ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до магістерської роботи

на тему:

«ТЕХНОЛОГІЯ ЗАХИЩЕНОГО ОБМІНУ ДАНИМИ НА БАЗІ ПРОТОКОЛУ SSL/TLS»

Виконала студентка 6 курсу, групи БСДМ-61
спеціальності 125 Кібербезпека
освітньо-професійної програми «Інформаційна та кібернетична безпеки»
(шифр і назва спеціальності)

Скупейко С.В.
(прізвище та ініціали)

Керівник к.т.н., Борсуковський Ю.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтролер _____
(прізвище та ініціали)

КИЇВ-2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Інститут	ННІЗІ
Кафедра	Інформаційної та кібернетичної безпеки
Ступінь вищої освіти	Магістр
Спеціальність	125 Кібербезпека
Освітньо-професійна програма	Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ

Завідувач кафедри ІКБ

_____ Гайдур Г.І.

“___”_____ 2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Скупейко Софії Володимирівні

(прізвище, ім’я, по батькові)

1. Тема магістерської роботи: «Технологія захищеного обміну даними на базі протоколу SSL/TLS»

керівник магістерської робот: Борсуковський Юрій Володимирович, к.т.н., доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «12» жовтня 2022 року №122.

2. Строк подання студентом магістерської роботи 15.12.2022 р.

3. Вихідні дані до магістерської роботи

корпоративна інформаційна система;

програмні комплекси управління захистом кінцевих точок;

наукова та технічна література, експлуатаційна документація, нормативні документи, міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Актуальність проблеми захищеного обміну даними.

2. Загальні відомості про роботу протоколу захищеного обміну даними SSL/TLS

3. Вимоги міжнародних стандартів до використання SSL/TLS.

4. Вразливості протоколів SSL/TLS. Найбільш відомі атаки.

5. Перелік графічного матеріалу

1. Тема магістерської роботи.

2. Об’єкт, предмет, мета та наукові завдання дослідження.

3. Результати аналізу складу та умов функціонування корпоративної інформаційної системи.

4. Результати аналізу методів та засобів захищеного обміну даними.

5. Історія створення і розвитку протоколів SSL/TLS.

6. Нормативно-правове регулювання використання SSL/TLS.

7. Порівняння протоколів SSL/TLS.

8. Організація захисту обміну даними на основі актуальних версій TLS.

9. Висновки за результатами роботи.

6. Дата видачі завдання 26.09.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	26.09.2022	
2.	Збір та аналіз літератури.	03.10.2022	
3.	Написання розділу 1.	17.10.2022	
4.	Написання розділу 2.	31.10.2022	
5.	Написання розділу 3.	14.11.2022	
6.	Формулювання висновків за результатами проведеного дослідження	01.12.2022	
7.	Оформлення роботи	15.12.2022	

Студент _____ Скупейко С.В.

(підпис) (прізвище та ініціали)

Керівник магістерської роботи _____ Борсуковський Ю.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота присвячена дослідженню технології захищеного обміну даними, протоколів SSL/TLS. Робота складається зі вступу, трьох розділів, що містять 9 рисунків та 1 таблицю, висновків та списку використаних джерел, що містить 15 найменувань. Загальний обсяг роботи становить 61 аркушів, з яких 2 аркуша займає список використаних джерел.

Об’єкт дослідження – технологія захищеного обміну даними на базі протоколу SSL/TLS.

Предмет дослідження – вразливості протоколів SSL/TLS.

Мета роботи – визначення актуальних вразливостей протоколів SSL/TLS та заходів протидії.

Для досягнення кінцевої мети в роботі розглянуто місце SSL/TLS серед мережевих протоколів, історію виникнення і розвитку даних протоколів, загальні принципи роботи і значення протоколів SSL/TLS в організації захищеної мережевої комунікації. Також розглянуто власне різницю між SSL і TLS та суть поняття протокол SSL/TLS, використовуваного на даний момент. Наведено нормативно-правові документи, що регламентують шифрування мережевих комунікацій з використанням SSL/TLS. Як результат дослідження наведено найбільш відомі вразливості SSL/TLS, їх опис, методи протидії та зроблено висновки щодо актуальності тих чи інших вразливостей.

Галузь використання. Висновки по актуальним вразливостям можуть бути використані в будь якому випадку для врахування загроз при побудові захищеної мережевої комунікації. Супутня інформація, наведена в роботі, дозволяє ознайомитись з принципами роботи і використання SSL/TLS, а також з вимогами до побудови захищеної мережевої комунікації нормативно-правових документів і міжнародних стандартів.

Ключові слова: SSL/TLS, ІНФОРМАЦІЙНА БЕЗПЕКА, ОБМІН ДАНИМИ, ЗАХИЩЕНА МЕРЕЖЕВА КОМУНІКАЦІЯ, ВРАЗЛИВОСТІ SSL/TLS, ПОШИРЕНІ СТАНДАРТИ.

ЗМІСТ

ВСТУП

А к т у а л ь н і с т ь т е м и. На сьогоднішній день обмін інформацією здійснюється переважно через Інтернет – від особистого листування до обміну конфіденційними даними всередині або між організаціями.

Ледь не вся інформація, якою обмінюються через Інтернет, вже захищена за допомогою технології шифрування. Наприклад, онлайн-магазини та Інтернет-банкінг не працювали б без хорошого шифрування. Шифрування призначене для захисту коштів та особистої інформації. У корпоративному середовищі шифрування слід використовувати для захисту інтелектуальної власності та інноваційних розробок компанії, а також інших конфіденційних даних.

Багато підприємств вважають, що вони не уразливі до кібератак чи порушень захисту даних через їх невеликий розмір та обмежені активи. Однак за даними дослідження IDC, малий та середній бізнес стає жертвою у більш ніж 70% випадках порушень безпеки.

Тому шифрування є необхідним на сьогоднішній день для будь яких мережевих комунікацій. Одними з найчастіше використовуваних для шифрування протоколів є протоколи SSL/TLS.

SSL (Secure Sockets Layer) та TLS (Transport Level Security) — криптографічні протоколи, що забезпечують захищену передачу даних у комп'ютерній мережі. Вони широко використовуються у веб-браузерах, а також під час роботи з електронною поштою, обміну миттєвими повідомленнями та в IP-телефонії.

Важливо розуміти, що дані протоколи в історії свого розвитку стикались з численними проблемами безпеки і відповідно оновлювались для запобігання викриттю інформації, що передається.

Деякі вразливості є неактуальними при використанні останніх оновлених версій протоколів SSL/TLS, тоді як інші активно використовуються. Розуміння

актуальних загроз є вкрай необхідним для побудови захищеної мережевої комунікації з використанням протоколів SSL/TLS.

Мета і завдання дослідження. **Мета роботи** полягає в аналізі і визначенні актуальних вразливостей протоколів SSL/TLS та заходів протидії.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Дослідити історія розробки і розвитку протоколів SSL/TLS та основних принципів їх функціонування.
2. Визначити роль протоколів SSL/TLS в побудові захищених мережевих комунікацій і вимоги міжнародних стандартів до шифрування даних з використанням SSL/TLS.
3. Дослідити вразливості протоколів SSL/TLS, виявлені протягом історії їх застосування і визначити актуальні на даний момент вразливості та атаки, методи протидії.

Об’єкт дослідження – технологія захищеного обміну даними на базі протоколу SSL/TLS. **Предмет дослідження** – вразливості протоколів SSL/TLS.

Наукова новизна одержаних результатів. Висновки щодо актуальних вразливостей і атак на протоколи SSL/TLS можуть буди використані при забезпеченні захищеного обміну даними з використанням протоколів SSL/TLS. В роботі систематизовано інформацію щодо вразливостей, а також зроблено висновки щодо того, які вразливості і атаки є актуальними і які заходи мають бути впроваджені для захисту мережевих комунікації з використанням SSL/TLS.

Практичне значення одержаних результатів. Висновки щодо актуальних вразливостей і атак на протоколи SSL/TLS можуть бути використані будь якою організацією або суб’єктом для врахування при побудові захищених мережевих комунікацій, при визначенні найбільш безпечних налаштувань використання протоколів SSL/TLS.

Розділ 1

ЗАГАЛЬНІ ВІДОМОСТІ ПРО РОБОТУ ПРОТОКОЛУ ЗАХИЩЕНОГО ОБМІНУ ДАНИМИ SSL/TLS

1.1. Протоколи захисту мережі

Протоколи безпеки мережі — це мережеві протоколи, які забезпечують цілісність і безпеку даних, що передаються через мережеві з'єднання. Використовуваний конкретний протокол безпеки мережі залежить від типу захищених даних і мережевого підключення. Кожен протокол визначає методи та процедури, необхідні для захисту мережевих даних від несанкціонованих або зловмисних спроб прочитати або викрасти інформацію.

1.1.1. Модель OSI.

Взаємозв'язок відкритих систем (OSI) є еталонною моделлю того, як додатки спілкуються через мережі. Він показує, як кожен рівень зв'язку будується поверх іншого, від фізичної проводки до програм, які намагаються спілкуватися з іншими пристроями через мережу.

OSI є еталонною моделлю, яка спрямовує постачальників технологій щодо проектування сумісного програмного та апаратного забезпечення, забезпечуючи чітку структуру, яка описує можливості мережі або системи зв'язку. Командам із безпеки модель OSI допомагає зрозуміти, які рівні мережі їм потрібно захищати, де можуть виникнути конкретні загрози безпеці та як їх запобігти та пом'якшити [1].

Модель OSI містить такі рівні [1]:

Рівень 1 — фізичний рівень—фізичний кабель або бездротове з'єднання між вузлами мережі.

Рівень 2 — канальний рівень — створює та припиняє з'єднання, розбиває пакети на кадри та передає їх від джерела до пункту призначення.

Рівень 3 — мережевий рівень — розбиває сегменти на мережеві пакети та збирає їх після отримання, а також направляє пакети за оптимальним шляхом у фізичній мережі.

Рівень 4 — транспортний рівень — відповідає за повторне збирання сегментів на приймальному кінці, перетворюючи їх на дані, які можуть використовуватися на рівні сеансу.

Рівень 5 — сеансовий рівень — створює канали зв'язку, які називаються сеансами, між пристроями. Зберігає сесії відкритими під час передачі даних і закриває їх, коли вона закінчується.

Рівень 6 — презентаційний рівень — готує дані для прикладного рівня, визначаючи, як два пристрої мають кодувати, шифрувати та стискати дані, щоб забезпечити їх правильне отримання.

Рівень 7 — прикладний рівень — використовується програмним забезпеченням для кінцевих користувачів, таким як веб-браузери та клієнти електронної пошти. Надсилає та отримує інформацію, значущу для кінцевих користувачів, використовуючи такі протоколи, як HTTP, FTP і DNS.

Зображення моделі OSI представлено на Рис. 1.1 [2]:

Модель OSI

Дані	7 прикладний application	Доступ до мережевих служб
	6 представлень presentation	Представлення і кодування даних
	5 сеансовий session	Управління сеансом зв'язку
Сегменти	4 транспортний transport	Прямий зв'язок між кінцевими пунктами і надійність
Пакети	3 мережевий network	Визначення маршруту і логічна адресація
Кадри	2 канальний data link	Фізична адресація
Біти	1 фізичний physical	Робота з середовищем передачі , сигналами і двійковими даними

Рис. 1.1. Модель OSI

(7) Прикладний рівень (рівень додатків; Application layer) - верхній рівень моделі, що забезпечує взаємодію користувача додатків з мережею:

- дозволяє додаткам використовувати мережеві служби:
  - віддалений доступ до файлів і баз даних,
  - пересилання електронної пошти;
- відповідає за передачу службової інформації;
- надає додаткам інформацію про помилки;
- формує запити до рівня уявлення.

Протоколи прикладного рівня: RDP, HTTP, SMTP, SNMP, POP3, FTP, XMPP, OSCAR, Modbus, SIP, TELNET і інші [2].

(6) Представницький рівень

Представницький рівень (Presentation layer) забезпечує перетворення протоколів і кодування/декодування даних. Запити програм, наданих прикладного рівня, на рівні уявлення перетворюються в формат для передачі по мережі, а отримані з мережі дані перетворюються в формат додатків. На цьому рівні може здійснюватися стиснення/розпакування або шифрування/дешифрування, а також перенаправлення запитів іншому мережному ресурсу, якщо вони не можуть бути оброблені локально.

Рівень представлень зазвичай являє собою проміжний протокол для перетворення інформації з сусідніх рівнів. Це дозволяє здійснювати обмін між додатками на різнорідних комп'ютерних системах прозорим для додатків чином. Рівень уявлень забезпечує форматування і перетворення коду. Форматування коду використовується для того, щоб гарантувати додатком надходження інформації для обробки, яка мала б для нього сенс. При необхідності цей рівень може виконувати переклад з одного формату даних в інший.

Рівень уявлень має справу не тільки з форматами та поданням даних, він також займається структурами даних, які використовуються програмами. Таким чином, рівень 6 забезпечує організацію даних при їх пересилці.

