НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота 1

Дослідження реалізацій протоколу SSL

Виконали:

Галіца О.О.

Литвиненко Ю.С.

Паршин О.Ю.

ФІ-22мн

Перевірила:

Байденко П.В.

Київ 2023

Стек протоколів TCP/IP.

Стек протоколів TCP/IP (Transmission Control Protocol/Internet Protocol, протокол управління передачею/міжмережевий протокол) — мережева модель, що описує процес передачі цифрових даних. Вона названа за двома головними протоколами, за цією моделлю побудовано глобальну мережу інтернет. Зараз це здається неймовірним, але у 1970-х інформація не могла бути передана з однієї мережі до іншої. Щоб забезпечити таку можливість, було розроблено стек інтернет-протоколів, відомий як TCP/IP.

Розробка мережевої моделі здійснювалася за сприяння Міністерства оборони США, тому іноді модель TCP/IP називають DoD (Department of Defence) модель. Модель TCP/IP має поділ на рівні, всередині яких діють певні протоколи та виконуються власні функції. Документами, що визначають сертифікацію моделі, є RFC-1122 та RFC-1123. Ці стандарти описують чотири рівні абстракції моделі TCP/IP: прикладний, транспортний, міжмережевий та канальний. Існують і інші версії опису моделі, у тому числі ті, які включають іншу кількість рівнів та їх найменувань. Однак ми дотримуємося оригінальної версії і розглянемо далі чотири рівні моделі.

Канальний рівень (link layer)

Призначення канального рівня — дати опис того, як відбувається обмін інформацією на рівні мережевих пристроїв, визначити, як інформація передаватиметься від одного пристрою до іншого. Інформація кодується, ділиться на пакети і відправляється по потрібному каналу.

Цей рівень також обчислює максимальну відстань, на яку можна передати пакети, частоту сигналу, затримку відповіді і т.д. Усе це — фізичні властивості середовища передачі. На канальному рівні найпоширенішим протоколом є Ethernet.

Міжмережевий рівень (internet layer)

Глобальна мережа інтернет складається з безлічі локальних мереж, що взаємодіють між собою. Міжмережевий рівень використовується для забезпечення такої взаємодії.

Міжмережева взаємодія — це основний принцип побудови інтернету. Локальні мережі по всьому світу об'єднані у глобальну, а передачу даних між цими мережами здійснюють маршрутизатори.

Маска підмережі допомагає маршрутизатору зрозуміти, як і куди передавати пакет. Підмережею може бути будь-яка мережа зі своїми протоколами. Маршрутизатор передає пакет безпосередньо, якщо одержувач знаходиться у тій самій підмережі, що й відправник. Якщо підмережі одержувача і відправника розрізняються, пакет передається на другий маршрутизатор, з другого на третій і далі по ланцюжку, поки не досягне одержувача.

Протокол *IP* (Internet Protocol) використовується маршрутизатором, щоб визначити, якої підмережі належить одержувач. Свою унікальну IP-адресу має кожен мережевий пристрій, при цьому в глобальній мережі не може існувати двох пристроїв з однаковим IP. Протокол має дві версії, перша з яких – IPv4 (IP version 4) – була описана в 1981 році.

IPv4 передбачає призначення кожному пристрою 32-бітної IP-адреси, що обмежує максимально можливу кількість унікальних адрес 4 мільярдами (2^{32}). У більш звичному для людини десятковому вигляді IPv4 виглядає як чотири блоки (октети) чисел від 0 до 255 розділених трьома точками.

У зв'язку зі швидким зростанням мережі інтернет гостро постала необхідність збільшення кількості можливих IP-адрес. У 1995 році вперше був описаний протокол IPv6 (IP version 6), який використовує 128-бітові адреси та дозволяє призначити унікальні адреси для 2^{128} пристроїв. IPv6 має вигляд восьми блоків по чотири шістнадцяткових значення, а кожен блок поділяється двокрапкою.

IP призначений для визначення адресата та доставки йому інформації. Він надає послугу для вищих рівнів, але не гарантує цілісність інформації, що доставляється.

Також варто згадати *ICMP*. ICMP в основному використовується пристроями в мережі для доставки повідомлень про помилки та операційну інформацію, що повідомляє про успіх або помилку при зв'язку з іншим пристроєм. Наприклад, саме з використанням ICMP здійснюється передача звітів про недоступність пристроїв у мережі. Крім того, ICMP використовується при діагностиці мережі, наприклад, в експлуатації утиліт ping або traceroute.

ICMP, аналогічно IP, працює на міжмережевому рівні і є невід'ємною частиною при реалізації моделі TCP/IP. Варто зазначити, що для різних версій IP використовуються різні версії протоколу ICMP.

Транспортний рівень (transport layer)

Постійні резиденти транспортного рівня – протоколи TCP та UDP, вони займаються доставкою інформації.

TCP (протокол управління передачею) — надійний, він забезпечує передачу інформації, перевіряючи, чи дійшла вона, наскільки повним є обсяг отриманої інформації і т.д. TCP дозволяє двом кінцевим пристроям здійснювати обмін пакетами через попередньо встановлене з'єднання. Він повторно запитує втрачену інформацію, усуває дублюючі пакети, регулюючи завантаженість мережі. TCP гарантує отримання та збирання інформації у адресата в правильному порядку.

UDP (протокол датаграм користувача) — ненадійний, він займається передачею автономних датаграм. UDP не гарантує, що всі датаграми дійдуть до одержувача. Датаграми вже містять всю необхідну інформацію, щоб дійти до одержувача, але вони все одно можуть бути втрачені або доставлені в порядку, відмінному від порядку при відправленні.

