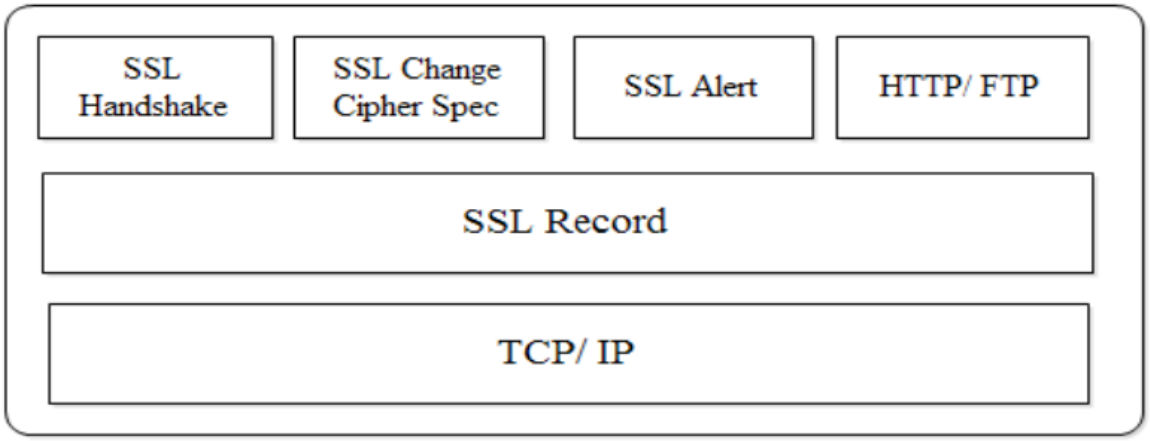


Fig 1: SSL protocol layer structure

Протокол запису SSL (SSL Record Protocol)

Він працює як основа для інших трьох протоколів і забезпечує конфіденційність і цілісність повідомлень верхнього рівня. На сайті відправника він сегментує інформацію на кілька фрагментів, стискає їх, обчислює MAC і шифрує фрагменти разом із відповідним MAC. На сайті одержувача ці процеси виконуються в протилежному напрямку до того, як оригінальні повідомлення будуть доставлені одержувачу. За замовчуванням стиснення вимкнено в SSLv3.0 і всіх версіях TLS.

Структура потоку створення SSL-пакетів розділена на п’ять частин (див. рис. 2).

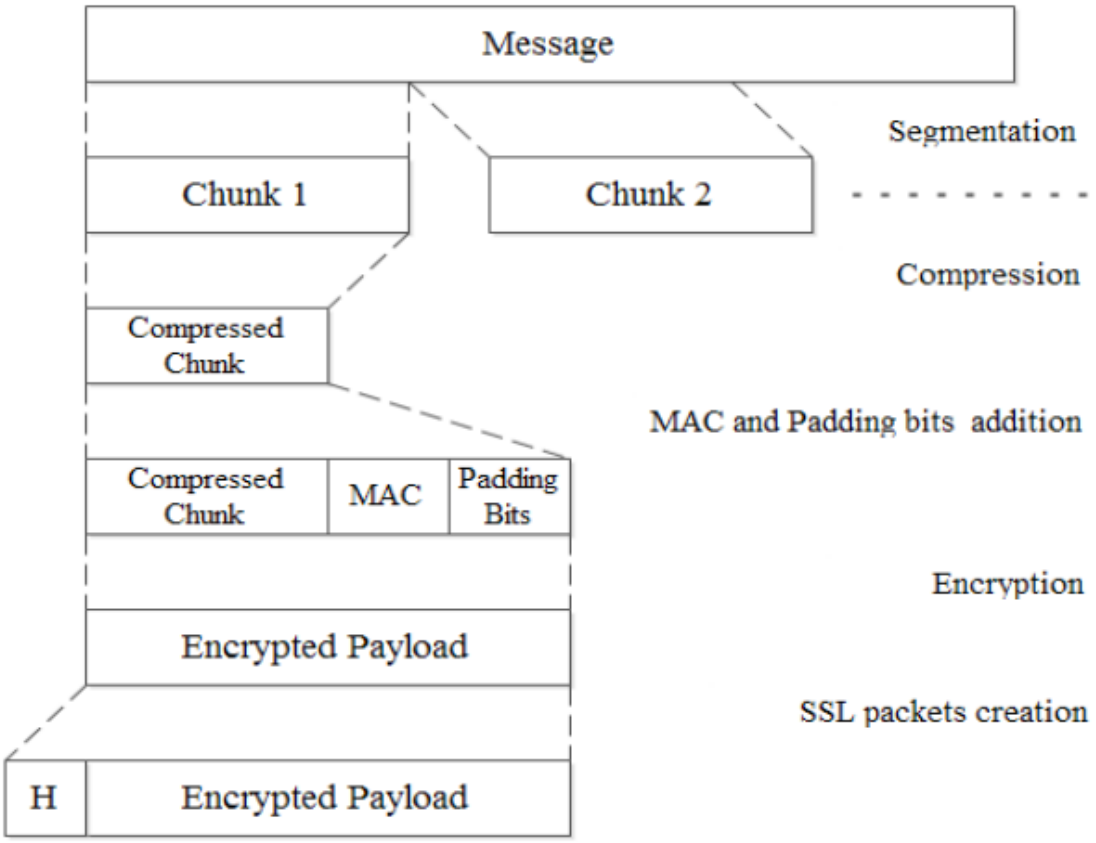


Fig 2: SSL record protocol working principle

1. *Алгоритми стиснення*: методи стиснення без втрат викликаються протоколом запису SSL для стиснення даних без будь-яких втрат. (наприклад, кодування Huffman, LZ77, GZIP тощо)
2. *Геш-алгоритми*. Захищені геш-функції відіграють важливу роль у забезпеченні конфіденційності кожного сегмента даних. Найбільш популярні алгоритми MD5 і SHA використовуються для обчислення MAC. (наприклад, MD5, SHA-1, SHA-224, SHA-256 тощо).
3. *Алгоритми шифрування*: для створення корисного навантаження SSL використовуються методи симетричного потоку або блокового шифрування. У разі шифрування потоковим шифром стиснений фрагмент і MAC шифруються разом. Біти заповнення додаються вздовж MAC до блоку перед шифруванням блокового шифру. Симетричні алгоритми з розмірами ключів наведено в таблиці 1 і 2.