Щоб зрозуміти, як це працює, уявімо, що є дві системи. Одна використовує для представлення даних розширений двійковий код обміну інформацією EBCDIC, наприклад, це може бути мейнфрейм компанії IBM, а інша - американський стандартний код обміну інформацією ASCII (його використовує більшість інших виробників комп'ютерів). Якщо цим двом системам необхідно обмінятися інформацією, то потрібен рівень уявлень, який виконає перетворення і здійснить переказ між двома різними форматами.

Інший функцією, виконуваної на рівні представлень, є шифрування даних, яке застосовується в тих випадках, коли необхідно захистити передану інформацію від доступу несанкціонованими одержувачами. Щоб вирішити це завдання, процеси і коди, що знаходяться на рівні уявлень, повинні виконати перетворення даних. На цьому рівні існують і інші підпрограми, які стискають тексти і перетворюють графічні зображення в бітові потоки, так, що вони можуть передаватися по мережі.

Стандарти рівня представлень також визначають способи представлення графічних зображень. Для цих цілей може використовуватися формат PICT - формат зображень, застосовуваний для передачі графіки QuickDraw між програмами.

Іншим форматом представлень є тегірованний формат файлів зображень TIFF, який зазвичай використовується для растрових зображень з високою роздільною здатністю. Наступним стандартом рівня уявлень, який може використовуватися для графічних зображень, є стандарт, розроблений Об'єднаної експертною групою по фотографії (Joint Photographic Expert Group); в повсякденному користуванні цей стандарт називають просто JPEG.

Існує інша група стандартів рівня вистав, яка визначає представлення звуку і кінофрагментів. Сюди входять інтерфейс електронних музичних інструментів (Musical Instrument Digital Interface, MIDI) для цифрового представлення музики, розроблений Експертною групою з питань кінематографії стандарт MPEG, який

використовується для стиснення і кодування відеороликів на компакт-дисках, зберігання в оцифрованному вигляді і передачі зі швидкостями до 1,5 Мбіт/с, і QuickTime - стандарт, що описує звукові та відео елементи для програм, які виконуються на комп'ютерах Macintosh і PowerPC.

Протоколи рівня представлень: AFP - Apple Filing Protocol, ICA - Independent Computing Architecture, LPP - Lightweight Presentation Protocol, NCP - NetWare Core Protocol, NDR - Network Data Representation, XDR - eXternal Data Representation, X.25 PAD - Packet Assembler/Disassembler Protocol [2].

(5) Сеансовий рівень

Сеансовий рівень (Session layer) моделі забезпечує підтримку сеансу зв'язку, дозволяючи додаткам взаємодіяти між собою тривалий час. Рівень управляє створенням/завершенням сеансу, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних і підтримкою сеансу в періоди неактивності додатків.

Протоколи сеансового рівня: ADSP (AppleTalk Data Stream Protocol), ASP (AppleTalk Session Protocol), H.245 (Call Control Protocol for Multimedia Communication), ISO-SP (OSI Session Layer Protocol (X.225, ISO 8327)), iSNS (Internet Storage Name Service), L2F (Layer 2 Forwarding Protocol), L2TP (Layer 2 Tunneling Protocol), NetBIOS (Network Basic Input Output System), PAP (Password Authentication Protocol), PPTP (Point-to-Point Tunneling Protocol), RPC (Remote Procedure Call Protocol), RTCP (Real-time Transport Control Protocol), SMPP (Short Message Peer-to-Peer), SCP (Session Control Protocol), ZIP (Zone Information Protocol), SDP (Sockets Direct Protocol) [2].

(4) Транспортний рівень

Транспортний рівень (Transport layer) моделі призначений для забезпечення надійної передачі даних від відправника до одержувача. При цьому рівень надійності може варіюватися в широких межах. Існує велика кількість класів протоколів транспортного рівня, починаючи від протоколів, які надають тільки

основні транспортні функції (наприклад, функції передачі даних без підтвердження прийому), і закінчуючи протоколами, які гарантують доставку в пункт призначення кількох пакетів даних в належній послідовності, мультиплексує кілька потоків даних, забезпечують механізм управління потоками даних і гарантують достовірність отриманих даних. Наприклад, UDP обмежується контролем цілісності даних в рамках однієї датаграми і не виключає можливості втрати пакета цілком або дублювання пакетів, порушення порядку отримання пакетів даних; TCP забезпечує надійну безперервну передачу даних, що виключає втрату даних або порушення порядку їх надходження або дублювання, може перерозподіляти дані, розбиваючи великі порції даних на фрагменти і навпаки, склеюючи фрагменти в один пакет.

Протоколи транспортного рівня: ATP (AppleTalk Transaction Protocol), CUDP (Cyclic UDP), DCCP (Datagram Congestion Control Protocol), FCP (Fiber Channel Protocol), IL (IL Protocol), NBF (NetBIOS Frames protocol), NCP (NetWare Core Protocol), SCTP (Stream Control Transmission Protocol), SPX (Sequenced Packet Exchange), SST (Structured Stream Transport), TCP (Transmission Control Protocol), UDP (User Datagram Protocol) [2].

(3) Мережевий рівень

Мережевий рівень (Network layer) моделі призначений для визначення шляху передачі даних. Відповідає за трансляцію логічних адрес і імен у фізичні, визначення найкоротших маршрутів, комутацію і маршрутизацію, відстеження неполадок і «заторів» в мережі.

Протоколи мережевого рівня маршрутизируют дані від джерела до одержувача. Працюючі на цьому рівні пристрою (маршрутизатори) умовно називають пристроями третього рівня (за номером рівня в моделі OSI).

Протоколи мережевого рівня: IP/IPv4/IPv6 (Internet Protocol), IPX (Internetwork Packet Exchange, протокол міжмережевого обміну), X.25 (частково цей протокол реалізований на рівні 2), CLNP (мережевий протокол без організації

з'єднань), IPsec (Internet Protocol Security). Протоколи маршрутизації - RIP (Routing Information Protocol), OSPF (Open Shortest Path First) [2].

(2) Канальний рівень

Канальний рівень (Data link layer) призначений для забезпечення взаємодії мереж на фізичному рівні і контролю за помилками, які можуть виникнути. Отримані з фізичного рівня дані, представлені в бітах, він упаковує в кадри, перевіряє їх на цілісність і, якщо потрібно, виправляє помилки (формує повторний запит пошкодженого кадру) і відправляє на мережевий рівень. Канальний рівень може взаємодіяти з одним або декількома фізичними рівнями, контролюючи і керуючи цим взаємодією.

Специфікація IEEE 802 розділяє цей рівень на два підрівні: MAC (Media access control) регулює доступ до поділюваного фізичного середовища, LLC (Logical link control) забезпечує обслуговування мережного рівня.

На цьому рівні працюють комутатори, мости та інші пристрої. Ці пристрої використовують адресацію другого рівня (за номером рівня в моделі OSI).

Протоколи канального рівня: ARCnet, ATM, Controller Area Network (CAN), Econet, IEEE 802.3 (Ethernet), Ethernet Automatic Protection Switching (EAPS), Fiber Distributed Data Interface (FDDI), Frame Relay, High-Level Data Link Control (HDLC), IEEE 802.2 (надає функції LLC для підрівня IEEE 802 MAC), Link Access Procedures, D channel (LAPD), IEEE 802.11 wireless LAN, LocalTalk, Multiprotocol Label Switching (MPLS), Point-to-Point Protocol (PPP), Point -to-Point Protocol over Ethernet (PPPoE), Serial Line Internet Protocol (SLIP, застарів), StarLan, Token ring, Unidirectional Link Detection (UDLD), x.25, ARP.

При розробці стеків протоколів на цьому рівні вирішуються завдання завадостійкого кодування. До таких способів кодування відноситься код Хеммінга, блочне кодування, код Ріда-Соломона.

У програмуванні цей рівень представляє драйвер мережевої плати, в операційних системах є програмний інтерфейс взаємодії канального і мережевого

рівнів між собою. Це не новий рівень, а просто реалізація моделі для конкретної ОС. Приклади таких інтерфейсів: ODI (англ.), NDIS, UDI [2].

(1) Фізичний рівень

Фізичний рівень (Physical layer) - нижній рівень моделі, який визначає метод передачі даних, представлених в двійковому вигляді, від одного пристрою (комп'ютера) до іншого. Складанням таких методів займаються різні організації, в тому числі: Інститут інженерів з електротехніки та електроніки, Альянс електронної промисловості, Європейський інститут телекомунікаційних стандартів і інші. Здійснюють передачу електричних або оптичних сигналів в кабель або в радіоефір і, відповідно, їх прийом і перетворення в біти даних відповідно до методами кодування цифрових сигналів.

На цьому рівні також працюють концентратори, повторювачі сигналу й медиаконвертери.

Функції фізичного рівня реалізуються на всіх пристроях, підключених до мережі. З боку комп'ютера функції фізичного рівня виконуються мережевим адаптером або послідовним портом. До фізичного рівня відносяться фізичні, електричні і механічні інтерфейси між двома системами. Фізичний рівень визначає такі види середовищ передачі даних як оптоволокно, кручена пара, коаксіальний кабель, супутниковий канал передач даних і т.п. Стандартними типами мережевих інтерфейсів, що відносяться до фізичного рівня, є: V.35, RS-232, RS-485, RJ-11, RJ-45, роз'єми AUI і BNC.

При розробці стеків протоколів на цьому рівні вирішуються завдання синхронізації і лінійного кодування. До таких способів кодування відноситься код NRZ, код RZ, MLT-3, PAM5, Манчестер II.

Протоколи фізичного рівня: IEEE 802.15 (Bluetooth), IRDA, EIA RS-232, EIA-422, EIA-423, RS-449, RS-485, DSL, ISDN, SONET / SDH, 802.11 Wi-Fi, Etherloop, GSM Um radio interface, ITU та ITU-T, TransferJet, ARINC 818, G.hn/G.9960 [2].

1.1.2. Основні мережеві протоколи відповідно до моделі OSI.

Нижче наведено основні протоколи безпеки мережі з більш детальним описом. Вони впорядковані мережевим рівнем, на якому працюють, знизу вгору [1].

Протокол безпеки Інтернет-протоколу (IPsec) — OSI Layer 3

IPsec — це набір протоколів і алгоритмів, які захищають дані, що передаються через загальнодоступні мережі, такі як Інтернет. Інженерна робоча група Інтернету (IETF) випустила протоколи IPsec у 1990-х роках. Вони шифрують і автентифікують мережеві пакети, щоб забезпечити безпеку рівня IP.

IPsec спочатку містив протоколи ESP і AH. Encapsulating Security Payload (ESP) шифрує дані та забезпечує автентифікацію, тоді як Authentication Header (AH) пропонує можливості захисту від повторного відтворення та захищає цілісність даних. З тих пір пакет розширився, включивши в нього протокол обміну ключами в Інтернеті (IKE), який надає спільні ключі, що встановлюють асоціації безпеки (SA). Вони дозволяють шифрувати та дешифрувати через брандмауер або маршрутизатор.

IPsec може захистити конфіденційні дані та VPN, забезпечуючи тунелювання для шифрування передачі даних. Він може шифрувати дані на прикладному рівні та забезпечує автентифікацію без шифрування.

SSL і TLS—OSI Layer 5

Протокол Secure Sockets Layer (SSL) шифрує дані, перевіряє джерела даних і забезпечує цілісність повідомлень. Він використовує сертифікати X.509 для автентифікації клієнта та сервера. SSL автентифікує сервер за допомогою рукостискання, погоджуючи параметри сеансу безпеки та генеруючи ключі сеансу. Потім він може безпечно передавати дані, автентифікуючи їх походження.

Сеанси SSL використовують криптографічні алгоритми, подібні до алгоритмів, що використовуються клієнтом і сервером (визначаються під час рукостискання). Сервери можуть підтримувати шифрування за допомогою таких алгоритмів, як AES і Triple DES.

Сертифікати сервера X.509 є обов’язковою умовою для SSL, що дозволяє клієнту перевіряти сервер. SSL також може використовувати клієнтські сертифікати X.509 для автентифікації. Ці сертифікати мають бути підписані довіреним центром сертифікації у зв’язку ключів сервера.

Безпека транспортного рівня (TLS) — це протокол на основі SSL, визначений IETF (SSL — ні).

Безпека транспортного рівня дейтаграм (DTLS)—OSI Layer 5

DTLS — це протокол безпеки обміну датаграмами на основі TLS. Це не гарантує доставку повідомлень або їх надходження в порядку. DTLS представляє переваги протоколів дейтаграм, включаючи меншу затримку та зменшення накладних витрат.

Протокол Kerberos—OSI Layer 7

Kerberos — це протокол автентифікації запиту служби для ненадійних мереж, як-от публічний Інтернет. Він автентифікує запити між довіреними хостами, пропонуючи вбудовану підтримку операційних систем Windows, Mac і Linux.

Windows використовує Kerberos як протокол автентифікації за замовчуванням і ключовий компонент таких служб, як Active Directory (AD). Постачальники послуг широкосмугового зв’язку використовують його для автентифікації приставок і кабельних модемів, які отримують доступ до їхніх мереж.

Системам, службам і користувачам потрібно лише довіряти KDC під час використання Kerberos. KDC пропонує автентифікацію та надає квитки, щоб вузли могли автентифікувати один одного. Kerberos використовує спільну секретну криптографію для автентифікації пакетів і захисту їх під час передачі.