UDP зазвичай не використовується, якщо потрібна надійна передача інформації. Використання UDP має сенс там, де втрата частини інформації не буде критичною для програми, наприклад, у відеоіграх або потоковій передачі відео. UDP необхідний, коли робити повторний запит складно чи невиправдано з якихось причин.

При передачі протоколу TCP дані діляться на сегменти. Сегмент — це частина пакету. Коли надходить пакет даних, який перевищує пропускну спроможність мережі, пакет ділиться на сегменти допустимого розміру. Сегментація пакетів також потрібна в ненадійних мережах, коли існує велика ймовірність того, що великий пакет буде втрачено. При передачі даних протоколу UDP пакети даних діляться вже на датаграми. Датаграма (datagram) — це також частина пакета, але її не можна плутати із сегментом. Головна відмінність датаграм в автономності. Кожна датаграма містить усі необхідні заголовки, щоб дійти кінцевого адресата, тому вони не залежать від мережі, можуть доставлятися різними маршрутами й у різному порядку. При втраті датаграм чи сегментів виходять «биті» шматки даних, які вийде коректно обробити.

Протоколи транспортного рівня не інтерпретують інформацію, отриману з верхнього чи нижніх рівнів, вони служать лише каналом передачі, але є винятки. RSVP (Resource Reservation Protocol, протокол резервування мережевих ресурсів) може використовуватися, наприклад, роутерами з метою аналізу трафіку та прийняття рішень про його передачу або відхилення залежно від вмісту.

Прикладний рівень (application layer)

У моделі TCP/IP відсутні додаткові проміжні рівні на відміну від OSI, про яку ми поговоримо трохи нижче. Функції форматування та представлення даних делеговані бібліотекам та програмним інтерфейсам додатків (API) – свого роду базам знань, що містять відомості про те, як програми взаємодіють між собою. Коли служби або програми звертаються до бібліотеки або API, ті у відповідь надають набір дій, необхідних для виконання завдання та повну інструкцію, як ці дії потрібно виконувати.

Протоколи прикладного рівня діють для більшості додатків, вони надають послуги користувачам чи обмінюються даними з «колегами» з нижніх рівнів. Тут для більшосты програм створено свої протоколи. Наприклад, браузери використовують HTTP для передачі гіпертексту по мережі, поштові клієнти — SMTP для передачі пошти, FTP-клієнти — протокол FTP для передачі файлів, служби DHCP — протокол призначення IP-адрес DHCP і так далі.

Мережева модель OSI.

Мережева модель OSI має 7 рівнів, ієрархічно розташованих від більшого до меншого. Найвищим є сьомий (прикладний), а найнижчим — перший (фізичний). Модель OSI розроблялася ще у 1970-х роках, щоб описати архітектуру та принципи роботи мереж передачі даних.

У процесі передачі даних завжди беруть участь пристрій-відправник, пристрій-одержувач, а також самі дані, які мають бути передані та отримані. З погляду рядового користувача завдання елементарне — потрібно взяти та надіслати ці дані. Все, що відбувається при надсиланні та прийомі даних, детально описує семирівневу модель OSI.

Усі сім рівнів моделі OSI можна умовно поділити на дві групи:

– Media layers (рівні середовища),

– Host layers (рівні хоста).

Рівні групи Media Layers (L1, L2, L3) займаються передачею інформації (кабелем або бездротовою мережею), використовуються мережевими пристроями, такими як комутатори, маршрутизатори і т.п. Рівні групи Host Layers (L4, L5, L6, L7) використовуються безпосередньо на пристроях, будь то стаціонарні комп'ютери або мобільні пристрої. Кожен рівень має свої PDU (Protocol Data Unit), представлені у тій формі, яка зрозуміла цьому рівні і, можливо, наступному до перетворення. Робота з чистими даними відбувається лише на рівнях з п'ятого до сьомого. Розглянемо рівні трохи докладніше.

Фізичний рівень (physical layer, L1)

Почнемо з найнижчого рівня. Він відповідає за обмін фізичними сигналами між фізичними пристроями, "залізом".

Пристрої фізичного рівня оперують бітами. Вони передаються кабелями (наприклад, через оптоволокно) чи, наприклад, через Bluetooth чи IRDA, Wi-Fi, GSM, 4G тощо.

Канальний рівень (data link layer, L2)

Коли два користувача знаходяться в одній мережі, що складається лише з двох пристроїв, це ідеальний випадок. Але що якщо цих пристроїв більше?

Другий рівень вирішує проблему адресації під час передачі інформації. Канальний рівень отримує біти і перетворює їх на кадри (frame, також «фрейми»). Завдання тут — сформувати кадри з адресою відправника та одержувача, після чого відправити їх по мережі.

У канального рівня є два підрівні — це MAC і LLC. MAC (Media Access Control, контроль доступу до середовища) відповідає за присвоєння фізичних MAC-адрес, а LLC (Logical Link Control, контроль логічного зв'язку) займається перевіркою та виправленням даних, керує їхньою передачею. Якщо бути точними, LLC не можна віднести повністю ні до першого, ні до другого рівня — він знаходиться між ними.

На другому рівні OSI працюють комутатори, їх завдання — передати сформовані кадри від одного пристрою до іншого, використовуючи як адреси лише фізичні MAC-адреси.

На каналі активно використовується протокол ARP (Address Resolution Protocol — протокол визначення адреси). За допомогою нього 64-бітові MAC-адреси зіставляються з 32-бітними IP-адресами і навпаки, тим самим забезпечується інкапсуляція та декапсуляція даних.