Table 1. Stream encryption algorithms and key sizes

Algorithms	Key Sizes (bits)
RC4	40 or 128

Table 2. Block encryption algorithms and key sizes

Algorithms	Key Sizes (bits)
RC2	40
DES	40 or 56
Fortezza	80
IDEA	128
3DES	168
AES	128 or 256

Як описано вище, геш-функція зі спільним секретним ключем використовується для обчислення хешованого коду автентифікації повідомлення (HMAC), де «+» означає операцію конкатенації. Його оцінка наведена нижче.

HASH (MAC_secret_key + pad_2 +
HASH (MAC_secret_key + pad_1 + seq_no + compression_type +
compressed_chunk_length + compressed_chunk)

(1)

Довжина бітів заповнення (pad_1 і pad_2) для MD5 і SHA-1 становить 384 біти і 320 бітів відповідно. Сегментація дозволяє максимальний розмір фрагмента 2^{14} байт. Якщо застосовано стиснення, довжина фрагмента після стиснення не перевищує 1024 байтів. Таким чином, максимальний розмір MAC-адреси становить 1024 байтів. 2^{14} + 2048 байт — це максимальна довжина корисного навантаження SSL, яка змушує алгоритм шифрування обмежувати інкрементну довжину не більше 1024 байтів. Нарешті заголовок SSL додається до корисного навантаження SSL перед тим, як пакети надсилаються на нижній рівень. Заголовок протоколу запису SSL складається з чотирьох основних полів, таких як тип вмісту, основна версія, додаткова версія та стиснута довжина. Тип вмісту визначає *handshake, change_cipher_spec, application_data ma alert*, які надають інформацію протоколу верхнього рівня для обробки фрагментів/сегментів на стороні приймача. Основна та допоміжна версії визначають використовувану основну та допоміжну версії SSL. Стиснута довжина вказує розмір корисного навантаження SSL у байтах.

SSL Change Cipher Spec Protocol

Це один із найпростіших протоколів, який використовує протокол запису SSL і має справу з одним байтом. Байт зі значенням 1 вказує на те, що поточний стан оновлено, а стан, що залишився, викликає активацію нового набору шифрів для поточного посилання. Зазвичай за новим хендшейком SSL слідує повідомлення про зміну специфікації шифру.

Протокол сповіщень SSL (SSL Alert Protocol)

Він поширює помилки на однорангові пристрої під час узгодження та з'єднання SSL. Він має справу з двома байтами, які стискаються та шифруються так само, як інші повідомлення. Перший байт вказує на рівень тривоги та містить два значення, наприклад «1» означає попередження або «2» означає летальний результат. Якщо сповіщення є фатальним, посилання має бути перервано, і нове посилання не може бути встановлено на цьому конкретному сеансі через SSL. Другий байт вказує на ступінь серйозності, визначений кодом, пов'язаним з різними повідомленнями попередження. Сповіщення з відповідними кодами та типами наведені в таблиці 3.

Table 3. Alert messages of SSL

Codes	Alerts	Representations	Types
0	close_notify	No more messages on this link to receiver	Warning
10	unexpected_message	Inappropriate message to receiver	Fatal
20	bad_record_mac	Incorrect MAC record to receiver	Fatal
21	decryption_failed	Invalid decryption due to improper chunk size	Fatal
30	decompression_failure	Decompression fail due to improper input	Fatal
40	handshake_failure	Negotiation fail due to improper security parameters set	Fatal
41	no_certificate	Reply to no proper certificate is available	Warning
42	bad_certificate	Corrupted certificate or contains invalid signature	Warning
43	unsupported_certificate	Sender certificate is unsupported	Warning
44	certificate_revoked	Certificate was withdrawn by signer	Warning
45	certificate_expired	Issued certificate is no longer valid	Warning
46	certificate_unknown	An uncertain problem causes certificate to be inappropriate while handling	Warning
47	illegal_parameter	Security parameter are inconsistent w.r.t. their field in handshake	Fatal

Протокол рукостискання SSL (SSL Handshake Protocol)

Це перший протокол, який починає діяти після встановлення з'єднання за допомогою протоколу транспортного рівня. Клієнт і сервер перевіряють один одного й обмінюються необхідними параметрами безпеки, такими як набір шифрів, методи стиснення, випадкове число тощо, перш ніж надсилати дані програми один одному (див. рис. 3). Пакет протоколу рукостискання складається з трьох полів. «Тип» має справу з 1 байтом, який представляє тип пакета, «Довжина» в 3 байти вказує на довжину пакета, а «Вміст» (≥ 0 байтів) містить необхідні параметри безпеки, які потрібно встановити під час узгодження. Повідомлення рукостискання з відповідними кодами та параметрами безпеки перераховані в таблиці 4 і таблиці 5.