Простий протокол керування мережею (SNMP)—OSI Layer 7

SNMP — це протокол керування та моніторингу мережевих пристроїв, який працює на прикладному рівні. Він може захистити пристрої в LAN або WAN. SNMP надає спільну мову, що дозволяє таким пристроям, як сервери та

маршрутизатори, спілкуватися через систему керування мережею. SNMP є оригінальною частиною набору Інтернет-протоколів, визначеного IETF.

Компоненти архітектури SNMP включають менеджер, агент і інформаційну базу управління (MIB). Менеджер — це клієнт, агент — сервер, а MIB — база даних. Агент SNMP відповідає на запити менеджера за допомогою MIB. Хоча SNMP широко доступний, адміністратори повинні налаштувати параметри за замовчуванням, щоб увімкнути зв’язок між агентами та системою керування мережею для реалізації протоколу.

З появою SNMPv3 у 2004 році протокол SNMP отримав три важливі функції безпеки: шифрування пакетів для запобігання прослуховування, перевірки цілісності, щоб переконатися, що пакети не були підроблені під час передачі, і автентифікацію, щоб перевірити, що зв’язок надходить із відомого джерела.

HTTP і HTTPS — OSI Layer 7

HTTP — це прикладний протокол, який визначає правила передачі веб-файлів. Користувачі опосередковано використовують HTTP, коли відкривають свій веб-браузер. Він працює поверх набору протоколів Інтернету.

HTTPS — це захищена версія HTTP, яка забезпечує зв’язок між браузерами та веб-сайтами. Це допомагає запобігти спуфінгу DNS і атакам типу "людина посередині", що важливо для веб-сайтів, які передають або отримують конфіденційну інформацію. Усі веб-сайти, які потребують входу користувачів або обробки фінансових транзакцій, є привабливими цілями крадіжки даних і повинні використовувати HTTPS.

HTTPS працює через протокол SSL або TLS із використанням відкритих ключів для шифрування спільних даних. HTTP за замовчуванням використовує порт 80, тоді як HTTPS використовує порт 443 для безпечних передач. За допомогою HTTPS сервер і браузер повинні встановити параметри зв’язку перед початком передачі даних [1].

1.2. Як виникли і розвивались протоколи SSL/TLS

SSL і TLS іноді використовуються як взаємозамінні, оскільки вони дуже тісно пов’язані. Фактично, можна розглядати TLS як оновлену версію SSL. Обидва протоколи шифрують інформацію на одному рівні, але, як і інші технології, вони розвиваються з часом.

Рівень захищених сокетів (SSL) і рівень безпеки транспортування (TLS) — це протоколи, які використовуються для шифрування зв’язку між веб-сервером і веб-браузером користувача. І SSL, і TLS шифрують ці зв’язки шляхом обміну відкритими та закритими ключами для створення безпечного сеансу. TLS використовує дещо інші криптографічні алгоритми для створення функції MAC секретних ключів і містить більше кодів сповіщень, ніж його попередник, SSL [3].

Перша доступна версія SSL - SSL 2.0, була розроблена Netscape і випущена в 1995 році. Однак у SSL 2.0 були виявлені вразливості, що вимагало від Netscape розробки кращої та безпечнішої версії. SSL 3.0 вийшов роком пізніше. SSL 3.0 все ще широко використовувався до осені 2014 року, коли команда безпеки Google виявила серйозну вразливість.

Основна причина, з якої SSL 3.0 вже не використовується, є численні відомі вразливості. Як згадувалося раніше, у 2014 році одна з команд Google виявила серйозну проблему під назвою POODLE.

Уразливість POODLE використовує резервні версії SSL 3.0, вбудовані в клієнти та сервери. По суті, атака змушує використовувати SSL 3.0, а потім використовує його для дешифрування частин вмісту сеансу. Виконуючи це байт за байтом, можна виявити великі частини з’єднання з сервером і клієнтом. Атака POODLE може бути використана на будь-якій системі, яка підтримує SSL 3.0 із шифруванням режиму ланцюгового блокування шифру.

Ще до POODLE у протоколі SSL були інші вразливості, такі як BEAST і BREACH. BEAST, спочатку знайдений у 2002 році, висвітлив уразливість CBC у

протоколах TLS 1.0 і попередніх протоколах, але на практиці він не був продемонстрований до 2011 року, що спонукало Microsoft, Apple і браузери створити обхідний шлях.

BREACH використовує стиснення HTTP для використання HTTPS і був представлений у 2013 році. Щоб обійти вразливість, клієнтам і серверам довелося використовувати захист від підробки міжсайтових запитів. [3].

TLS було вперше розроблено як ще одне оновлення протоколу SSL 3.0 у 1999 році. Хоча відмінності не вважаються драматичними, вони достатньо значні, щоб SSL 3.0 і TLS 1.0 не взаємодіяли. SSL 3.0 вважається менш безпечним, ніж TLS.

TLS 1.1 було створено в 2006 році, а TLS 1.2 було випущено в 2008 році. TLS 1.3 – це версія, яка використовується сьогодні. Як і будь-яке інше оновлення протоколу, TLS вважається більш безпечним, ніж SSL 3.0, завдяки доданим заходам для блокування використання та пом’якшення вразливостей у кожній версії [3].

1.3. Що собою представляє протокол SSL/TLS.

SSL розшифровується як Secure Sockets Layer і являє собою протокол для шифрування, захисту та автентифікації комунікацій, що відбуваються в Інтернеті. Хоча деякий час тому SSL було замінено оновленим протоколом під назвою TLS (Transport Layer Security), «SSL» досі є загальновживаним терміном для цієї технології.

Основним варіантом використання SSL/TLS є захист зв’язку між клієнтом і сервером, але він також може захистити електронну пошту, VoIP та інші комунікації через незахищені мережі [4].

Основні принципи роботи SSL/TLS [4]:

- Захищене спілкування починається з рукостискання TLS, під час якого дві сторони, що спілкуються, відкривають безпечне з’єднання та обмінюються відкритим ключем.
- Під час рукостискання TLS дві сторони генерують ключі сеансу, а ключі сеансу шифрують і розшифровують усі повідомлення після рукостискання TLS
- Різні ключі сеансу використовуються для шифрування зв’язку в кожному новому сеансі
- TLS гарантує, що сторона на стороні сервера або веб-сайт, з яким взаємодіє користувач, насправді є тим, за кого себе видає.
- TLS також гарантує, що дані не були змінені, оскільки код автентифікації повідомлення (MAC) додається до передачі
- За допомогою TLS шифруються дані HTTP, які користувачі надсилають на веб-сайт (натискаючи, заповнюючи форми тощо), і дані HTTP, які веб-сайти надсилають користувачам. Зашифровані дані мають бути розшифровані одержувачем за допомогою ключа.

Надсилаючи інформацію в Інтернеті, ми стикаємося з трьома основними проблемами безпеки:

- Як можливо знати, чи справді людина, з якою ми спілкуємось, є такою, якою вони кажуть?
- Як можливо знати, що дані не були підроблені з моменту надсилання?
- Як можливо завадити іншим людям бачити та отримувати доступ до даних?

Ці питання є вирішальними, особливо коли необхідно надіслати конфіденційну або цінну інформацію. TLS використовує низку криптографічних методів для вирішення кожної з цих трьох проблем. Разом вони дозволяють протоколу аутентифікувати іншу сторону у зв’язку, перевірити цілісність даних та забезпечити зашифрований захист.

Наприклад склалась ситуація, коли необхідно передати інформацію другу, який живе в іншій країні. Якщо інформація є конфіденційною, три вищезгадані основні проблеми мають бути враховані.

Не можна надіслати лист і сподіватись на краще, особливо якщо є підозра, що на дану комунікацію націлені зловмисники. Натомість необхідна система, яка дозволяє переконатися, що одержувач легітимний, спосіб перевірити чи змінили повідомлення та спосіб захистити їх від сторонніх очей.

TLS виконує ці вимоги, використовуючи ряд різних процесів. Починається з того, що відомо як TLS рукостискання, саме там відбувається аутентифікація та встановлюються ключі.

Повертаючись до аналогії листів, частина аутентифікації TLS була б такою, як відправлення листа через кур'єра, який вимагає ідентифікації. Коли кур'єр доставить лист, він порівняє посвідчення особи з їх обличчям і перевірить, чи був одержувач правильною особою чи ні.

Ключова фаза встановлення була б трохи схожа на те, якби лист містив половину PIN-коду, який використовуватиметься в майбутніх комунікаціях. Одержувача придумую другу половину номера і надсилає його у своєму зворотному листі.

Після того, як кур'єр підтвердив особу та номер PIN-коду було встановлено, отримано все необхідне для надійного надсилання інформації. Насправді ці етапи є набагато складнішими, але це буде висвітлено згодом.

TLS надійно обмінюється інформацією з протоколом програми. За вищенаведеною аналогію надійно передавати дані через TLS було б як написати повідомлення та помістити їх у конверт. Лист буде підписано через печатку, так що якщо лист був підроблений, одержувач зможе сказати по зірваному підпису (в реальності при застосуванні TLS це робиться із застосуванням математики).

Потім лист кладеться у невеликий металевий ящик, який має комбінований замок, комбінацію встановлюється як контактний номер, який був встановлений

спільно з одержувачем. Коробка відправляється через кур’єра, який перевіряє ідентифікацію при доставці пакетів. Одержувач відповідає тим самим чином, і будь-яке майбутнє повідомлення виконує ті ж дії.

TLS вирішує всі три проблеми відносно подібним чином. Кур’єр служить для аутентифікації одержувача та переконання в тому, що коробка доставлена її призначеній особі. Заблокована скринька служить формою шифрування, не даючи доступу до листів будь-кому, крім одержувача. Підписаний конверт дає змогу дізнатися, чи не було підроблено повідомлення.

Це дуже грубе наближення до того, що робить TLS. Насправді, TLS відбувається між клієнтами та серверами, а не двома людьми, які пересилають пошту один одному. Аналогія полягає лише в тому, щоб надати уявлення про те, що відбувається, і міркування щодо цього [5].

Таким чином протокол TLS призначений для надання трьох послуг усім додаткам, що працюють над ним, а саме: шифрування, аутентифікацію та цілість. Технічно, не всі три можуть використовуватися, однак на практиці, для забезпечення безпеки, як правило використовуються всі три:

Шифрування – укриття інформації, переданої від одного комп’ютера до іншого;

Аутентифікація – перевірка авторства інформації, що передається;

Цілісність – виявлення підміни інформації підробкою.

Для того щоб встановити криптографічно безпечний канал даних, вузли з'єднання повинні узгодити використовувані методи шифрування та ключі. Протокол TLS однозначно визначає дану процедуру (TLS Handshake). Далі відзначимо, що TLS використовує криптографію з відкритим ключем, який дозволяє встановити загальний секретний ключ шифрування без будь-яких попередніх відомостей іншого або іншого.

Також в рамках процедури TLS Handshake є можливість встановити довжину особистостей і клієнта, і сервера. Наприклад, клієнт може бути впевнений, що

сервер, який надає йому інформацію про банківський рахунок, дійсно банківський сервер. І навпаки: сервер компанії може бути впевнений, що клієнт, підключений до нього – саме співробітник компанії, а не стороння особа (Chain of Trust).

Після цього TLS забезпечує виправлення кожного повідомлення з кодом MAC (Message Authentication Code), алгоритмом створення якого – одностороння криптографічна функція хешування (фактично – контрольна сума), ключі, які відомі обом учасникам зв'язку. Кожен раз при відправленні повідомлень, генерується його MAC-значення, яке може згенерувати і приймати, це забезпечує повноту інформації та захист від її підменю.

Таким чином, коротко розглянуті всі три механізми, що лежать в основі криптобезпеки протоколу TLS [6].