Мережевий рівень (network layer, L3)

На третьому рівні з'являється нове поняття — маршрутизація. Для цього завдання було створено пристрої третього рівня — маршрутизатори (їх ще називають роутерами). Маршрутизатори отримують MAC-адресу від комутаторів з попереднього рівня та займаються побудовою маршруту від одного пристрою до іншого з урахуванням усіх потенційних несправностей у мережі.

Транспортний рівень (transport layer, L4)

Четвертий рівень — це посередник між Host Layers та Media Layers, що належить швидше до перших, ніж до останніх. Його головним завданням є транспортування пакетів. При транспортуванні можливі втрати, але деякі типи даних більш чутливі до втрат, ніж інші. Про протоколи TCP та UDP ми вже згадували раніше.

Сеансовий рівень (session layer, L5)

П'ятий рівень оперує чистими даними. Крім п'ятого, чисті дані використовуються також на шостому та сьомому рівні. Сеансовий рівень відповідає за підтримку сеансу чи сесії зв'язку. П'ятий рівень керує взаємодією між програмами, відкриває можливості синхронізації завдань, завершення сеансу, обміну інформацією.

Служби сеансового рівня найчастіше застосовуються серед додатків, потребують віддаленого виклику процедур, тобто щоб вимагати виконання дій на віддалених комп'ютерах або незалежних системах на одному пристрої (за наявності кількох ОС).

Рівень подання даних (presentation layer, L6)

Про завдання рівня подання даних свідчить його назва. Шостий рівень відповідає за перетворення протоколів та кодування/декодування даних. Шостий рівень також займається представле-

нням картинок (JPEG, GIF і т.д.), а також відео-аудіо (MPEG, QuickTime). А також шифруванням даних, коли під час передачі їх необхідно захистити.

Прикладний рівень (application layer)

Прикладний рівень - це те, з чим взаємодіють користувачі, свого роду графічний інтерфейс усієї моделі OSI. Протоколам сьомого рівня не потрібно забезпечувати маршрутизацію або гарантувати доставку даних, коли про це вже подбали попередні шість. Завдання сьомого рівня – використовувати свої протоколи, щоб користувач побачив дані у зрозумілому йому вигляді.

Наступний малюнок показує співвідношення рівнів двох розглянутих моделей.

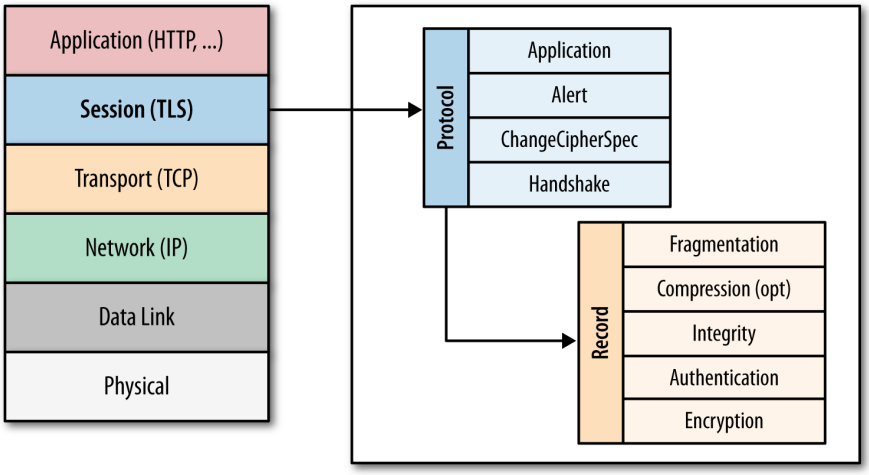
<table>
<tr><th>Модель OSI</th><th></th><th></th><th>Модель TCP/IP</th></tr>
<tr><td>Прикладний рівень (application layer)</td><td>7</td><td rowspan="3">4</td><td rowspan="3">Прикладний рівень (application layer)</td></tr>
<tr><td>Рівень подання даних (presentation layer)</td><td>6</td></tr>
<tr><td>Сеансовий рівень (session layer)</td><td>5</td></tr>
<tr><td>Транспортний рівень (transport layer)</td><td>4</td><td>3</td><td>Транспортний рівень (transport layer)</td></tr>
<tr><td>Мережевий рівень (network layer)</td><td>3</td><td>2</td><td>Міжмережевий рівень (internet layer)</td></tr>
<tr><td>Канальний рівень (data link layer)</td><td>2</td><td rowspan="2">1</td><td rowspan="2">Канальний рівень (link layer)</td></tr>
<tr><td>Фізичний рівень (physical layer)</td><td>1</td></tr>
</table>

На відміну від TCP/IP, OSI ніколи не асоціювалася з UNIX. Домогтися широкого поширення OSI не вдалося тому, що вона проектувалася як закрита модель, що просувається Європейськими телекомунікаційними компаніями та урядом США. Стек протоколів TCP/IP спочатку був відкритий всім, що дозволило йому набрати популярності серед прихильників відкритого програмного коду.

SSL vs TLS.

SSL і TLS це розвиток однієї і тієї ж технології. Абревіатура TLS (Transport Layer Security) з'явилася як заміна позначення SSL (Secure Sockets Layer) після того, як протокол остаточно став інтернет-стандартом. Така заміна викликана юридичними аспектами, оскільки специфікація SSL спочатку належала компанії Netscape. І зараз нерідко назви SSL і TLS продовжують використовувати синоніми, але канонічним ім'ям є TLS, а протоколи сімейства SSL давно остаточно застаріли і не повинні використовуватися.