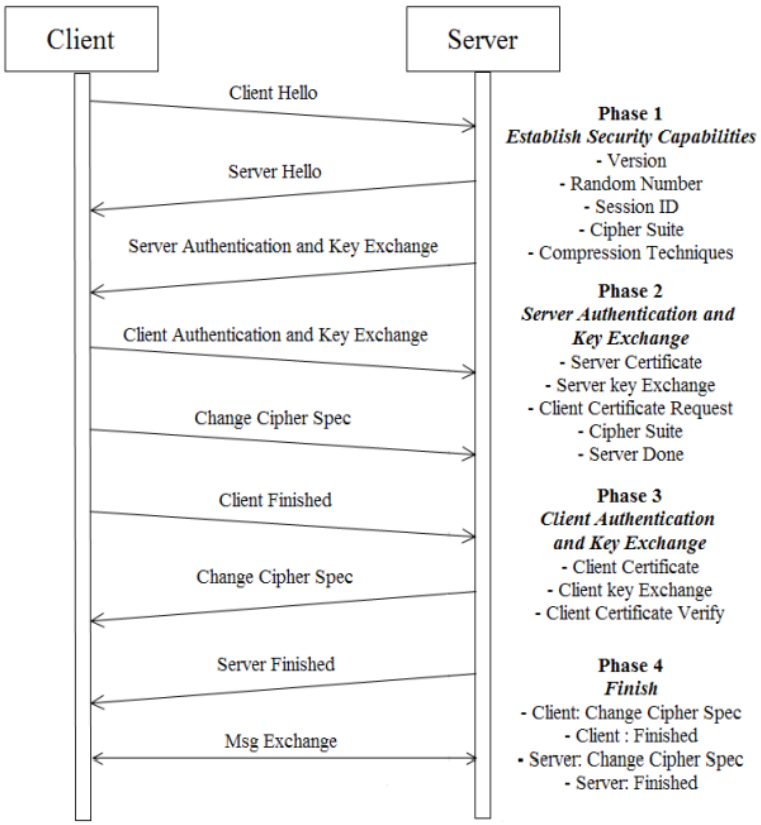


Fig 3: SSL/TLS handshaking protocol operation

Table 4. Handshake messages of SSL

Codes	Messages	Parameters
0	MT_hello_request	Void
1	MT_client_hello	version,random_no, session_id, cipher_suite, compression_tech
2	MT_sever_hello	version,random_no, session_id, cipher_suite, compression_tech
11	MT_certificate	X.509 certificates chain
12	MT_server_key_exchange	msg_signature, public_parameters
13	MT_certificate_request	cert_authorities, cert_type
14	MT_server_done	Void
	MT_client_key_exchange	msg_signature, public_parameters
15	MT_certificate_verify	cert_signature
20	MT_finished	MD5_hash + SHA_hash

Table 5. SSL cipher suite

Parameters	Values
Key exchange algorithms	RSA, Diffie-Hellman, Fortezza
Cipher algorithm	RC4, RC2, DES, 3DES or IDEA, Fortezza
MAC algorithm	MD5 or SHA
Cipher type	Stream or Block
MAC size	MD5(0 or 16 bytes) or SHA (20 bytes)
IV size	Initialization vector size used in CBC

CHR: MT_client_hello.random_no
SHR: MT_server_hello.random_no
SPM: secret_pre_master
SM: secret_master HSM: Handshake_messages upto current message CVSM: MT_certificate_verify.cert_signature.MD5_hash CVSS: MT_certificate_verify.cert_signature.SHA_hash KB: Key_block

Щоб підтримувати автентичність повідомлень обміну ключами сервера, підпис береться шляхом шифрування гешу за допомогою закритого ключа відправника. Загальнодоступні параметри містять інформацію про різні

криптографічні алгоритми, наведені в таблиці 6. SHA-1 використовується для створення підпису стандарту цифрового підпису (DSS), а MD5 і SHA-1 (36 байт) використовуються для підпису RSA.

Table 6. Algorithms and its parameters from server

Algorithms	Public Parameters
Ephemeral Diffie-Hellman	A prime no. and its primitive root
RSA	Public key (exponent and Modulo)

Обчислення гешу наведено нижче.

HASH (CHR+SHR+public_parameters) (2)

Після сертифіката сервера та обміну ключами він запитує сертифікат клієнта через повідомлення із запитом на сертифікат. У відповідь клієнт надсилає власний сертифікат, параметри обміну ключами та закінчується повідомленням *certificate_verify*. Параметри обміну ключами клієнта наведені в таблиці 7.

Table 7. Algorithms and parameters from client

Algorithms	Public Parameters
Ephemeral Diffie-Hellman	A prime no. and its primitive root
RSA	48 bits encrypted *secret_pre_master*

certificate_verify містить підпис сертифіката клієнта та обчислюється наступним чином.

CVSM= MD5 (SM+pad_2+MD5 (HSM+SM+pad_1)) (3)

CVSS=SHA(SM+pad_2+SHA (HSM+SM+pad_1)) (4)

Як згадувалося вище, SHA-1 використовується для DSS підпис, а для підпису RSA використовуються MD5 і SHA-1. Повідомлення рукостискання містять усі повідомлення від *MT_client_hello* до *MT_client_key_exchange*. Готове повідомлення підтверджує, що обмін ключами успішний чи ні в новому наборі шифрів, після чого одразу

надходить повідомлення про зміну специфікації шифру. Він обчислюється, як наведено нижче.

MD5 (SM+pad_2+MD5 (HSM+sender_id+SM+pad_1) +
SHA (SM+pad_2+SHA (HSM+sender_id+SM+pad_1) (5)

sender_id означає, чи є поточний відправник клієнтом чи сервером. Параметри обміну Diffie-Hellman використовуються для обчислення відповідних відкритих ключів, які обмінюються для обчислення SPM з обох сторін. Параметри обміну клієнтськими ключами RSA містять зашифрований SPM, який розшифровується ключем сервера для обчислення SM.

SM =MD5 (SPM+SHA („A“ +SPM+CHR+SHR)) +
MD5 (SPM+SHA („BB"+SPM+CHR+SHR)) +
MD5 (SPM+SHA („CCC"+SPM+CHR+SHR)) (6)

У випадку 3DES_ECE_CBC_SHA повідомлення про зміну специфікації шифру SSL вимагає *server_send_key* (168-бітний ключ + 24 біти керування), *client_send_key* (24 байти), *server_send_MACsecret* (20 байтів), *client_send_MACsecret* (20 байтів), *server_send_IV* (8 байтів), *client_send_IV* (8 байтів).), щоб змінити стан очікування на поточний стан. Таким чином, для цього потрібно, щоб із SM був згенерований ключовий блок розміром 104 байти, який містить наведені вище параметри в послідовному порядку і оцінюється таким чином.