Перед тим, як розпочати обмін даними через TLS, клієнт і сервер повинні узгодити параметри з'єднання, а саме: версія протоколу, спосіб шифрування даних, а також перевірити сертифікати, якщо це необхідно. Схема початку з'єднання називається TLS Handshake і показана на Рис 1.2 [6]:

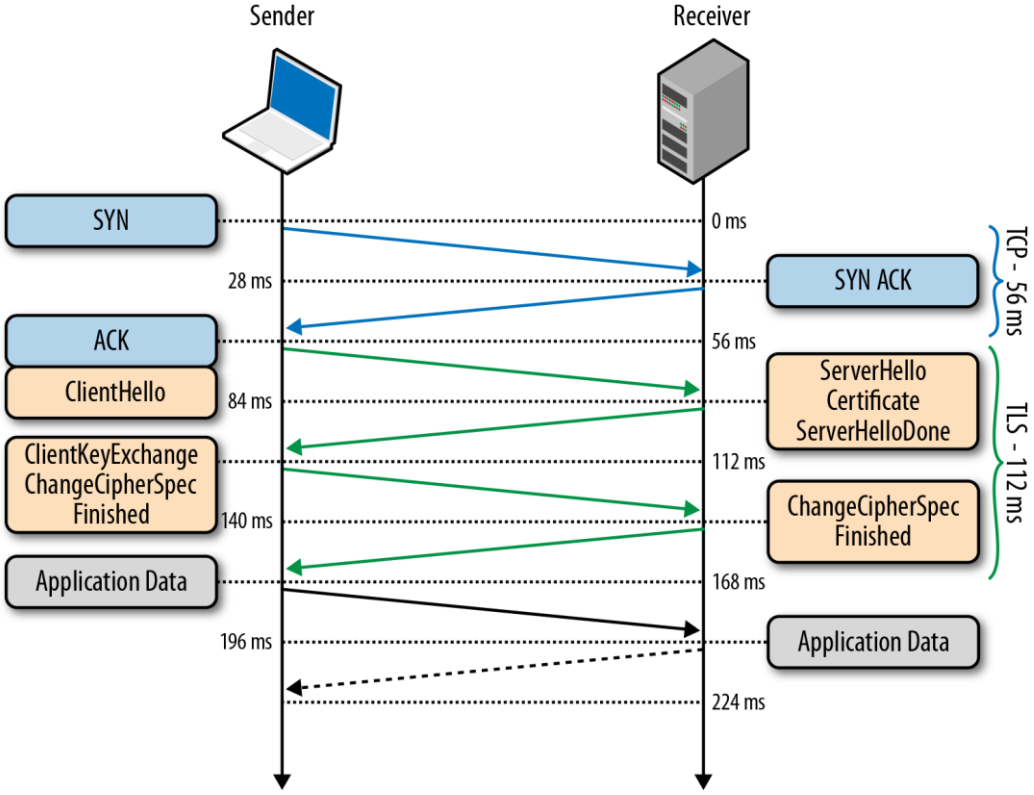

Рис. 1.2. TLS Handshake

Розберемо докладніше кожен крок цієї процедури [6]:

1) Оскільки TLS працює над TCP, спочатку між клієнтом і сервером встановлюється TCP-з'єднання.
2) Після встановлення TCP, клієнт посилає на сервер специфікацію у вигляді звичайного тексту (а саме версію протоколу, яку він хоче використовувати, методи шифрування, що підтримуються тощо).
3) Сервер затверджує версію протоколу, що використовується, вибирає спосіб шифрування з наданого списку, прикріплює свій сертифікат і відправляє відповідь клієнту (при бажанні сервер може також запросити клієнтський сертифікат).
4) Версія протоколу та спосіб шифрування на даний момент вважаються затвердженими, клієнт перевіряє надісланий сертифікат та ініціює або RSA, або обмін ключами по Діффі-Хеллману, залежно від встановлених параметрів.
5) Сервер обробляє надіслане клієнтом повідомлення, звіряє MAC, і відправляє клієнту заключне (Finished) повідомлення в зашифрованому вигляді.
6) Клієнт розшифровує отримане повідомлення, звіряє MAC, і якщо все добре, з'єднання вважається встановленим і починається обмін даними додатків.

Зрозуміло, що встановлення з'єднання TLS є, власне кажучи, тривалим і трудомістким процесом, у стандарті TLS є кілька оптимізації. Зокрема, є процедура під назвою "abbreviated handshake", яка дозволяє використовувати раніше узгоджені параметри для відновлення з'єднання (зрозуміло, якщо клієнт і сервер встановлювали TLS-з'єднання в минулому). Цю процедуру розглянуто докладніше у пункті «Відновлення сесії».

Також є додаткове розширення процедури Handshake, що має назву TLS False Start. Це розширення дозволяє клієнту та серверу розпочати обмін зашифрованими даними одразу після встановлення методу шифрування, що скорочує встановлення з'єднання на одну ітерацію повідомлень [6].

Відновлення сесії TLS

Як уже зазначалося раніше, повна процедура TLS Handshake є досить тривалою та дорогою з погляду обчислювальних витрат. Тому було розроблено процедуру, що дозволяє відновити раніше перерване з'єднання з урахуванням вже сконфігурованих даних.

Починаючи з першої публічної версії протоколу (SSL 2.0), сервер в рамках TLS Handshake (а саме початкового повідомлення ServerHello) може згенерувати та відправити 32-байтний ідентифікатор сесії. Природно, в такому випадку сервер зберігає кеш згенерованих ідентифікаторів і параметрів сеансу для кожного клієнта. У свою чергу клієнт зберігає у себе надісланий ідентифікатор і включає його (звичайно, якщо він є) у початкове повідомлення ClientHello. Якщо і клієнт, і сервер мають ідентичні ідентифікатори сесії, то встановлення загального з'єднання відбувається за спрощеним алгоритмом, показаним на Рис 1.3. Якщо ні, то потрібна повна версія TLS Handshake.

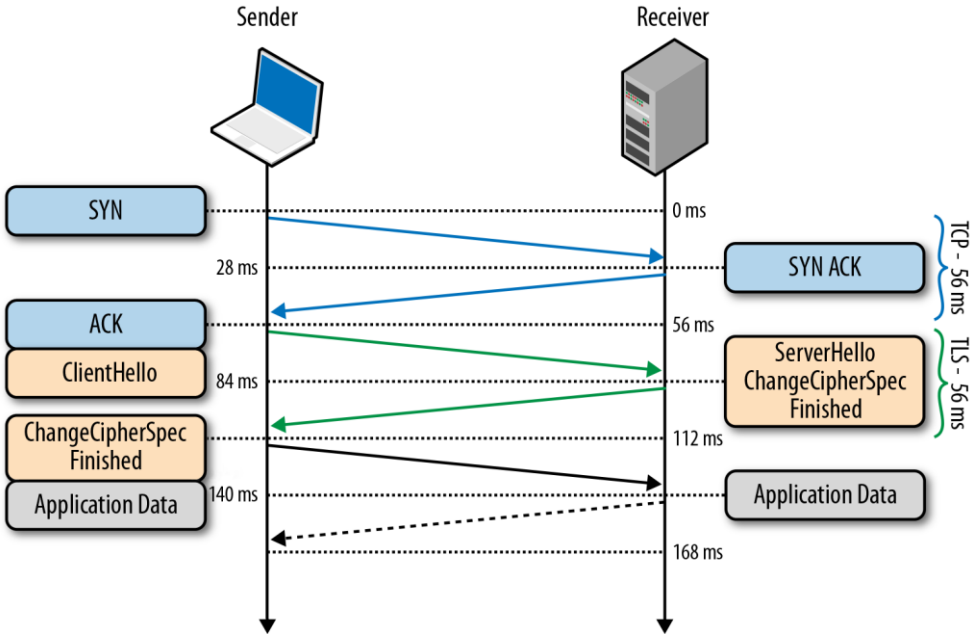

Рис. 1.3. Відновлення сесії TLS

Процедура відновлення сесії дозволяє пропустити етап генерації симетричного ключа, що суттєво підвищує час встановлення з'єднання, але не

впливає на його безпеку, оскільки використовуються раніше нескомпрометовані дані попередньої сесії.

Однак тут є практичне обмеження: оскільки сервер повинен зберігати дані про всі відкриті сесії, це призводить до проблеми з популярними ресурсами, які одночасно запитуються тисячами та мільйонами клієнтів.

Для обходу цієї проблеми розроблено механізм «Session Ticket», який усуває необхідність зберігати дані кожного клієнта на сервері. Якщо клієнт при початковій установці з'єднання вказав, що він підтримує цю технологію, то сервер в ході TLS Handshake відправляє клієнту так званий Session Ticket - параметри сесії, зашифровані закритим ключем сервера. При наступному поновленні сесії, клієнт разом з ClientHello відправляє Session Ticket. Таким чином, сервер позбавлений необхідності зберігати дані про кожне з'єднання, але з'єднання, як і раніше, безпечне, оскільки Session Ticket зашифрований ключем, відомим тільки на сервері [6].

TLS False Start

Технологія поновлення сесії безперечно підвищує продуктивність протоколу та знижує обчислювальні витрати, проте вона не застосовна у початковому з'єднанні з сервером, або у випадку, коли попередня сесія вже закінчилася.

Для отримання ще більшої швидкодії була розроблена технологія TLS False Start, яка є опціональним розширенням протоколу і дозволяє надсилати дані, коли TLS Handshake завершено лише частково. Детальна схема TLS False Start представлена на Рис 1.4.

Важливо, що TLS False Start не змінює процедуру TLS Handshake. Він заснований на припущенні, що в той момент, коли клієнт і сервер вже знають про параметри з'єднання та симетричні ключі, дані програм вже можуть бути надіслані, а всі необхідні перевірки можна провести паралельно. В результаті, з'єднання готове до використання на одну ітерацію обміну повідомленнями раніше [6].

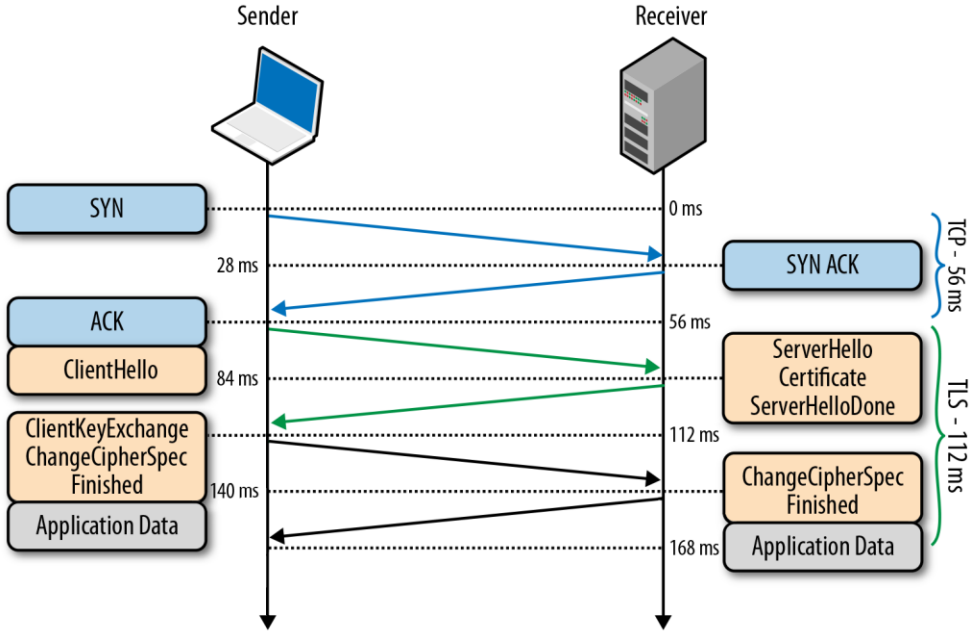

Рис. 1.4. TLS False Start

TLS Chain of trust

Аутентифікація є невід'ємною частиною кожного з'єднання TLS. Розглянемо найпростіший процес аутентифікації між Алісою та Бобом:

І Аліса, і Боб генерують власні відкриті та закриті ключі.

Аліса та Боб обмінюються відкритими ключами.

Аліса генерує повідомлення, шифрує його своїм закритим ключем та відправляє Бобу.

Боб використовує отриманий від Аліси ключ, щоб розшифрувати повідомлення, і таким чином перевіряє справжність отриманого повідомлення.

Очевидно, що ця схема побудована на довірі між Алісою та Бобом. Передбачається, що обмін відкритими ключами відбувся, наприклад, під час особистої зустрічі, і, таким чином, Аліса впевнена, що отримала ключ безпосередньо від Боба, а Боб у свою чергу впевнений, що отримав відкритий ключ Аліси.

Нехай тепер Аліса отримує повідомлення від Чарлі, з яким вона не знайома, але який стверджує, що дружить із Бобом. Щоб це довести, Чарлі заздалегідь

попросив підписати власний відкритий ключ закритим ключем Боба і прикріплює цей підпис до повідомлення Алісі. Аліса спочатку перевіряє підпис Боба на ключі Чарлі (це вона в змозі зробити, адже відкритий ключ Боба їй уже відомий), переконується, що Чарлі дійсно друг Боба, приймає його повідомлення і виконує вже відому перевірку цілісності, переконуючись, що повідомлення дійсно від Чарлі (Рис 1.5) [6]:

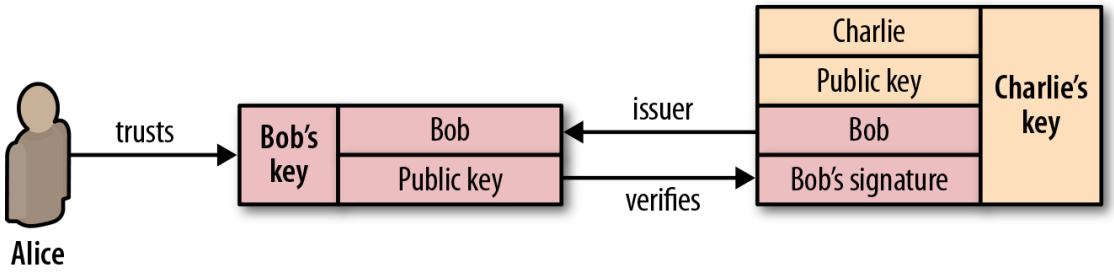

Рис. 1.5. Перевірка цілісності

Описане в попередньому абзаці є створення «ланцюжка довіри» (або «Chain of trust», якщо англійською).