Протокол TLS (SSL) знаходиться між транспортним та прикладним рівнями. Конкретне місце TLS (SSL) у стеку протоколів Інтернету показано на схемі:



TLS/SSL спочатку розробили для захисту комерційних транзакцій через Інтернет. Тобто, основною метою було отримання щодо безпечного каналу для здійснення покупок або управління банківським рахунком — хоча ні перше, ні друге ще не набули популярності у пересічних користувачів за часів становлення SSL, оскільки на дворі була середина 90-х років XX століття. Натомість у сучасному Інтернеті на TLS/SSL покладаються не тільки і не стільки в діяльності "з продажу товарів а й при вирішенні набагато загальнішого завдання збереження "приватності"та конфіденційності важливої інформації. Наприклад, TLS використовується і в сучасних месенджерах, однак одним із найпоширеніших застосувань TLS залишається HTTPS. HTTPS швидко витісняє незахищену версію (HTTP) в Інтернеті: основний обсяг веб-трафіку зараз передається в зашифрованому вигляді, при цьому і так невелика частка відкритого трафіку продовжує зменшуватися. У 2022 році вже можна сказати, що майже весь значний веб-трафік зараз зашифрований. У новій версії HTTP/2 захист інформації засобами TLS має використовуватися за умовчанням.

Вперше SSL впровадили як пропрієтарну технологію, реалізовану в браузері Netscape Navigator, одному з перших веб-браузерів. Версія 1 протоколу не була опублікована, а так і залишилася внутрішньою розробкою Netscape, що розвивалася у 1994-95 роках. SSLv2 – наступну, другу версію протоколу – опублікували, проте специфікація так і не вийшла зі стану чернетки (draft). Більше того, хоч у SSLv2 і скоригували окремі дефекти та вразливості v1, протокол виявився ненадійним, що містить серйозні архітектурні огріхи. SSLv2 дуже давно і без застережень визнаний небезпечним, тому зараз не має використовуватися. Якщо постаратися, то в мережі все ще можна знайти архаїчні сервери, які підтримують SSLv2, оскільки витіснення дефектних технологій йде повільно.

SSLv3 — це розвиток SSLv2, але з дуже суттєвими доробками. SSLv3 представлений у 1996 році. Цей протокол отримав повноцінну специфікацію RFC, але значно пізніше за свою появу, і в статусі історичного документа: RFC-6101. На час SSLv3 довелося становлення TLS як інтернет-технології. На базі SSLv3 і з'явився протокол TLS.

З 2015 року SSLv3, слідом за публікацією чергових уразливостей, перейшов у статус нерекомендованого. Цей статус є цілком офіційним, яким тільки офіційним він може бути в рамках технологічних традицій Інтернету: у червні 2015 року випущено документ RFC-7568, який вимагає

виключити SSLv3 з розряду протоколів, що підтримуються клієнтами та серверами. Залишилися лише версії TLS.

Зараз існує чотири версії TLS, всі вони описані в RFC: TLS 1.0, TLS 1.1, TLS 1.2 і версія TLS 1.3, що вийшла в 2018 році. TLS 1.0 багато в чому повторює SSLv3, крім низки деталей, які, втім, є дуже важливими. У криптографічних протоколах деталі важливі як ні в яких інших протоколах Інтернету. Ключовою відмінністю TLS від SSL є наявність цілої низки розширень, що дозволяють реалізувати сучасні методи захисту інформації. TLS 1.1 та 1.2 досить близькі до 1.0, але у версії 1.2 з'явилися помітні поліпшення, наприклад, використовується інша основа псевдовипадкової функції та введена підтримка шифрів у режимі автентифікованого шифрування.

Незважаючи на те, що номер наступної версії отримав лише збільшення одиниці в "молодшому розряді TLS 1.3 істотно відрізняється від усіх попередніх версій TLS. Зокрема, радикально змінено логіку встановлення з'єднання, а сам протокол став архітектурно стрункішим, завдяки тому, що зі специфікації видалено багато повідомлень (елементи протоколу), а решту – помітно перероблено.

TLS. Версії та особливості.

Модель загроз TLS передбачає, що атакуючий може як завгодно втручатися у канал зв'язку, зокрема активно підміняти пакети і взагалі – переривати зв'язок. Ключові завдання TLS:

1. забезпечити *конфіденційність*
   (реалізувати захист від витоків інформації, що передається);

2. реалізувати *збереження цілісності інформації*
   (забезпечити виявлення підміни інформації, що передається);

3. забезпечити *аутентифікацію* вузлів
   (дати механізм автентифікації джерела повідомлень).

З фундаментальних завдань захисту інформації, TLS не охоплює лише одну: забезпечення доступності інформації - і це завдання далеко за рамками даного протоколу.

Крім того, у TLS 1.3 суттєва увага приділена задачі приховування метаінформації (це підзавдання з першого пункту, забезпечення конфіденційності), під метаінформацією розуміється сукупність таких відомостей про TLS-з'єднання, які дозволяють зі сторони судити про дані, що передаються в захищеному режимі. Наприклад, до метаінформації відносяться відомості про відкриті криптографічні ключі вузлів, час з'єднання, адреси, імена вузлів і так далі. Суттєві зусилля були спрямовані на те, щоб навіть добре оснащена третя сторона не могла дізнатися нічого більше, крім факту встановлення TLS-з'єднання.

TLS якось справляється з описаними щойно завданнями, але слід вважати, що TLS вирішує їх повністю. Така думка є великою помилкою, на жаль, дуже поширеною. Доказом того, що TLS не вирішує ці завдання повністю, є вразливості, виявлені в самому протоколі (а не тільки в його реалізаціях). Протокол та її реалізації намагаються вирішити описані завдання максимально доступному рівні надійності. Однак TLS в цілому не має доведеної стійкості, як не мають її і багато найважливіших складових протоколу.