KB= MD5 (SM+SHA („A"+SPM+CHR+SHR)) +
MD5 (SM +SHA („BB"+SPM+CHR+SHR)) +
MD5 (SM+SHA („CCC"+SPM+) CHR+SHR)) + […] (7)

Структура протокола TLS та підпротоколів з визначенням функцій

Оскільки TLS є оновленою версією SSL, він має ту саму архітектуру та протоколи, за винятком деяких змін у параметрах безпеки та обчисленні MAC, цифрового підпису та блоку ключа. Він також представляє деякі

нові сповіщення та псевдовипадкову функцію для посилення безпеки порівняно з SSL, описану нижче.

Протокол запису TLS

1. *Версія*: вказує на основну та проміжну версії, посилені клієнтом для поточного TLS. Основна та допоміжна версії для різних версій TLS перераховані в таблиці 8, що суперечить основній та допоміжній версіям для SSL 3 і 0 відповідно.

Table.8.Versions of TLS

Major Version	Minor Version	Class
3	1	TLS 1.0
3	2	TLS 2.0
3	3	TLS 3.0

2. *MAC*: для обчислення MAC використовується криптографічна хеш-функція зі спільним секретним ключем. Тут «версія стиснення» об'єднана з іншими полями, які є такими ж, як і поля послідовності SSL, показані в Eq.1.

   (MAC_secret_key, seq_no +TLS_compression_type +
   TLS_compression_version + TLS_compressed_chunk_length +
   TLS_compressed_chunk) *(8)*

3. *Псевдовипадкова функція (PRF)*: вона приймає спільний секрет, тег і дані як вхідні дані для PRF. Він обчислюється методом XOR двох хеш-значень (MD5 і SHA). shared_secret_left і shared_secret_right вказують на ліву половину та праву половину спільного секрету. Pseudo_MD5 і Pseudo_SHA викликаються три рази, щоб створити (3x 16 байт) і (3x 20 байт) остаточний результат розміром 48 байт. У випадку Pseudo_SHA з 60 байтів останні 12 байтів скорочуються, щоб отримати 48 байтів.

   PRF (shared_secret, tag, data) =
   (Pseudo_MD5 (shared_secret_left, tag +data)) XOR
   (Pseudo_SHA (shared_secret_right, tag+ data)) *(9)*

Тут псевдовипадкова функція викликається двічі функціями pseudo_MD5 і pseudo_SHA, де дані є конкатенація тегів і подача даних у PRF.

Pseudo_hash (key, data") =
HMAC_hash (key, pMAC (1) +data") +
HMAC_hash (key, pMAC (2) + data") +
HMAC_hash (key, pMAC (3) + data") + ... *(10)*

pMAC (0) = data" pMAC (k) = MAC_hash (ключ, pMAC(k-1)) *(11)*

HMAC_hash є *HMAC_MD5* для *Pseudo_MD5* і *HMAC_SHA* для *Pseudo_SHA*. І *pad_2* (512 біт), і *pad_1* (512 біт) містять двійкові значення X5C і X36 відповідно, які повторюються 64 рази.

HMAC_hash (key, data") = hash ((key" XOR pad_2) +
hash ((key" XOR pad_1) + data")) (12)

Протокол рукостискання TLS

1. *Набір шифрів*: алгоритми обміну ключами та шифрування, які підтримуються TLS, такі ж, як SSL, перелічені в таблиці V, за винятком Fortezza. Крім фіксованих і ефемерних алгоритмів Діффі-Хеллмана, він також підтримує Еліптичну криву Діффі-Хеллмана.
2. *Типи сертифікатів*. Відповідь на повідомлення із запитом на сертифікат, підписаний сертифікатом TLS, RSA або DSS, видається, якщо параметри обміну ключами є загальнодоступними параметрами RSA або Diffie-Hellman, переліченими в таблиці VI та VII.
3. *Повідомлення перевірки сертифіката*: воно містить підпис сертифіката клієнта, обчислений на основі попередніх повідомлень рукостискання шляхом їх хешування. Це усуває конкатенацію головного секрету та бітів заповнення з повідомленнями рукостискання перед обчисленням хешу, оскільки це не додасть додаткового захисту сертифікату.

MT_certificate_verify.cert_signature.MD5_hash =

MD5 (handshake_messages) *(13)*
MT_certificate_verify.cert_signature.SHA_hash =
MD5 (handshake_messages) *(14)*

4. *Готове повідомлення*: на відміну від SSL, хеш (MD5 і SHA) обчислюється на основі повідомлень рукостискання, а конкатенація їх передається як вхідні дані для PRF.

   PRF (secret_master, tag, MD5 (handshake_messages) +SHA (handshake_messages)) *(15)*

5. *Обчислення головного секрету та блоку ключів*: головний секрет обчислюється шляхом виклику функції PRF через три вхідні дані, такі як попередній головний секрет, тег і конкатенація клієнта і випадкове число сервера, яке набагато простіше, ніж SSL.

   secret_master = PRF (secret_pre_master,
   «master secret»,
   MT_client_hello.random_no +
   MT_server_hello.random_no) *(16)*

   Ключовий блок обчислюється за трьома вхідними даними, такими як головний секрет, тег і конкатенація випадкового no клієнта та сервера шляхом передачі PRF як параметрів .

   key_block = PRF (secret_master, "key expansion",
   MT_client_hello.random_no + MT_server_hello.random_no)
   (17)

6. *Паддинг (Padding)*: на відміну від SSL, перед шифруванням довільна довжина бітів заповнення, довжиною від 0 до 2^8 -1, об'єднується після MAC, щоб зробити розмір фрагмента кратним розміру блоку шифру. У SSL додається мінімальна кількість бітів заповнення, щоб зробити кратний блок шифру.