У протоколі TLS дані ланцюга довіри засновані на сертифікатах автентичності, що надаються спеціальними органами, які називають центрами сертифікації (CA – certificate authorities). Центри сертифікації проводять перевірки і, якщо виданий сертифікат скомпрометований, цей сертифікат відкликається.

З виданих сертифікатів складається вже розглянутий ланцюжок довіри. Коренем її є так званий "Root CA certificate" - сертифікат, підписаний великим центром, довіра до якого незаперечна. У загальному вигляді ланцюжок довіри виглядає приблизно таким чином (Рис 1.6):

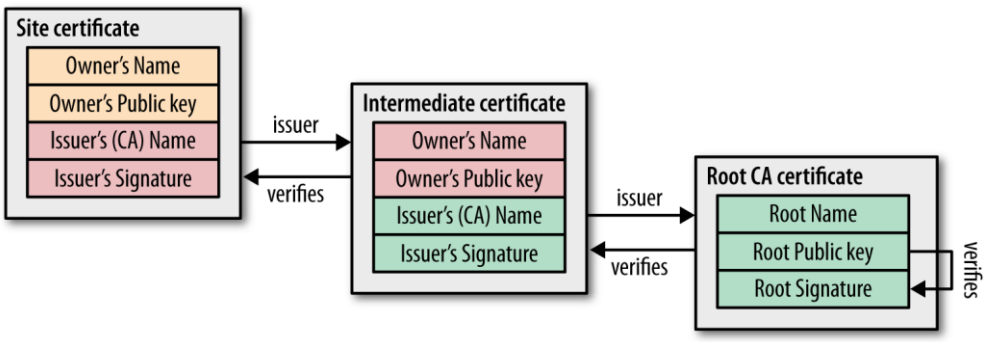

Рис. 1.6. Ланцюжок довіри

Природно, виникають випадки, коли вже виданий сертифікат необхідно відкликати або анулювати (наприклад, було скомпрометовано закритий ключ сертифіката, або скомпрометовано всю процедуру сертифікації). Для цього сертифікати автентичності містять спеціальні інструкції щодо перевірки їх актуальності. Отже, при побудові ланцюжка довіри необхідно перевіряти актуальність кожного довірчого вузла.

Механізм цієї перевірки є простим і в його основі лежить так званий "Список відкликаних сертифікатів" (CRL - "Certificate Revocation List"). У кожного з центрів сертифікації є список, що представляє простий перелік серійних номерів відкликаних сертифікатів. Відповідно будь-хто, хто хоче перевірити справжність сертифіката, просто завантажує даний список і шукає в ньому номер сертифіката, що перевіряється. Якщо номер виявиться – це означає, що сертифікат відкликано.

Тут очевидно присутня деяка технічна нераціональність: для перевірки лише одного сертифіката потрібно вимагати весь список відкликаних сертифікатів, що спричиняє уповільнення роботи. Для боротьби з цим розроблено механізм під назвою «Протокол статусів сертифікатів» (OCSP – Online Certificate Status Protocol). Він дозволяє здійснювати перевірку статусу сертифіката динамічно. Звичайно, це знижує навантаження на пропускну здатність мережі, але водночас породжує кілька проблем:

- Центри сертифікації повинні справлятися із навантаженням у режимі реального часу;
- Центри сертифікації повинні гарантувати доступність у будь-який час;
- Через запити реального часу центри сертифікації одержують інформацію про те, які сайти відвідував кожен конкретний користувач.

Власне, у всіх сучасних браузерах обидва рішення (OCSP і CRL) доповнюють один одного, більше того, як правило є можливість налаштування політики перевірки статусу сертифіката [6].

Висновки до розділу 1

Отже, в даному розділі розглянуто місце протоколу SSL/TLS в стеку протоколів захисту мережі – TLS використовується на транспортному рівні.

Хоча назви SSL і TLS використовуються взаємозамінно, фактично TLS є новою версією SSL, а сам SSL є застарілим і вразливим та не використовується з 2014 року.

TLS забезпечує належне шифрування даних, що передаються, таким чином надаючи три основні послуги:

- Шифрування – укриття інформації, переданої від одного комп’ютера до іншого;
- Аутентифікація – перевірка авторства інформації, що передається;
- Цілісність - виявлення підміни інформації підробкою.

Розділ 2

ВИМОГИ МІЖНАРОДНИХ СТАНДАРТІВ ДО ВИКОРИСТАННЯ SSL/TLS

2.1. Основні відмінності SSL та TLS.

Основні переваги TLS протоколу, які роблять його більш безпечним та ефективним, ніж протокол SSL:

- Аутентифікація повідомлень HMAC
- Псевдовипадкова функція (PRF) для генерації ключів
- Набір шифрів AES, який є найбільш безпечним алгоритмом
- Простий спосіб отримання повідомлення CertificateVerify

TLS використовує TCP порт 443, який працює практично у будь-якому середовищі та відкритий на більшості брандмауерів. Це може бути особливо корисним для користувачів та працівників, які працюють віддалено і знаходяться поза зоною дії брандмауера організації. TLS протокол доступний для будь-якого сучасного комп'ютера чи пристрою. Його можливо налаштувати для забезпечення більш високого рівня захисту, ніж при використанні IPSec.

Глобальному бізнесу часто доводиться керувати досить великою мережею співробітників. Використання протоколу TLS дозволяє отримати більше контролю над мережею і відповідно більш безпечно розширюватися [7].

SSL розшифровується як Secure Socket Layer, тоді як TLS означає Transport Layer Security. Протоколи Secure Socket Layer і Transport Layer Security обидва використовуються для забезпечення безпеки між веб-браузерами та веб-серверами. Основна відмінність між Secure Socket Layer і Transport Layer Security полягає в тому, що в SSL (Secure Socket Layer) дайджест повідомлень використовується для створення головного секрету та забезпечує базові служби безпеки, як-от

автентифікацію та конфіденційність. тоді як у TLS (Transport Layer Security) для створення головного секрету використовується псевдовипадкова функція [8].

Існують деякі відмінності між SSL і TLS, які наведені в таблиці 2.1 [8]:

Таблиця 2.1

Відмінності між SSL і TLS

SSL	**TLS**
SSL означає Secure Socket Layer	TLS означає Transport Layer Security
SSL (Secure Socket Layer) підтримує алгоритм Fortezza	TLS (Transport Layer Security) не підтримує алгоритм Fortezza
SSL (Secure Socket Layer) — версія 3.0	TLS (Transport Layer Security) — це версія 1.3
У SSL (Secure Socket Layer) дайджест повідомлень використовується для створення головного секрету	У TLS (Transport Layer Security) для створення головного секрету використовується псевдовипадкова функція
У SSL (Secure Socket Layer) використовується протокол коду автентифікації повідомлення	У TLS (Transport Layer Security) використовується протокол коду автентифікації хешованого повідомлення
SSL (Secure Socket Layer) є більш складним, ніж TLS (Transport Layer Security)	TLS (Transport Layer Security) простий
SSL (Secure Socket Layer) менш захищений порівняно з TLS (Transport Layer Security)	TLS (Transport Layer Security) забезпечує високий рівень безпеки

SSL є менш надійним і повільним	TLS є високонадійним і оновленим. Це забезпечує меншу затримку
SSL є застарілим	TLS все ще широко використовується
SSL використовує порт для встановлення явного підключення	TLS використовує протокол для встановлення неявного підключення

Функціональні особливості різних версій протоколів SSL/TLS [9]:

1) SSL 3.0

Цей протокол був випущений у 1996 році, але спочатку він почався зі створення SSL 1.0, розробленого Netscape. Версія 1.0 не була випущена, а версія 2.0 мала низку недоліків безпеки, що призвело до випуску SSL 3.0. Деякі основні вдосконалення SSL 3.0 порівняно з SSL 2.0:

- Відокремлення транспортування даних від рівня повідомлень
- Використання повних 128 біт матеріалу ключа навіть при використанні шифру експорту
- Здатність клієнта та сервера надсилати ланцюжки сертифікатів, що дозволяє організаціям використовувати ієрархію сертифікатів, глибина якої перевищує два сертифікати.
- Реалізація узагальненого протоколу обміну ключами, що дозволяє обмінюватися ключами Діффі-Хеллмана та Fortezza, а також сертифікатами, що не є RSA.
- Дозволяє стискати та розпаковувати записи
- Можливість повернутися до SSL 2.0, коли зустрічається клієнт 2.0

2) TLS 1.0

Цей протокол було вперше визначено в RFC 2246 у січні 1999 року. Це було оновлення SSL 3.0, і відмінності не були драматичними, але вони досить значні,

щоб SSL 3.0 і TLS 1.0 не взаємодіяли. Деякі з основних відмінностей між SSL 3.0 і TLS 1.0:

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

3) TLS 1.1

Цей протокол було визначено в RFC 4346 у квітні 2006 року, він є оновленням до TLS 1.0. Основні зміни:

- Вектор неявної ініціалізації (IV) замінено на явний IV для захисту від атак з ланцюжком блоків шифру (CBC).
- Обробку доповнених помилок змінено на використання сповіщення bad_record_mac замість сповіщення decryption_failed для захисту від атак CBC.
- Реєстри IANA визначені для параметрів протоколу
- Передчасне закриття більше не призводить до неможливості відновлення сеансу.

4) TLS 1.2

Цей протокол було визначено в RFC 5246 у серпні 2008 року. На основі TLS 1.1 TLS 1.2 містить покращену гнучкість. Основні відмінності включають:

- Комбінацію MD5/SHA-1 у псевдовипадковій функції (PRF) було замінено PRF, визначеними набором шифрів.
- Комбінацію MD5/SHA-1 в елементі з цифровим підписом було замінено одним хешем. Елементи зі знаком включають поле, яке явно визначає використовуваний алгоритм хешування.

- Відбулося суттєве очищення здатності клієнта та сервера вказувати, які хеш-алгоритми та алгоритми підпису вони будуть приймати.
- Додано підтримку автентифікованого шифрування з додатковими режимами даних.
- Визначення розширень TLS і пакети шифрів AES були об’єднані.
- Посилена перевірка номерів версій EncryptedPreMasterSecret.
- Багато вимог було посилено
- Довжина Verify_data залежить від набору шифрів
- Опис захисту від атаки Bleichenbacher/Dlima очищено.

5) TLS 1.3

Цей протокол наразі переглядається та знаходиться у своєму 28-му проекті. Основні відмінності від TLS 1.2:

- Список підтримуваних симетричних алгоритмів було вилучено з усіх застарілих алгоритмів. Усі решта алгоритмів використовують алгоритми автентифікованого шифрування з пов’язаними даними (AEAD).
- Було додано режим нульового RTT (0-RTT), який зберігає зворотну передачу під час встановлення з’єднання для деяких даних програми за рахунок певних властивостей безпеки.
- Статичний RSA та набори шифрів Діффі-Хеллмана були видалені; усі механізми обміну ключами на основі відкритого ключа тепер забезпечують пряму секретність.
- Усі повідомлення рукостискання після ServerHello тепер зашифровані.
- Функції похідного ключа були перероблені з використанням функції вилучення та розширення ключа (HKDF) на основі HMAC як примітиву.
- Кінцевий автомат рукостискання було реструктуровано, щоб бути більш послідовним і видалити зайві повідомлення.

- ECC тепер є базовою специфікацією та включає нові алгоритми підпису. Узгодження формату точки було видалено на користь формату однієї точки для кожної кривої.
- Стиснення, користувацькі групи DHE та DSA було вилучено, доповнення RSA тепер використовує PSS.
- Механізм перевірки узгодження версії TLS 1.2 застарів на користь списку версій у розширенні.
- Відновлення сеансу зі станом на стороні сервера та без нього, а також набори шифрів на основі PSK попередніх версій TLS були замінені одним новим обміном PSK.

2.2. Нормативно-правове регулювання використання SSL/TLS.

TLS зазвичай працює тихо у фоновому режимі, але всупереч тому, що можна подумати, TLS не є чорною скринькою, яка просто працює. Натомість TLS забезпечує безпеку завдяки взаємодії різних криптографічних алгоритмів. Крім того, TLS, як і SSL раніше, постійно розвивається разом із індустрією безпеки — потрібно задовольняти нові технології та бізнес-вимоги, а останні загрози безпеці — пом’якшувати. Алгоритми можуть застаріти з часом, або практики можуть бути відмовлені, причому кожна зміна впливає на загальну безпеку примірника TLS (наприклад, того, що зараз захищає з’єднання).

Ця нестабільність спонукала різні організації зі стандартизації публікувати керівні документи, щоб можна було встановити мінімальний базовий рівень безпеки TLS на певному ринку, секторі чи службі. На жаль, існує велика кількість таких стандартів, причому різні сектори вимагають дотримання різних застосовних документів, а самі стандарти також змінюються з часом, враховуючи зміни в секторі, для захисту якого вони розроблені.

Зрозуміло, що навігація в цьому морі стандартів для налаштування сучасного екземпляра TLS може стати справжнім головним болем для адміністраторів [10].