TLS працює із *записами (records)*. Записи перебувають у фундаменті протоколу. У деяких випадках TLS-повідомлення вимагають складання кількох записів, що істотно впливає на логіку обробки станів протоколу. Кожний TLS-запис є блоком, що складається з короткого заголовка і, власне, самих даних.

У TLS, за деякими винятками, використовується *формат*, у якому представлення структури даних починається із заголовка, що позначає тип даних та їх довжину. Частини цього заголовка, що визначають структуру, завжди чітко визначені Тобто, визначено кількість байтів, виділених для запису типу, перераховані всі допустимі значення цих байтів і спосіб їх інтерпретації; визначено кількість байтів для запису довжини корисного навантаження, спосіб інтерпретації запису.

Заголовок завжди передається у відкритому вигляді. У тому числі при використанні TLS 1.3. Однак TLS 1.3 передбачає приховування реального типу запису (для захищених записів). У TLS до версії 1.3 можливе використання *стиснення даних*. Під "стисненням"тут мається на увазі використання кодування, що мінімізує кількість бітів, необхідних для запису даних — той самий процес, який багатьом знайомий за програмами-архіваторами. У TLS 1.3 стискання даних прямо заборонено специфікацією, тому дане "доповнення"не використовується.

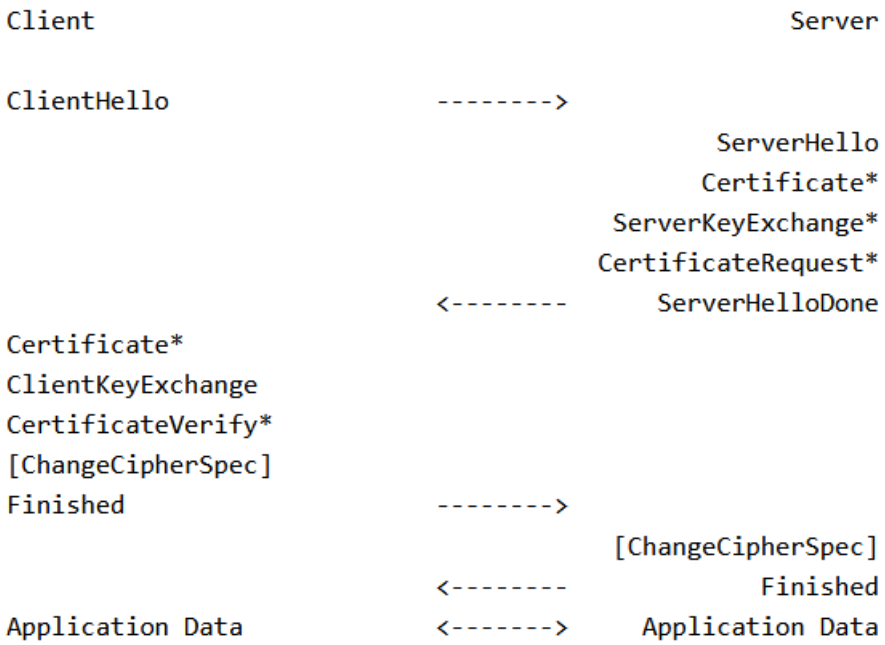
Код аутентифікації повідомлення — найважливіший елемент захисту інформації у TLS. У версіях до 1.3 зазвичай використовується HMAC — алгоритм обчислення коду аутентифікації, що базується на тій чи іншій хеш-функції (типовий вибір: SHA-1 або SHA-256). Код автентифікації приписується до даних. Один з архітектурних дефектів версій TLS до 1.3 полягає в тому, що відповідні специфікації наказують спочатку обчислювати MAC, а потім зашифровувати повідомлення. Тобто обчислений код автентифікації приєднується до відкритого тексту повідомлення, а потім все разом зашифровується. При цьому сторона повинна спочатку розшифрувати отримані дані, а потім перевірити MAC. Такий метод призводить до виникнення криптографічних оракулів, на використанні яких створено кілька ефективних атак проти TLS. Сучасний підхід: код аутентифікації додається після зашифрування відкритого тексту, тобто MAC обчислюється для вже зашифрованого повідомлення та прикріплюється до нього.

Щоб розпочати обмін інформацією по захищеному каналу, клієнт і сервер повинні узгодити *шифронабір*. Узгодження проводиться на етапі встановлення з'єднання (Handshake). Шифронабори мають важливе значення для безпеки TLS. Вибір нестійкої комбінації означає, що достатнього захисту конкретна реалізація TLS не забезпечує. Тому в TLS 1.3 число допустимих шифронаборів різко скорочено — рекомендовано лише чотири.

Щойно згадувався етап встановлення з'єднання. Клієнт і сервер повинні домовитися про шифри і методи автентифікації, що використовуються, узгодити ключі та інші параметри сеансу зв'язку. Набір узгоджених параметрів називається криптографічним *контекстом*. Узгодження відбувається при встановленні з'єднання шляхом обміну спеціальними повідомленнями — *handshake-повідомленнями*. Такий обмін TLS здійснюється поверх обміну записами. Кожне handshake-повідомлення містить спеціальний заголовок, що складається із чотирьох байтів. Перший байт означає код типу повідомлення, три наступних байта — довжину повідомлення.

У TLS 1.3 встановлення з'єднання організовано таким чином, що вже на другому кроці використовується захищений режим, тобто Handshake версії 1.3 вимагає, як мінімум, використання

протоколу Діффі-Хеллмана та симетричних шифрів. До TLS 1.2 і раніше цей момент не відноситься. Розглянемо схему обміну у TLS 1.2 і раніше.