Протокол сповіщень TLS

У TLS додаткові повідомлення сповіщень вводяться, щоб зробити зв'язок надійним, перераховані в таблиці 9. Він підтримує всі сповіщення SSL, крім коду сповіщення 41.

Table 9. Alert messages of TLS

Codes	Alerts	Representations	Types
22	Record_overflow	Payload size exceeded more than $2^{14} + 2048$ bytes	Fatal
48	unknown_ca	CA certificate cannot be trusted or discovered	Fatal
49	accessed_ denied	Negotiation failed due to access control provided by receiver	Fatal
50	decode_ error	Information could not be decoded properly due to incorrect message length	Fatal
51	decrypt_ error	Unable to decrypt the secret key, verify digital signature or authenticity of finished message	Warning/ Fatal
60	export_ restriction	Negotiation against export restriction are detected and terminated	Fatal
70	protocol_ version	Protocol version is not supported by server	Fatal
71	insufficient_ security	Handshaking fail due to stronger cipher suite required by server	Fatal
80	internal_error	Error associated to local system and not related to SSL.	Fatal
90	User_cancelled	Abnormal termination of session by user	Fatal
100	no_renegotiation	Client or server response w.r.t hello request is not suitable for renegotiation	Warning

З'єднання блоків шифру

З'єднання блоків шифру є однією з операцій у режимі блочного шифру, яка використовується для шифрування серії фрагментів відкритого тексту. Для створення першого блоку зашифрованого тексту він отримує вектор ініціалізації (IV) з КВ, представлений у рівнянні 7. Послідовність

відкритого тексту від 2 до необхідних IV є попередньою частиною зашифрованого тексту відповідно (див. Малюнок 3).

Cipher (k) = E [key, (Plain (k) XOR cipher (k-1))]
Cipher (0) = IV *(18)*

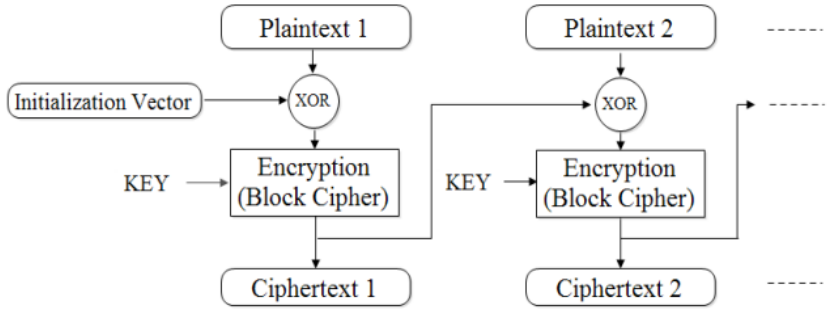


Fig 3: CBC for encryption

У розшифровці CBC операція XOR розміщується після зашифрованого тексту, розшифрованого ключем, де IV для першого відкритого тексту походить від KB.
Попередній зашифрований текст діє як IV для наступного звичайного тексту (див. рис. 4).

Plain (k) = D [k, Cipher (k)] XOR Cipher (k-1) (19)

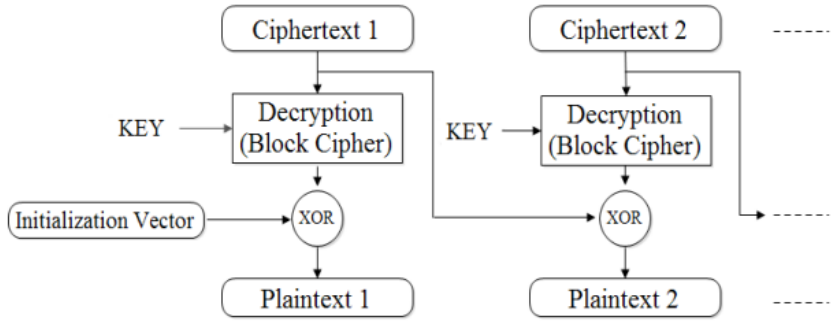


Fig.4. CBC for decryption

Типи атак

Одна з найбільших загроз для безпеки транспортного рівня через недоліки в SSL/TLS, який використовується для захисту зв’язку між відправником і одержувачем. Уразливості в SSL/TLS викликають як активні, так і пасивні атаки, такі як BEAST, CRIME, TIME, BREACH, LUCKY 13, RC4 BIASES,

SSL Renegotiation, POODLE, Truncation, Bar Mitzvah тощо. І їх виправлення наведено в таблиці 10.

Атака BEAST

Це скорочена форма Browser Exploit Against SSL/ TLS атаки, що відбувається за допомогою експлойту TLS 1.0 і була розроблена T. Duong і J. Riazo. Він використовує переваги симетричного шифрування та техніки ланцюжка блоків шифру (CBC), щоб вгадати секретний ключ, який використовується для шифрування відкритого тексту. У TLS 1.0 останній блок зашифрованого тексту є вектором ініціалізації для поточного відкритого тексту. Операція XOR між вектором ініціалізації та відкритим текстом шифрується симетричним ключем для отримання відповідного зашифрованого тексту. Якщо хакер може вгадати блок відкритого тексту, він може вгадати симетричний ключ і перевірити, чи збігається зашифрований текст чи ні. Це один тип атаки грубої сили, який фіксується відповідними TLS 1.1 і TLS 1.2.

Атака CRIME

Це коротка форма атаки Compression Ratio Leak Mass Exploitation, яка відбувається шляхом викрадення сеансу шляхом дешифрування файлів cookie сеансу в TLS 1.0 і розроблена J. Riazo і T. Duong. Він використовує переваги стиснення заголовків TLS і SPDY. SPDY — це відкритий мережевий протокол, який керує HTTP-трафіком, розроблений Google. І методи стиснення TLS, і SPDY використовують алгоритм DEFLATE, який усуває повторюваний рядок шляхом стиснення та шифрує його. Ключ отримується шляхом обману браузера та надсилання зашифрованого стисненого запиту на справжній веб-сайт, очікування розміру відповіді HTTP та посилення атаки щодо відповідей HTTP. Хакер повторює прийоми з різними значеннями, поки не буде отриманий ключ. Це один тип атаки грубої сили, виправлений шляхом вимкнення механізму стиснення в TLS 1.1 і TLS 1.2.