Є кілька організацій, які дотримуються вказівок щодо TLS щодо безпеки мережі, як-от Департамент охорони здоров'я та соціальних служб США (HHS) або Національний інститут стандартів і технологій (NIST). Далі буде розглянуто три найпоширеніші документи [10]:

- Закон про перенесення та підзвітність медичного страхування (HIPAA - The Health Insurance Portability and Accountability Act)
- Рекомендації NIST SP 800-52r2

Стандарт безпеки даних платіжних карток (PCI-DSS - The Payment Card Industry Data Security Standard)

1) HIPAA

HIPAA – це постанова, прийнята урядом США в 1996 році, яка стосується безпечної обробки захищеної медичної інформації (PHI). PHI відноситься до будь-якої цифрової інформації про пацієнта, такої як результати аналізів або діагнози. Керівний документ HIPAA, опублікований у 2013 році, зазначає наступне:

Дійсні процеси шифрування для даних у русі – це ті, які відповідають, відповідно, NIST Special Publications 800-52, Рекомендації щодо вибору та використання реалізацій безпеки транспортного рівня (TLS); 800-77, Guide to IPsec VPNs; або 800-113, Guide to SSL VPNs, або інші, які підтверджені Федеральними стандартами обробки інформації (FIPS) 140-2.

2) Стандарти NIST

У 2005 році NIST опублікував спеціальну публікацію (SP) 800-52, яка описує правильні робочі процедури для безпечного налаштування примірника TLS для державних серверів. З тих пір SP 800-52 було замінено версіями SP 800-52r1 (2014) і SP 80052r2 (2019). Наразі вказівки версії SP 800-52r2 є стабільними.

3) PCI-DSS

PCI-DSS — це стандарт відповідності, підтримуваний Радою безпеки стандартів індустрії платіжних карток (SSC), який встановлює, як платіжні та

карткові дані обробляються веб-сайтами електронної комерції. Щодо належної конфігурації екземплярів TLS PCI-DSS зазначає:

«Зверніться до галузевих стандартів і найкращих практик, щоб отримати інформацію про надійну криптографію та безпечні протоколи (наприклад, NIST SP 800-52 і SP 800-57, OWASP тощо)»

Стандарти TLS: об’єднавши все це разом

Наразі слід зазначити, що кожен стандарт впливає на різні системи залежно від їх функції та даних, які вони обробляють. Наприклад, лікарняний сервер електронної пошти може підпадати під дію рекомендацій HIPAA, оскільки обмінювані повідомлення можуть містити інформацію про пацієнта, тоді як CRM-система лікарні може підпадати під PCI-DSS, оскільки вона може містити дані кредитних карток та інші дані клієнтів. Для відповідності всім трьом стандартам потрібно використовувати загальні параметри TLS, присутні в усіх документах.

Всі стандарти на даний момент дотримуються вказівок NIST щодо вибору параметрів TLS. Це означає, що сумісність із SP 800-52r2 повинна зробити сервер також сумісним із HIPAA та PCI-DSS [10].

Настроювані параметри TLS

На рівень безпеки, який забезпечує TLS, найбільше впливає версія протоколу (наприклад, 1.0, 1.1 тощо) і дозволені набори шифрів. Шифри — це алгоритми, які виконують шифрування та дешифрування. Однак набір шифрів — це набір алгоритмів, включаючи шифр, алгоритм обміну ключами та алгоритм хешування, які використовуються разом для встановлення безпечного з’єднання TLS. Більшість клієнтів і серверів TLS підтримують кілька альтернатив, тому під час встановлення захищеного з’єднання їм потрібно узгодити загальну версію TLS і набір шифрів.

Версія протоколу TLS

Щодо підтримки версії TLS, NIST SP 800-52r2 зазначає наступне:

Сервери, які підтримують лише державні програми, мають бути налаштовані на використання TLS 1.2, а також на використання TLS 1.3. Ці сервери не повинні бути налаштовані на використання TLS 1.1 і не повинні використовувати TLS 1.0, SSL 3.0 або SSL 2.0.

Сервери, які підтримують громадянські або бізнес-програми (тобто клієнт може не бути частиною урядової IT-системи), повинні бути налаштовані для узгодження TLS 1.2 і повинні бути налаштовані для узгодження TLS 1.3. Використання TLS версій 1.1 і 1.0, як правило, не рекомендується, але ці версії можуть бути налаштовані, якщо необхідно, щоб увімкнути взаємодію з громадянами та підприємствами… Ці сервери не повинні дозволяти використання SSL 2.0 або SSL 3.0.

Агентства мають підтримувати TLS 1.3 до 1 січня 2024 року. Після цієї дати сервери підтримуватимуть TLS 1.3 як для державних, так і для громадян чи бізнес-додатків. Загалом, сервери, які підтримують TLS 1.3, також повинні бути налаштовані на використання TLS 1.2. Проте TLS 1.2 може бути вимкнено на серверах, які підтримують TLS 1.3, якщо було визначено, що TLS 1.2 не потрібен для взаємодії.

Хоча TLS 1.0 заборонено, а TLS 1.1 не підтримується для урядових сайтів, інструкції NIST стверджують, що для забезпечення сумісності зі службами третіх сторін державні сервери можуть застосовувати TLS 1.0 і 1.1, коли це необхідно. Відповідно до PCI-DSS 3.2.1 (поточна версія), сумісні сервери повинні відмовитися від підтримки TLS 1.0 і «перейти на мінімум TLS 1.1, бажано TLS 1.2». Технічно HIPAA дозволяє використовувати всі версії TLS. Таким чином, мінімально підтримувана версія TLS становить 1.1; проте PCI-DSS і NIST наполегливо рекомендують використовувати більш безпечний TLS 1.2 (і, як видно вище, NIST рекомендує прийняти TLS 1.3 і планує вимагати підтримки до 2024 року).

Шифри

TLS 1.2 і раніше

SP 800-52r2 визначає різноманітність прийнятних наборів шифрів для TLS 1.2 і попередніх версій. Стандарт не вимагає підтримки будь-яких конкретних наборів шифрів, але пропонує вказівки щодо вибору більш надійних:

- Віддавайте перевагу ефемерним ключам над статичними (тобто віддайте перевагу DHE над DH і ECDHE над ECDH). Ефемерні ключі забезпечують ідеальну секретність.
- Надавайте перевагу режимам GCM або CCM над режимом CBC. Використання автентифікованого режиму шифрування запобігає кільком атакам (додаткову інформацію див. у розділі 3.3.2 [SP 800-52r2). Зауважте, що вони недоступні у версіях до TLS 1.2.
- Віддавайте перевагу CCM над CCM_8. Останній містить коротший тег автентифікації, що забезпечує нижчу надійність автентифікації.

Крім того, хоча це дозволені набори шифрів, якщо сервер TLS не працює з великою різноманітністю різних платформ і клієнтів, рекомендується використовувати лише невелику підмножину цих алгоритмів. Дозвіл більшої кількості наборів шифрів може лише розширити поверхню атаки для сервера, якщо (або коли) буде виявлено нову вразливість протоколу.

Висновки до розділу 2

Отже, TLS 1.3 на даний момент є найбільш безпечною, найбільш використовуваною версією протоколу захисту мережевих з’єднань.

Використання TLS і правильне його налаштування так чи інакше є вимогою основних міжнародних стандартів. З найбільш відомих стандартів правильне налаштування використання SSL/TLS регламентують такі нормативні документи як HIPAA, NIST SP 800-52r2, PCI-DSS.

Розділ 3

ВРАЗЛИВОСТІ ПРОТОКОЛІВ SSL/TLS. НАЙБІЛЬШ ВІДОМІ АТАКИ

3.1. Renegotiation Attack – Атака повторного ініціювання з'єднанняа

Основні моменти уразливості Renegotiation Attack [11]:

- Відкрита у 2009 році Маршем Реєм і Стівом Діспенсою
- Уразливість у функції повторного узгодження TLS
- Повторне узгодження використовується під час спроби автентифікації під час сеансу SSL
- Зловмисник ініціює сеанс із сервером
- Викрадає з'єднання між клієнтом і сервером, утримуючи пакети рукостискання, надіслані клієнтом
- Запитує автентифікацію від сервера
- Натомість показує рукостискання клієнта
- Тепер сервер вважає, що зловмисник є клієнтом
- Для пом'якшення потрібно вимкнути функцію або запровадити розширення бібліотеки

Процес узгодження шифрування SSL використовує значно більше ресурсів на сервері, ніж на клієнті. Таким чином, якщо клієнт може ініціювати процес повторного узгодження, зловмисник може зробити сервер недоступним за допомогою атаки «Відмова в обслуговуванні».

Помилка повторного узгодження SSL може впливати на різні типи систем. По суті, це спричинено уразливістю в ініційованому клієнтом повторному узгодженні SSL/TLS для існуючих підключень до сервера.

Деякі симптоми проблем із переглядом переговорів включають [12]:

- Це не вдається, якщо віртуальний сервер обробляє підключення SSL

- Є системні повідомлення про помилки BIG-IP щодо рукостискання/повторного узгодження SSL
- Віртуальний сервер, який відхиляє спроби повторного узгодження, реєструється в статистиці SSL

У двох словах, зв’язок SSL — або процес рукостискання — передбачає обмін повідомленнями між сервером і клієнтом. Вони визначають параметри зашифрованого зв’язку, включаючи ввімкнені набори шифрів, версію протоколу, безпеку повторного узгодження та інші.

Рукостискання SSL містить такі компоненти, як ClientHello, ServerHello, сертифікат сервера, ServerHelloDone, ClientKeyExchange, ChangeCipherSpec, Finished, Renegotiated SSL sessions і Renegotiation settings.

Останні два компоненти стосуються повторного узгодження SSL. Початковий процес рукостискання можна повторно узгодити з іншими криптографічними параметрами, якщо це необхідно. Інші параметри налаштування повторного узгодження включають спосіб обробки запитів на повторне узгодження, період (повторне узгодження через певний період часу), розмір (після певного обсягу даних) і максимальну затримку запису (кількість відкладених записів) тощо [12].

1. Як працює ця вразливість?

Чотири різні типи переговорних процесів можуть відбуватися, якщо початкові зв’язки вже встановлені, і бажано не розривати їх.

Повторне узгодження може бути ініційоване клієнтом, яке може бути безпечним або незахищеним. Найбільш проблемним є незахищене повторне узгодження, ініційоване клієнтом. Більш безпечним варіантом є те, що повторне узгодження ініціюється сервером, що може бути безпечним або незахищеним, оскільки повторне узгодження на стороні клієнта несе внутрішній ризик.

Існує два потенційних сценарії кібератак, пов’язаних із переглядом SSL/TLS:

Атака Man-in-the-Middle (вразливість ін’єкції), яка вставляє шкідливі дані (простий текст можна вставити як префікс до з’єднання TLS) у сеанси HTTPS за допомогою неавтентифікованого запиту (CVE-2009-3555). Роблячи це, зловмисник може виконувати команди з обліковими даними вже авторизованого користувача та навіть збирати облікові дані інших користувачів.

Умова відмови в обслуговуванні запускає сотні рукостискань від клієнтів для того самого TCP-з’єднання, зловживаючи тим фактом, що безпечне з’єднання SSL у 15 разів споживає більше енергії для серверів, ніж для клієнтів.

2. Виявлення вразливості

Група THC підтвердила атаку типу «відмова в обслуговуванні» в 2009 році. Група створила інструмент під назвою THC-SSL-DoS, щоб продемонструвати ідею вразливості. Навіть без повторного узгодження SSL таку DoS-атаку можна здійснити шляхом встановлення нових спроб підключення TCP для кожного рукостискання. Тим не менш, повторне узгодження ще більше полегшує здійснення атаки.

3. Вплив

Оскільки повторне узгодження SSL є поширеною практикою, не слід недооцінювати вразливість, пов’язану з повторним узгодженням, ініційованим клієнтом. Це може вплинути на різні види бізнесу та організації. Ось чому розрахований ступінь серйозності недоліку безпеки є значним.

У той же час, оскільки більшість систем за замовчуванням не допускають повторного узгодження, ініційованого клієнтом, це не повинно бути великою проблемою для поточних користувачів, якщо тільки не дозволено незахищене повторне узгодження або ініційоване клієнтом [12].

Як запобігти незахищеному повторному узгодженню SSL, ініційованому сервером і клієнтом

Як показано в попередніх розділах, небезпечне повторне узгодження SSL, ініційоване клієнтом, становить серйозну загрозу для ваших систем. Проблему

було вирішено в послідовних версіях веб-сервера, таких як останні версії Apache і Nginx. Вони не дозволяють переглядати протокол SSL, ініційований клієнтом.

Це означає, що завжди необхідно переконатися, що веб-сервер оновлений [12].

4. Як вимкнути повторне узгодження

Ви повинні вручну вимкнути ці параметри конфігурації, якщо ваш веб-сервер не запобігає повторному узгодженню SSL, ініційованому клієнтом, за замовчуванням. Керівний принцип полягає в тому, що лише сервер повинен мати дозвіл ініціювати повторне узгодження з'єднання SSL/TLS.

У деяких випадках відключити спробу повторного узгодження клієнта може бути неможливо. Тоді дуже важливо налаштувати лише безпечне повторне узгодження та визначити кількість можливих рукостискань SSL.