Першим повідомленням у протоколі встановлення TLS-з'єднання є повідомлення ClientHello. Повідомлення містить такі дані:

1. версія протоколу;
2. 32 байти випадкових значень — ClientRandom;
3. ідентифікатор TLS-сесії — SessionID;
4. список шифронаборів, які підтримує клієнт — Cipher Suites;
5. список методів стиснення, які підтримуються — Compression Methods, порядок відповідає ступеню переваги, але зазвичай у цьому полі лише одне значення — null, оскільки стиснення не рекомендується використовувати. У TLS 1.3 — стиснення прямо заборонено, але, з "історичних причин це поле зберігається, з фіксованим значенням null;
6. розширення протоколу.

Кожне з полів у повідомленні ClientHello має свій формат, а полю передують дані про його довжину і, у разі розширень, додатковий заголовок.

Згідно зі специфікацією, після відправки ClientHello, клієнт чекає на відповідь сервера. У відповідь може прийти або повідомлення про помилку, як Alert, або повідомлення ServerHello. Alert — короткі повідомлення, що містять інформацію про рівень помилки та її тип. Якщо ж сервер зміг успішно обробити ClientHello, він відповідає повідомленням ServerHello. ServerHello (тут ми розглядаємо версії до 1.3) містить такі поля:

1. версію протоколу, яку будуть використовувати клієнт та сервер;

2. 32 байти випадкових значень - ServerRandom;

3. ідентифікатор сесії — SessionID, присвоєний новій сесії сервером (у разі TLS 1.3 це поле обов'язково повторює SessionID клієнта);

4. вибраний сервером шифронабір — Cipher Suite, цей шифронабір буде використовуватися надалі і клієнтом, і сервером. Сервер вибирає шифронабір із запропонованих клієнтом у ClientHello, але не обов'язково слідує пріоритету клієнта;

5. вибраний сервером метод стиснення - швидше за все, це null;

6. деякий набір розширень.

Тож межах кожної сесії TLS клієнт та сервер узгоджують параметри, перелічені вище. Ці параметри дозволяють створити контекст для роботи криптосистем, які будуть обробляти захищені TLS-записи на стороні сервера і клієнта. Крім ClientHello та ServerHello під час встановлення з'єднання вузли обмінюються кількома іншими повідомленнями.

Ключовим аспектом для сучасних реалізацій TLS є використання *сертифікатів* TLS. Сертифікати надсилаються сервером у повідомленні Certificate. Це повідомлення є практично завжди. Серверний сертифікат містить відкритий ключ сервера. Теоретично, специфікація дозволяє встановити з'єднання без відправки серверних сертифікатів. Це так званий "анонімний"режим, але він фактично не вживається, як і режими TLS без шифрів. У типовій конфігурації повідомлення Certificate включатиме кілька TLS-сертифікатів, серед яких один серверний сертифікат і так звані "проміжні сертифікати що дозволяють клієнту побудувати ланцюжок валідації. Серверний сертифікат — це сертифікат, який відповідає серверу і містить серверний відкритий ключ електронного підпису (у більшості випадків - RSA або ECDSA). Специфікація наказує строгий порядок проходження сертифікатів у повідомленні: першим повинен йти серверний сертифікат, а наступні — у порядку посвідчення попереднього. У TLS 1.3 цю вимогу пом'якшили, зробивши допустимим довільний порядок проміжних сертифікатів (але серверний все одно очікується першим).

ServerKeyExchange — повідомлення, яке містить серверну частину даних, необхідних для обчислення загального сеансового ключа. Повідомлення може бути відсутнім, а в TLS 1.3 воно не використовується, оскільки нова схема Handshake для обміну криптографічними параметрами заснована на розширеннях Hello-повідомлень. У версії TLS 1.2 і раніше ServerKeyExchange зазвичай містить параметри протоколу Діффі-Хеллмана (DH), проте історичний варіант передбачає передачу тимчасового ключа RSA.

У TLS можлива взаємна (двостороння) автентифікація вузлів, що використовує сертифікати TLS. Зазвичай клієнтські сертифікати використовуються при доступі до банківських, платіжних систем, корпоративних веб-шлюзів різного призначення, а також до державних інформаційних систем через веб-інтерфейс. Сервер може запросити сертифікат клієнта за допомогою повідомлення *CertificateRequest*.

Повідомлення *ServerHelloDone* у TLS 1.2 і раніше сигналізує про закінчення набору повідомлень, очолюваного ServerHello. Повідомлення має нульову довжину і служить простим прапором,

що означає, що сервер передав свою частину початкових даних і тепер чекає на відповідь від клієнта.

Отже, сервер відповідає на ClientHello послідовністю повідомлень TLS Handshake, з максимум п'ятьма повідомленнями. Клієнт повинен відповісти своїм набором повідомлень, описаних нижче.

Certificate — це повідомлення містить сертифікат клієнта, якщо він був запрошений сервером. Якщо клієнт сертифіката не має, а сервер його запитує, то клієнт або пропускає це повідомлення (SSLv3), або відповідає порожнім повідомленням з типом Certificate (TLS).

ClientKeyExchange — клієнтська частина обміну даними, які використовуються при отриманні сеансових ключів. Зміст цього повідомлення залежить від обраного шифронабору. У версіях TLS до 1.3 є два основних типи — RSA та кілька варіантів протоколу Діффі-Хеллмана. Обмін сеансовим ключем за допомогою RSA продовжує використовуватися в деяких реалізаціях TLS (крім 1.3), проте сучасний метод використання протоколу Діффі-Хеллмана.