Атака за часом

Атака Timing Info-Leak Made Easy (TIME), за допомогою якої зловмисник витягує секретну інформацію, не підслуховуючи мережу, була розроблена

Т. Беері та А. Шульманом з Imperva. Щоб здійснити цю атаку, хакер хоче знати розташування файлів cookie, префікс/суфікс і місце для вставлення відкритого тексту. Інформація про сеансові файли cookie отримується за часом, необхідним для отримання відповіді від сервера/одержувача. Через перешкоди в мережі один процес повторюватиметься певну цілісну кількість часу, і мінімальний час відповіді береться як остаточний час відповіді для цього конкретного запиту. Припустимо, що дані клієнта містять «секретний елемент = невідомі дані», який є корисним навантаженням і секретним елементом, і його значення відображається у відповіді. У першій ітерації для довільного введення користувачем розмір відповіді становить 1028 байт. Якщо на другій ітерації користувач вводить «секретний елемент = a», а розмір відповіді становить 1008 байт. Тому на це потрібно менше часу, ніж на першу ітерацію. З кількома запитами обчислюється найкоротший час відповіді для кожного символу для кожної позиції в корисному навантаженні, що є правильним припущенням і конкретним значенням секретного елемента.

BREACH атака

Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext – це злочинна атака проти органу реагування, і її розробили А. Прадо, Н. Харріс і Ю. Глюк. Зловмисник використовує техніку стиснення HTTP (алгоритм LZ77), вгадуючи символи та символи без зниження або підробки SSL, щоб запустити цю атаку, і його припущення буде відображено в тілі відповіді. Для досить стабільних сторінок знадобилося менше 30 секунд, щоб отримати секрет, як-от токен CSRF, стан перегляду тощо... Він вразливий до будь-якої версії SSL або TLS. Щоб розпочати атаку злому, і зловмисник, і жертва мають бути в одній мережі. У командно-контрольному центрі є драйвер веб-сервера, який називається iframe streamer, який збирається впроваджувати HTTP-запит у жертву, слухач зворотного виклику, робота якого полягає у виконанні зворотного виклику, коли жертва отримує відповідь, і монітор трафіку спостерігає за довжиною зашифрованого тексту, що повертається. Основна логіка оракула - це набір алгоритмів, які використовуються для вгадування секретів. Для боротьби з кодуванням Хаффмана використовується пул набору символів плюс випадкове доповнення, а для боротьби з блоковим шифром використовується віконна техніка. Це одна з найбільш вразливих атак на SSL, яку ще потрібно виправити.

Атака LUCKY 13.

Це одна з найбільш вразливих атак у SSL на сьогоднішній день. Її розробили N. A. Fardan і K. Paterson з Royal Holloway Лондонського університету в лютому 2013 року. Вона використовує техніку padding oracle – це атака на боковий канал, на яку впливає на заповнення шифрованого тексту. Зловмисник використовує ланцюжок блоків шифрування TLS, замінюючи кілька останніх байтів вибраними байтами та спостерігаючи за часом, потрібним серверу для відповіді. Обробка пакетів TLS, які містять справжнє заповнення, потребує менше часу. Якщо TLS генерує транзакцію як помилку, вона створює повідомлення з помилками, яке допомагає зловмиснику надсилати зловмисні пакети в новому сеансі, багаторазово підтверджуючи кожну попередню помилку. Результат показує, що 2^{23} сеанси потребують вилучення інформації про файли cookie та 2^{19}сеансів, якщо використовується 64-бітна схема кодування TLS. Загалом атака LUCKY 13 вимагає 2^{13}сеансів; якщо відомий байт інформації щодо тегу MAC або заповнення.

Атака RC4 BIASES

Вона також відома як ARC4 або ARCFOUR атака, виявлена Альфарданом, Бернштейном, Патерсоном, Поеттерінгом і Шульдтом, яка використовує всі версії SSL/ TLS. Для шифрування корисного навантаження використовується алгоритм шифрування RC4-128. Він використовує 128 ключів і генерує рядок випадкових ключів. Ці ключі об'єднуються з різними блоками відкритих текстів для отримання блоку зашифрованих текстів. Проблема полягає в тому, що випадкові ключі, згенеровані RC4, не зовсім випадкові, що допомагає відновити частину відкритого тексту при великій кількості TLS-шифрувань. Якщо одне і те ж повідомлення зашифрувати різними ключами RC4, то будуть згенеровані випадкові тексти шифрів. Оскільки ключі не зовсім випадкові або мають невеликі зсуви, то і шифротексти будуть не зовсім випадковими або матимуть дуже малі зсуви. Зловмисники підраховують ці відхилення від випадковості, проводячи статистичний аналіз окремих місць шифрованих текстів. Експериментальні результати показують, що приблизно 2^{32} шифротексти дають майже всі відкриті тексти. Для вилучення відкритих текстів із зашифрованих текстів потрібно близько 2^{30}сеансів.

SSL Renegotiation Атака

Відбувається за допомогою експлойту SSL 3.0 і всіх версій TLS і була виявлена M. Ray і S. Dispensa в серпні 2009 року. Зловмисник перехоплює HTTP-з'єднання для додавання відкритого тексту в перетворення. Він не розшифровує зв'язок між клієнтом і сервером. зв'язок між клієнтом і сервером. Під час безпечної онлайн-транзакції клієнт ініціює процес рукостискання SSL. Хакер блокує запит і перехоплює ці пакети. Потім він ініціює нову сесію і і завершує процес рукостискання. Після завершення зловмисник просить сервер зарахувати гроші на його рахунок під час банківської транзакції. гроші на його рахунок під час банківської транзакції. Сервер просить про повторне узгодження. Ці блокові пакети жертви будуть відправляються на сервер, який буде виконувати нове SSL-рукостискання поверх сесію, яка була встановлена раніше. Двох сесій достатньо для проведення ланч-атаки на жертву. Це можна виправити або за допомогою відключити повторне узгодження на стороні сервера або клієнт-сервер повинен клієнт-сервер повинен перевіряти попередні рукостискання.