5. Як перевірити безпечне повторне узгодження

Чи знаєте ви, чи є ваша система або окрема програма вразливою до загрози повторного узгодження SSL, ініційованого клієнтом, через невиправлені клієнти чи інші проблеми? Ви можете безкоштовно виконати неінвазивне сканування за допомогою програмного забезпечення Crashtest Security для тестування вразливостей. І якщо у вас виникнуть запитання щодо впровадження SSL/TLS або виникають труднощі, ми будемо раді надати рішення для вашого конкретного випадку [12].

Детальний опис атаки (Рис. 3.1):

Крок 1: зловмисник позиціонується між клієнтом і сервером до першого TLS рукостискання

Крок 2: Клієнт починає зв'язок TLS із сервером, зловмисник утримує ці пакети

Крок 3: Зловмисник проходить власне рукостискання TLS із сервером

Крок 4: Зловмисник ініціює запит на повторне узгодження з сервером

Крок 5: Під час запиту на повторне узгодження зловмисник використовує початкове рукостискання клієнта, щоб спілкуватися з сервером

Крок 6: Зловмисник, який тепер діє як проксі, може здійснити атаку на протокол прикладного рівня, наприклад HTTPS

Крок 7: Наприклад, він може додати свій власний запит GET до всього, що надсилає клієнт: напр. Зловмисник надсилає:

GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1

X-Ignore-This і клієнт надсилає:

GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1

Cookie: victimscookie, два запити склеюються, і pizza надсилається до нападника

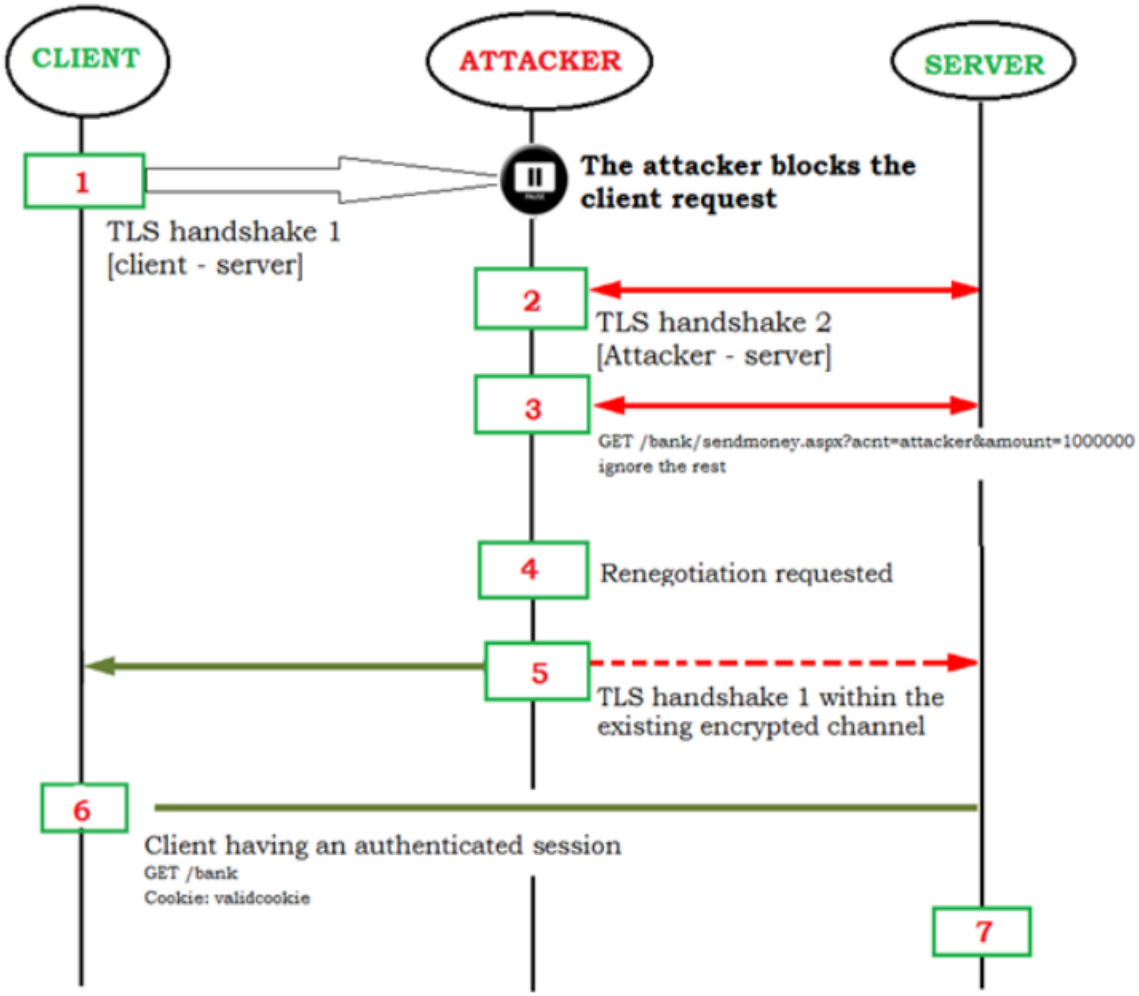

Рис. 3.1. Renegotiation Attack

3.2. BEAST Attack

Основні моменти уразливості BEAST [11]:

- BEAST (Browser Exploit Against SSL/TLS) — це вразливість у способі реалізації ланцюжка блоків шифру у версіях до TLS 1.1
- Тай Дуонг і Джуліано Ріццо виявили в 2011 році
- Використаний вектор ініціалізації (IV) є зашифрованим текстом попереднього блоку
- Зловмисник може використати вибрану атаку відкритого тексту, оскільки IV є передбачуваним
- Зловмисник продовжуватиме пробувати різні відкриті тексти, доки не зможе розшифрувати

Browser Exploit Against SSL/TLS (BEAST) — це атака, яка використовує вразливість у протоколі SSL Transport-Layer Security (TLS) 1.0 і старіших протоколах, використовуючи шифрування в режимі ланцюжка блоків шифру (CBC). Це дозволяє зловмисникам перехоплювати та розшифровувати сесії клієнт-сервер HTTPS і отримувати маркери автентифікації. Він поєднує в собі атаку "людина посередині" (MitM), розділення записів і атаку обраного кордону.

Теоретична вразливість була описана Філліпом Рогавеєм ще в 2002 році, а підтвердження концепції було продемонстровано в 2011 році дослідниками безпеки Тай Дуонгом і Джуліано Ріццо. Атака BEAST схожа на атаки з пониженням версії протоколу, такі як POODLE, оскільки вона також використовує підхід MITM і використовує вразливості в CBC.

Хоча ймовірність цієї атаки дуже низька, і її можна в кращому випадку використовувати для читання коротких рядків відкритого тексту, це одна в ряду багатьох атак, які використовують уразливості CBC. Більше того, його потенційно можна використовувати разом із атакою на пониження версії, наприклад у POODLE, щоб змусити сервер повернутися до TLS 1.0 або старішої версії.

Сервер, який використовує TLS 1.0 із блоковими шифрами, вразливий до BEAST. Це пов’язано з властивою слабкістю CBC, яка дозволяє зловмисникам діяти як людина посередині та виконувати решту атак.

Щоб атака BEAST була можлива, потрібно виконати кілька умов:

- Зловмисник повинен мати можливість відстежувати мережеве з’єднання та сесії між клієнтом і сервером
- Повинна використовуватися вразлива версія, наприклад TLS 1.0 або старіший протокол SSL (або має бути можливим повернення до попередньої версії) з блоковим шифром
- Аплет або ін’єкція JavaScript має бути вірогідною, замінюючи політику того самого походження веб-сайту

Через особливі умови, необхідні для запуску BEAST, малоймовірно, що це вдасться. Тому це не дуже практична атака. Крім того, веб-додатки тепер захищені за замовчуванням через політику того самого походження в сучасних браузерах. Якщо політику не обійти якимось чином, наприклад, за допомогою заголовків CORS на стороні сервера або іншим способом, виконати ін’єкцію буде неможливо.

Тим не менш, не зовсім виключено, що ця атака може бути виконана, особливо на застарілі системи, знайдені в організаційних інтрамережах. Крім того, корисно знати про BEAST через простий факт розуміння того, як зловмисники можуть використовувати багатосторонній підхід, і підкреслити необхідність своєчасного виправлення та заходів [13].

1. Як працює атака BEAST?

BEAST покладається на передбачуваний характер того, як генеруються вектори ініціалізації як частина процесу шифрування в режимі CBC. Завдяки передбачуваності цього процесу та фіксованому розміру шифрів (тобто блоків), зловмисники можуть маніпулювати межами блоків шифру (вибрана частина атаки на межі BEAST) і повільно розкривати відкритий текст без фактичної необхідності розшифровувати його шляхом отримання ключа.

Для розуміння цього процесу потрібне коротке пояснення того, як працюють блокові шифри.

TLS, блокові шифри та вектори ініціалізації

TLS використовує в основному набори шифрів із симетричним шифруванням і блоковими шифрами. Симетричне шифрування означає, що для шифрування та дешифрування інформації використовується один і той же ключ. Однак для більшого захисту спочатку використовується асиметричний механізм шифрування під час процесу узгодження між браузером і сервером для отримання спільного ключа. Після узгодження спільного ключа шифрування виконується симетричним способом, що є швидшим.

Блокові шифри називаються так, тому що вони шифрують дані в блоках фіксованої довжини (8 байт, якщо бути точним). Якщо шифрується інформація, менша за всю довжину блоку, довжина, що залишилася, доповнюється випадковим блоком даних. Поширені типи блокових шифрів включають DES, AES і 3DES.

Щоб зробити дешифрування даних більш складним і безпечним, CBC використовує так званий вектор ініціалізації (IV). Без вектора ініціалізації одні й ті самі дані завжди створюватимуть той самий блок зашифрованого тексту після шифрування, піддаючи його атаці відкритого тексту. Щоб ввести випадковість у рівняння, перший блок зашифрованих даних поєднується з IV (випадковими даними), який потім шифрується узгодженим ключем і генерує блок зашифрованого тексту.

Після цього кожен наступний блок використовує зашифрований текст попереднього блоку як IV. Потім він об’єднує його з відкритим текстом повідомлення за допомогою так званого XOR (логічна операція з’єднання блоків разом, звідси й назва «ланцюжок блоків шифру»). Потім усе це шифрується узгодженим ключем.

Це з’єднання блоків використовується замість генерації випадкового IV для кожного повідомлення, але воно робить кожне наступне IV передбачуваним і

відомим, оскільки це вивід попереднього блоку зашифрованого тексту. Крім того, XOR є оборотною операцією, яка вводить ще один елемент уразливості. Ця передбачуваність є основою BEAST [13].

2. Запуск атаки BEAST

Припустимо, що зловмисник може «пронюхати» обмін повідомленнями між клієнтом і сервером. Враховуючи також те, що сервер використовує TLS 1.0 або SSL і що зловмисник здатний обманом змусити користувача запустити JavaScript або аплет через шкідливий веб-сайт.

Це призведе до введення зловмисником блоків даних у сеанс. Вони обидва матимуть IV повідомлення та XOR його з блоком відкритого тексту, який вони хочуть додати. Потім вони могли б надіслати їх на сервер і спостерігати за відповіддю сервера, тобто запустити атаку «людина посередині» та виконати так звану атаку з розділенням записів. Таким чином вони можуть отримати доступ до інформації, якою обмінюються веб-сервери та браузери, як-от паролі, номери кредитних карток тощо.

Спочатку, під час атаки BEAST, проблема полягала в тому, що здавалося можливим лише вгадати цілий блок зашифрованого тексту. Але, на жаль, вгадування цілого блоку даних, навіть лише 8-бітного блоку, є складним завданням, яке може вимагати 2568 спроб. Це зробило BEAST теоретичною атакою, яка здавалася нездійсненною, якщо не неможливою.

Але, як виявили Тай Дуонг і Джуліано Ріццо в 2011 році, можливий інший підхід. Дует зробив це замість того, щоб намагатися вгадати весь блок. Вони перемістили межі блоку шифру, виділивши лише один байт. Як наслідок, вгадування одного байта значно легше, оскільки воно обмежує максимальну кількість спроб вгадування однієї цифри числа, наприклад, до 10.байт, при цьому межі зсуваються після кожного успішного вгадування. Це обрана прикордонна частина атаки [13].

3. Чи ймовірно, що ви постраждаєте від уразливості BEAST?

Враховуючи багато умов, необхідних для виконання атаки BEAST, навряд чи вас атакуватимуть таким чином. Крім того, якщо зловмисникам вдасться розташувати себе таким чином, щоб відповідати всім вимогам, вони матимуть набагато більше можливостей для компрометації вашої системи та даних користувача.

Оскільки BEAST використовує вразливості, які використовуються в інших атаках, важливо вжити заходів, щоб уникнути викриття. Наприклад, якщо ваш сервер підтримує TLS 1.0 або будь-яку версію SSL, буде присутній безліч уразливостей. Дивіться розділ нижче, щоб отримати коротку інформацію щодо запобігання можливості атаки BEAST [13].

4. Як запобігти нападу BEAST?

Спочатку було рекомендовано перейти на потоковий шифр RC4, щоб швидко запобігти вразливостям, пов’язаним із CBC у TLS. Але в 2013 році шифр RC4 виявився небезпечним у багатьох відношеннях. Тому через два роки його використання в реалізаціях TLS було заборонено.