CertificateVerify — якщо клієнт передав у відповідь на запит сервера свій сертифікат, то серверу потрібен деякий механізм, що дозволяє перевірити, що клієнт дійсно має секретний ключ, пов'язаний із сертифікатом. Для цього клієнт підписує масив переданих та прийнятих раніше повідомлень Handshake. Повідомлення буде надіслано лише у випадку, якщо надіслано клієнтський сертифікат. У TLS 1.3 аналог цього повідомлення перенесений на бік сервера і є механізмом ранньої автентифікації криптографічних параметрів.

Клієнтські сертифікати використовують рідко. Тому в типовому випадку клієнт на даному етапі передає одне повідомлення - ClientKeyExchange. Це повідомлення є, відповідно до специфікації, обов'язковим.

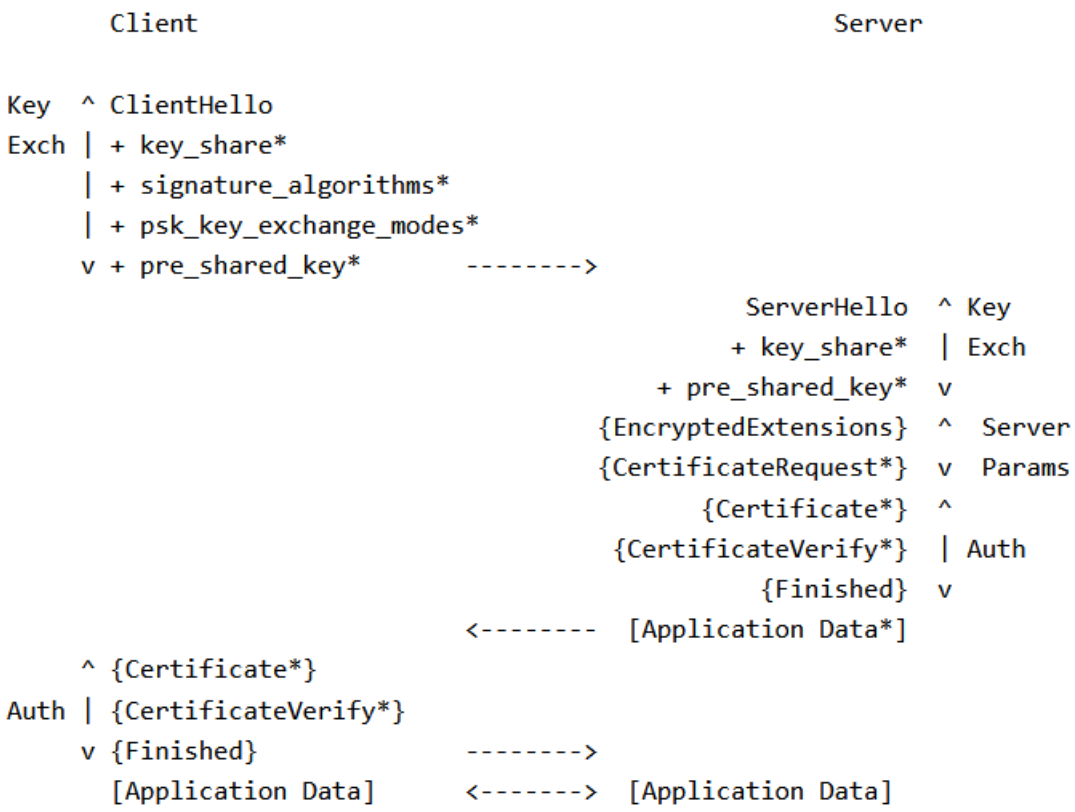
Слідом за повідомленням ClientKeyExchange, якщо воно було єдиним, або CertificateVerify, клієнт повинен передати повідомлення ChangeCipherSpec. Важливий момент: це повідомлення не є повідомлення Handshake. Так, у специфікації TLS зустрічаються такі, не зовсім прозорі моменти, коли начебто логічний перебіг протоколу переривається спеціальними "вставками". При розробці TLS 1.3 від цієї сумнівної практики відмовилися, але ChangeCipherSpec все одно рекомендується передавати, правда, як фіктивне повідомлення, яке просто ігнорується вузлами (повідомлення потрібно для того, щоб "обдурити"неправильно налаштовані проміжні вузли, замаскувавши з'єднання 3.). ChangeCipherSpec - це спеціальне повідомлення-сигнал, що означає, що з цього моменту клієнт переходить на вибраний шифр, а TLS-записи будуть зашифровані.

З боку клієнта встановлення з'єднання завершується надсиланням повідомлення Finished. Finished є першим захищеним повідомленням у рамках нового сеансу TLS. Оскільки повідомлення ClientHello і ServerHello не містять підписів, Finished не захищає сесію від атак, заснованих на заміні параметрів до відправки ChangeCipherSpec. Цей дефект протоколу використаний в гучній атаці Logjam в 2015 році (про атаки на TLS/SSL ми детально поговоримо нижче). У TLS 1.3 автентифікація параметрів сесії починається раніше: саме - сервер негайно засвідчує першу частину повідомлень Handshake, включаючи вибір криптосистем, з допомогою серверного повідомлення CertificateVerify.

Після того, як клієнт і сервер обмінялися парами ChangeCipherSpec і Finished - захищене з'єднання успішно встановлено і дані можуть передаватися в захищеній формі.