Атака POODLE

Атака Padding Oracle On Downgraded Legacy Encryption - це однією з атак типу "людина посередині", де зловмисник використовує уразливості SSL 3.0 для розшифрування HTTP cookies. Вона була була виявлена B. Moller, T. Doung та K. Kotowicz 14 жовтня 2014 року. жовтня 2014 року. Зловмисник знаходиться між клієнтом та сервером намагається знизити версію TLS v1.0 або останню версію хендшейку між ними для безпечної передачі на SSL v3.0. Padding в SSL v3.0 використовується техніка, яка є випадковою за своєю природою, тобто додавання від 1 до L байт, які не є детермінованими для отримання цілого числа фрагментів для виконання операції ланцюжка блоків шифрування для виконання операції ланцюгового шифрування. Ці байти не покриваються MAC і не перевіряються при розшифровці. Останній байт заповнення вказує на кількість використаних байт, що допомагає хакеру запустити атаку. Зловмисник копіює проміжний байт(и) до проміжні байти в останні байти і намагається їх використати. Якщо модифікований останній байт збігається

з попереднім з попереднім байтом, то після розшифрування буде отримано правильну кількість байт заповнення буде додано, не впливаючи на байти MAC-адреси. Тепер повідомлення буде прийнято сервером, що буде що допоможе хакеру відновити відкритий текст байт за байтом, але по одному байт за раз, виконавши операцію XOR. 1 з 256 разів повідомлення буде прийнято; в гіршому випадку 255 разів з з 256 результатів буде видано повідомлення про помилку і сеанс буде перервано, але в але в останньому випадку все буде нормально.

FREAK-атака

Факторизація ключів експорту RSA (FREAK) - це одна з TLS-уразливостей, знайдена в декількох відомих браузерах (наприклад, Safari, Android браузер, Cisco, Opera). Її також називають атакою підміни сервера проти браузерів. Атака націлена на групу слабких наборів експортних шифрів, що використовуються в TLS. Ці пакети алгоритмів реалізовані в декількох клієнтських бібліотеках TLS, таких як Open SSL, Boring SSL, LibReSSL, IBM JSSE, SChannel тощо. Реалізація вищевказаних бібліотек в браузері некоректно використовує експортний набір шифрів, навіть якщо між сервером і клієнтом для обміну інформацією узгоджено використання неекспортного набору шифрів. Узгодження набору експортних шифрів між сервером та клієнтом дозволяє зловмиснику обманом змусити браузер клієнта використовувати слабкий експортний ключ шляхом проведення MITM-атаки. Атака FREAK знижує стійкість шифру, який використовує алгоритм обміну ключами RSA, де розмір ключа менший за 512 біт. Таким чином, факторизація займе менше 12 годин. Як і FREAK, Logjam уразливість в SSL/TLS дозволяє зловмиснику понизити версію експортного набору шифрів, що використовує алгоритм обміну ключами Diffie-Hellman. Цьому можна запобігти, відключивши набір експортних шифрів у браузерах.

Атака Bar Mitzvah

Експлойт Алгоритм парового шифру RC4, що підтримується SSL/TLS, допомагає отримувати інформацію через зашифрований зв’язок. Зловмисник намагається отримати слабкі ключі, націлившись на перші 100 байтів зашифрованої інформації, з яких 36 байтів належать до готового

повідомлення SSL/TLS. Оскільки готове повідомлення містить найбільш передбачувану інформацію, звичайні готові повідомлення об’єднуються XOR із зашифрованими готовими повідомленнями, щоб отримати частину послідовностей генератора псевдовипадкових чисел (PRNGS). Після видалення PRNGS, які не відповідають шаблону слабких ключів, згенерованих PRNGS, усі ключі вибраних PRNGS використовуються для розшифровки зашифрованого тексту, захопленого зловмисником за допомогою алгоритму RC4. Ключі з імовірністю 0,5 успішно визначені, що мінімізує кількість спроб, виконаних атакою грубої сили, як різницю 211,2. Ця атака не може витягнути повні відкриті тексти із зашифрованих текстів. 3.11 Атака на усікання TLS Аномальне завершення TLS-з’єднання, яке виконує зловмисник, щоб зберегти сеанс жертви за допомогою підключення кількох браузерів . Він був розроблений Б. Смітом і А. Піронті в липні 2013 року. Щоб підвищити продуктивність, веб-браузер завантажує вміст через кілька підключень. Оскільки TLS забезпечує цілісність і конфіденційність через одне з’єднання, кілька підключень браузера клієнта до одного сервера впорядковуються через одне з’єднання TLS. Перш ніж здійснити цю атаку, зловмисник має повний контроль над мережею, що допомагає вставляти/скидати пакети в різні з’єднання. Він запускається під час запиту клієнта на вихід із системи шляхом введення повідомлення TCP FIN або RESET для цього з’єднання до того, як повідомлення запиту стає недоступним для сервера через ненормальне завершення з’єднання. Оскільки підтвердження виходу з системи надходить до запиту на вихід із системи, отриманого сервером, зловмисник запускає цю атаку, щоб підтримувати сеанс без відома жертви. Нарешті, інше підключення браузера використовується для доступу до облікового запису жертви та його зміни.