Наразі найпростіший і найефективніший спосіб запобігти атаці BEAST — вимкнути підтримку TLS 1.0 і 1.1., а також SSL на сервері. Це також захищає від інших недоліків безпеки, які використовують уразливості в SSL і попередніх версіях TLS, таких як POODLE або DROWN [13].

3.3. CRIME/BREACH Attack

Основні моменти CRIME/BREACH Attack [11]:

- Вперше теоретично описана Джоном Келсі в 2002 році
- Атака CRIME (Compression Ratio Info-leak Made Easy) використовує стиснення, що використовується в TLS
- Стиснення працює за рахунок зменшення надмірності даних

- У випадках, коли використовується стиснення, можна побачити зміни розміру
- Вибрану атаку відкритого тексту можна змонтувати та спостерігати за розмірами файлів, щоб побачити, чи знайдено відповідність
- BREACH можна пом'якшити, вимкнувши стиснення
- BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext) була тією самою атакою на основі стиснення проти самого протоколу HTTP
- Важко пом’якшити, оскільки широко використовується стиснення HTTP

Візуалізація даної атаки зображена на Рис 3.2 і Рис 3.3 [11]:

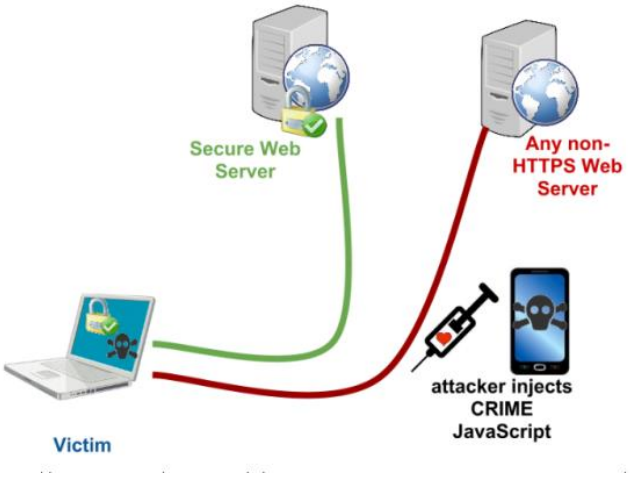

Рис. 3.2. Ін'єкція CRIME

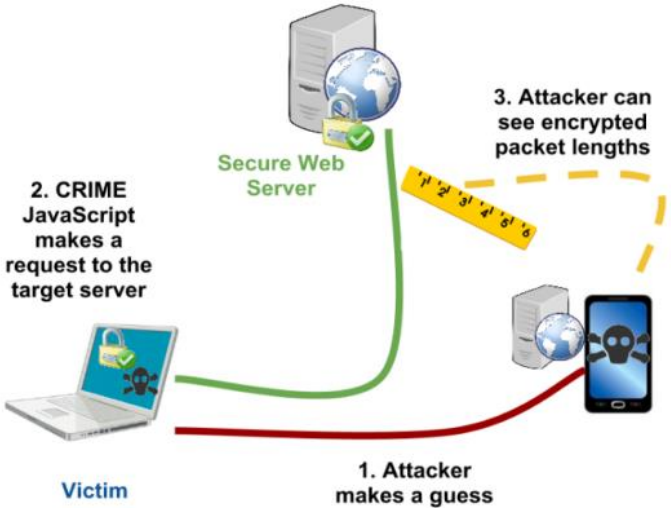

Рис. 3.3. Використання CRIME

3.4. Padding Attacks – Атака заповнення

Основні моменти атак заповнення:

- Оригінальна атака виявлена Сержем Водене у 2002 році
- Довіра серверу щодо витоку інформації про успіх/невдалість перевірки заповнення
- Зловмисник вводить власний відкритий текст і підтверджує, чи сталася помилка заповнення або MAC
- Це дозволяє зловмиснику розшифровувати/шифрувати повідомлення без знання ключа
- Архітектор TLS спочатку відповів, змінивши повідомлення про помилку, яке генерується, коли виникає помилка
- Диференціал часу залишився
- Відзначаючи різницю в часі, необхідному для дешифрування, зловмисники могли отримати ту саму інформацію, що й раніше
- TLS 1.2 вимагає, щоб навіть якщо сталася помилка заповнення, сервер припускає відсутність заповнення та обчислює MAC усього повідомлення
- Все ще існує дуже мала різниця в часі
- Атака Lucky 13

Кроки атак заміщення:

Крок 1: зловмисник скорочує зашифроване повідомлення, щоб відкритий текст, який потрібно розшифрувати, знаходився в останньому блоці, де зазвичай очікується доповнення.

Крок 2: потім зловмисник змінює передостанній зашифрований блок, останній байт якого обробляється XOR з припущенням відкритого тексту.

Крок 3: дешифрування CBC спробує розшифрувати цей байт і якщо це вийде, він створить 0x00, який, у свою чергу, породить помилку MAC.

Крок 4: перевірка MAC-адреси в TLS 1.0 займає більше часу, ніж помилка заповнення, щоб створити сповіщення, і таким чином зловмисник може визначити, чи його припущення про заповнення було успішним.

Крок 5: якщо зловмисник встановив, що останній байт модифікованого зашифрованого тексту створює дійсне доповнення з певним значенням, і він знає, що значення відкритого тексту дорівнює 01 (це значення доповнення), тоді він може змінити операцію та підтвердити проміжний стан.

Крок 6: Використовуючи проміжний стан, він може подати фактичний зашифрований текст і розшифрувати його у відкритий текст.

Атака padding oracle є вражаючою атакою, оскільки вона дозволяє розшифрувати повідомлення, яке було перехоплено, якщо воно було зашифровано в режимі CBC. POODLE (Padding Oracle On Downgraded Legacy Encryption) — це експлойт "людина посередині", який використовує переваги клієнта програмного забезпечення для безпеки в Інтернеті та повернення до SSL 3.0. Якщо зловмисники успішно скористаються цією вразливістю, їм у середньому потрібно зробити лише 256 запитів SSL 3.0, щоб відкрити один байт зашифрованих повідомлень.

Потрібно лише переконатися, що ми зможемо отримати відповідь від сервера, який буде виконувати роль Oracle (ми повернемося до них більш детально пізніше в цьому звіті). Тоді ми зможемо розшифрувати все повідомлення, крім першого блоку, якщо не знаємо вектор ініціалізації [13].

Oracle заповнення — це функція програми, яка розшифровує зашифровані дані, надані клієнтом, наприклад стан внутрішнього сеансу, що зберігається на клієнті, і витікає стан дійсності доповнення після дешифрування. Існування Oracle заповнення дозволяє зловмиснику розшифровувати зашифровані дані та шифрувати довільні дані без знання ключа, який використовується для цих криптографічних операцій. Це може призвести до витоку конфіденційних даних або до вразливості підвищення привілеїв, якщо програма передбачає цілісність зашифрованих даних.

Блокові шифри шифрують дані лише блоками певного розміру. Розміри блоків, які використовуються звичайними шифрами, становлять 8 і 16 байт. Дані, розмір яких не відповідає кратному розміру блоку використаного шифру, мають бути доповнені певним чином, щоб дешифратор міг видалити доповнення. Часто використовуваною схемою доповнення є PKCS#7. Він заповнює решту байтів значенням довжини доповнення [16].

Приклад:

Якщо заповнення має довжину 5 байтів, значення байта 0x05 повторюється п’ять разів після звичайного тексту.

Помилка наявна, якщо заповнення не відповідає синтаксису використаної схеми заповнення. Oracle заповнення присутній, якщо програма витікає з цієї конкретної умови помилки заповнення для зашифрованих даних, наданих клієнтом. Це може статися через пряме виявлення винятків (наприклад, BadPaddingException у Java), через тонкі відмінності у відповідях, надісланих клієнту, або через інший бічний канал, як-от поведінка синхронізації.

Певні режими роботи криптографії допускають атаки з перевертанням біта, коли перевертання біта в зашифрованому тексті призводить до того, що біт також перевертається у відкритому тексті. Перевертання біта в n-му блоці зашифрованих даних CBC призводить до того, що той самий біт у (n+1)-му блоці перевертається в розшифрованих даних. n-й блок розшифрованого шифрованого тексту смітить цією маніпуляцією.

Атака за допомогою Oracle заповнення дає змогу зловмиснику розшифрувати зашифровані дані, не знаючи ключа шифрування та використаного шифру, надсилаючи вміло оброблені тексти шифру до Oracle заповнення та спостерігаючи за результатами, які він повертає. Це призводить до втрати конфіденційності зашифрованих даних. наприклад у випадку даних сесії, що зберігаються на стороні клієнта, зловмисник може отримати інформацію про внутрішній стан і структуру програми.

Атака Oracle заповнення також дозволяє зловмиснику шифрувати довільні прості тексти без знання використовуваного ключа та шифру. Якщо програма припускає, що цілісність і автентичність розшифрованих даних надано, зловмисник зможе маніпулювати станом внутрішнього сеансу та, можливо, отримати вищі привілеї [16].

Висновки до розділу 3

Отже, основні атаки на протоколи SSL/TLS з моменту їх виникнення і до сьогодні включають BEAST, CRIME, BREACH, Renegotiation Attack, Padding Attacks.

Основні деталі по найбільш відомим вразливостям і атакам:

- BEAST було успішно пом’якшено для версій 1.1+, і його також важко виконати
- CRIME було пом'якшено в нових версіях браузера
- BREACH все ще можливо реалізувати, і йому важко протидіяти
- Атаки повторного узгодження можна пом’якшити за допомогою бібліотеки, яка містить розширення, запропоноване в RFC 5746
- Все ще можливі атаки заповнення

ВИСНОВКИ

Отже, в ході виконання кваліфікаційної роботи було досягнуто поставленої мети – проаналізовано і визначено актуальні вразливості та атаки на мережеві комунікації, захищені з використанням протоколів SSL/TLS, та наведено деякі заходи протидії.

Також виконано поставлені задачі:

1. Розглянуто місце протоколу SSL/TLS в стеку протоколів захисту мережі, досліджено історію розвитку SSL/TLS та принцип роботи. Визначено, що хоча назви SSL і TLS використовуються взаємозамінно, фактично TLS є новою версією SSL, а сам SSL є застарілим і вразливим та не використовується з 2014 року.

2. При дослідженні нормативно-правових документів визначено, що використання TLS і правильне його налаштування так чи інакше є вимогою основних міжнародних стандартів. З найбільш відомих стандартів правильне налаштування використання SSL/TLS регламентують такі нормативні документи як HIPAA, NIST SP 800-52r2, PCI-DSS.

3. В ході аналізу актуальних вразливостей протоколів SSL/TLS визначено такі основні загрози – BREACH, атаки повторного узгодження, атаки заповнення. Дані загрози необхідно враховувати при побудові процесу захищеного обміну даними з використанням SSL/TLS.

Висновки щодо актуальних вразливостей і атак на протоколи SSL/TLS можуть бути використані будь якою організацією або суб’єктом для врахування при побудові захищених мережевих комунікацій, при визначенні найбільш безпечних налаштувань використання протоколів SSL/TLS.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 6 Network Security Protocols You Should Know. URL: https://www.catonetworks.com/network-security/network-security-protocols/
2. Модель OSI. URL: https://e-tk.lntu.edu.ua/mod/page/view.php?id=3544&forceview=1
3. The Evolution of SSL and TLS. URL: https://www.digicert.com/blog/evolution-of-ssl
4. How does SSL work? | SSL certificates and TLS. URL: https://www.cloudflare.com/learning/ssl/how-does-ssl-work/
5. Що таке TLS і як він працює? URL: https://instagalleryapp.com/informacijna-bezpeka/shho-take-tls-i-jak-vin-pracjue/
6. Що таке TLS. URL: https://habr.com/ru/post/258285/
7. Що таке протокол TLS? URL: https://www.vpnunlimited.com/ru/help/vpn-protocols/tls-protocol
8. Difference between Secure Socket Layer (SSL) and Transport Layer Security (TLS). URL: https://www.geeksforgeeks.org/difference-between-secure-socket-layer-ssl-and-transport-layer-security-tls/
9. Differences between SSL and TLS Protocol Versions. URL: https://www.wolfssl.com/differences-between-ssl-and-tls-protocol-versions-6/
10. Guide to TLS Standards Compliance. URL: https://www.ssl.com/guide/tls-standards-compliance/
11. SSL/TLS: HISTORY AND VULNERABILITIES. URL: https://people-ece.vse.gmu.edu/coursewebpages/ECE/ECE646/F19/project/F15_presentations/Session_4_SSL_TLS/1_SSL_TLS_Attacks_1.pdf
12. What Is the SSL Renegotiation Vulnerability? URL: https://crashtest-security.com/secure-client-initiated-ssl-renegotiation/
13. What Is the SSL BEAST Attack and How Does It Work. URL: https://crashtest-security.com/ssl-beast-attack-tls/

14. Padding Oracle Attack Explained. URL: https://flast101.github.io/padding-oracle-attack-explained/

15. OWASP Testing for Padding Oracle. URL: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/02-Testing_for_Padding_Oracle