Table 10. Attacks and their fixes

Attacks	Fixes
BEAST	Use RC4, 3DES, AES 256
CRIME	Disable TLS compression
TIME	Encrypt then MAC, use AES-GCM ciphers
LUCKY 13	Add random time delays, use authenticated encryption, use RC4
BREACH	Disable HTTP compression
RC4 BIASES	Disable RC4 in SSL/TLS
SSL Renegotiation	Client and server verify previous hand shake
POODLE	disable SSL 3.0 in web browser
FREAK	Configure SSL/TLS with higher version of cipher
Bar Mitzvah	Disable RC4 in SSL/TLS
TLS Truncation	Centralized authentication and chain sign outs

Порівняльний аналіз версій протоколів SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1 та TLS 1.2.

SSL 1.0/2.0/3.0

Цей протокол був випущений в 1996 році, але його історія почалася зі створення SSL 1.0, розробленого компанією Netscape. Версія 1.0 не була випущена, а версія 2.0 мала ряд недоліків у безпеці, що призвело до випуску SSL 3.0. Деякі основні поліпшення SSL 3.0 в порівнянні з SSL 2.0 наступні:

- Відокремлення транспортування даних від рівня повідомлень
- Використання повних 128 біт ключового матеріалу, навіть при використанні шифру "Експорт
- Можливість клієнта і сервера надсилати ланцюжки сертифікатів, що дозволяє організаціям використовувати ієрархію сертифікатів глибиною більше двох сертифікатів.
- Реалізація узагальненого протоколу обміну ключами, що дозволяє обмінюватися ключами Diffie-Hellman і Fortezza, а також не-RSA сертифікатами.
- Дозволяє стискати та розпаковувати записи
- Можливість повернутися до SSL 2.0, коли зустрічається клієнт 2.0

TLS 1.0

Цей протокол був вперше описаний в RFC 2246 в січні 1999 року. Це було оновлення в порівнянні з SSL 3.0, і відмінності не були кардинальними, але вони досить значні, щоб SSL 3.0 і TLS 1.0 не були сумісними. Ось деякі з основних відмінностей між SSL 3.0 і TLS 1.0:

Функції виведення ключів відрізняються

- MAC-адреси відрізняються - SSL 3.0 використовує модифікацію раннього HMAC, тоді як TLS 1.0 використовує HMAC.
- Готові повідомлення відрізняються
- TLS має більше сповіщень
- TLS вимагає підтримки DSS/DH

TLS 1.1

Цей протокол був визначений в RFC 4346 в квітні 2006 року і є оновленням TLS 1.0. Основні зміни наступні:

- Неявний вектор ініціалізації (IV) замінено на явний IV для захисту від атак Cipher block chaining (CBC).
- Обробка помилок з доповненням змінена на використання попередження bad_record_mac замість попередження decryption_failed для захисту від атак CBC.
- Реєстри IANA визначені для параметрів протоколу
- Передчасне закриття більше не призводить до неможливості відновлення сеансу.

TLS 1.2

Цей протокол було визначено в RFC 5246 у серпні 2008 року. Заснований на TLS 1.1, TLS 1.2 містить покращену гнучкість. Основні відмінності полягають в наступному:

- Комбінація MD5/SHA-1 в псевдовипадковій функції (PRF) була замінена на псевдовипадкові функції, визначені набором шифрів.
- Комбінація MD5/SHA-1 в елементі з цифровим підписом замінена на єдиний хеш. Підписані елементи включають поле, яке явно вказує на використаний хеш-алгоритм.
- Було суттєво очищено можливість клієнта та сервера вказувати, які алгоритми хешування та підпису вони прийматимуть.
- Додано підтримку автентифікованого шифрування з додатковими режимами передачі даних.
- Об'єднано визначення розширень TLS та наборів шифрів AES.
- Посилена перевірка номерів версій EncryptedPreMasterSecret.
- Було посилено багато вимог
- Довжина verify_data залежить від набору шифрів
- Покращено опис захисту від атак Блейхенбахера/Дліми.

TLS 1.3

Цей протокол наразі перебуває на стадії доопрацювання і є 28-ю редакцією. Основні відмінності від TLS 1.2 включають

- Список підтримуваних симетричних алгоритмів був очищений від усіх застарілих алгоритмів. Всі алгоритми, що залишилися, використовують алгоритми автентифікованого шифрування з асоційованими даними (AEAD).
- Додано режим з нульовим RTT (0-RTT), який дозволяє заощаджувати цикл при встановленні з'єднання для деяких даних додатків, але за рахунок певних властивостей безпеки.
- Видалено статичні набори шифрів RSA та Diffie-Hellman; всі механізми обміну ключами на основі відкритих ключів тепер забезпечують пряму секретність.
- Всі повідомлення рукостискання після ServerHello тепер зашифровані.
- Функції отримання ключів були перероблені, а функція вилучення та розширення ключів (HKDF) на основі HMAC використовується як примітив.
- Реструктуризовано машину станів рукостискання, щоб зробити її більш послідовною і видалити зайві повідомлення.

- ECC тепер входить до базової специфікації і включає нові алгоритми підпису. Узгодження формату точки було вилучено на користь формату однієї точки для кожної кривої.
- Видалено стиснення, кастомні групи DHE та DSA, замість RSA тепер використовується PSS.
- Механізм перевірки узгодження версій TLS 1.2 застарів на користь списку версій у розширенні.
- Відновлення сесії зі станом на стороні сервера і без нього, а також шифропакети на основі PSK попередніх версій TLS були замінені єдиним новим обміном PSK.

ВИСНОВОК

SSL/TLS, два ізольовані протоколи, використовуються для захисту каналів зв’язку між двома сторонами шляхом надання двох рівнів безпеки, таких як автентифікація та шифрування даних користувача. Логічна або операційна помилка в цих протоколах дає можливість зловмиснику використати її. У цьому документі описано архітектуру та робочий процес цих протоколів, а також узагальнено різні типи атак і способи їх вирішення. Нарешті потрібно провести більше досліджень у цій галузі, щоб підвищити ступінь безпеки SSL/TLS шляхом зменшення кількості помилок.