У TLS 1.3 від попередніх версій успадковано лише базові принципи встановлення з'єднання: збереглися ролі вузлів (з'єднання ініціює клієнт), не змінилася послідовність ClientHello – ServerHello, є повідомлення-сигнал Finished. Однак на цьому схожість закінчується. У TLS 1.3 скорочено і загальну кількість повідомлень Handshake, і кількість повідомлень, що передаються у відкритому вигляді: вузли практично відразу переходять на зашифрований обмін. Тому зі схеми встановлення з'єднання видалено сигнал ChangeCipherSpec — він більше не потрібен. Змінилася роль повідомлення CertificateVerify — під час передачі з боку сервера воно засвідчує повідомлення першої частини Handshake.

Схема повного Handshake TLS 1.3 виглядає так:



Тут символом "*"відзначені необов'язкові повідомлення (передаються залежно від контексту), а у фігурних дужках — повідомлення Handshake, які передаються у зашифрованому вигляді. Квадратні дужки позначають дані корисного навантаження, що передаються у зашифрованому вигляді, але з використанням іншого набору ключів шифрування. Зауважимо, що вже перша відповідь сервера містить зашифровані дані. У TLS попередніх версій практично всі суттєві Handshake-повідомлення передаються у відкритому вигляді.

У протоколі встановлення з'єднання TLS 1.3 виділяються три фази: вироблення початкового значення загального криптографічного секрету (ключів), визначення параметрів з'єднання, автентифікація (сервера та клієнта).

До першої фази відносяться ClientHello та ServerHello (а також їх розширення, позначені на схемі символом "+"). Результатом обміну повідомленнями у першій фазі є отримання першого секрету. На основі цього секрету, використовуючи додатково самі повідомлення, передані на цей

момент, сторони отримують перший набір симетричних ключів. Ці ключі призначені для зашифрування повідомлень Handshake. Перша фаза завершується передачею ServerHello та розширень.

Друга фаза включає серверні повідомлення EncryptedExtensions і, при необхідності автентифікації клієнта, CertificateRequest. Нове повідомлення — EncryptedExtensions — додано до TLS 1.3.

У третій фазі, представленій групами повідомлень Certificate, CertificateVerify та Finished, відбувається автентифікація сервера та клієнта. Як і в попередніх версіях TLS, автентифікація може бути повністю виключена (анонімний режим), проте типовий сценарій використання передбачає принаймні автентифікацію сервера клієнтом. При цьому TLS 1.3 сервер може перейти до відправки даних корисного навантаження (Application Data на схемі) безпосередньо після серверного Finished — це можливо тому, що до цього моменту сервером вже отримано перший набір симетричних ключів.

Навіть повний формат Handshake в 1.3 виявляється коротшим на одну ітерацію: якщо поглянути на попередню версію, то там, після отримання відповіді сервера (закінчується ServerHelloDone), клієнт повинен був відправити свою порцію повідомлень, що завершується Finished, і дочекатися відповідного серверного Finished, тільки після цього переходити до передачі корисного навантаження. У новій версії серверне повідомлення Finished приходить у першій відповіді сервера, відповідно, клієнт може відразу переходити до відправки корисного навантаження (після свого Finished, звичайно), не чекаючи ще одного пакета від сервера. Це заощаджує час, необхідний доставки пакетів від клієнта до серверу і назад.

TLS/SSL сертифікати

TLS-сертифікати (SSL-сертифікати — застаріла назва, що є синонімом) відіграють важливу роль у сучасній інфраструктурі TLS. TLS-сертифікат має одне призначення: прив'язати відкритий ключ до деякого мережного імені. Сертифікати TLS не містять жодної секретної інформації. Вони не є безпосередніми носіями ключів шифрування даних, що передаються в TLS (за винятком відкритого серверного ключа RSA, який може бути використаний для зашифрування сеансового секрету в застарілих режимах роботи TLS).

Клієнт погоджується вірити деякій третій стороні (не зловмиснику, а центру сертифікації — ЦС), що ця третя сторона перевірила відповідність ключа і мережевого імені (у вебі це зазвичай домен). Результат такої перевірки підтверджується підписом ЦС, який ЦС ставить на сертифікаті сервера. Передбачається, що клієнт TLS має у своєму розпорядженні деякий набір сертифікатів центрів (частина з цих сертифікатів є кореневими) і може перевірити підписи на сертифікатах, що надаються сервером, вибудувавши ланцюжок довіри, що веде від серверного ключа до одного з довірених коренів.

У SSL/TLS використовуються сертифікати формату X.509, це дуже старий формат, який спочатку розроблявся для телеграфу, але пізніше був адаптований для використання в Інтернеті.

Сертифікат є електронним документом, що має певну структуру. Так кожен сертифікат містить поля Issuer та Subject. Поле Issuer визначає ім'я сторони, що випустила цей сертифікат (що поставила підпис на ньому), а поле Subject — ім'я сторони, для якої сертифікат випущений.

У випадку Інтернету, в Issuer зазвичай вказано ім'я з сертифіката ЦС (не обов'язково кореневого), а в Subject для серверного сертифіката — доменне ім'я, що є адресою сайту. Клієнт отримує сертифікати в повідомленні Certificate, вибудовує їх в ієрархію, де кожен сертифікат засвідчує наступний, перевіряє підписи та відповідність імені серверного сертифіката імені сервера, з яким планує встановити з'єднання.

Сертифікат випускається на певний термін, який зазначено у самому сертифікаті. Сертифікат може бути відкликаний з якоїсь причини раніше закінчення терміну дії. Браузери та інші TLS-клієнти повинні перевіряти статус відкликання сертифікатів, для цього існує ряд механізмів, однак у повсякденній реальності перевірка відкликання сертифікатів фактично не працює. Цей момент потрібно враховувати при аналізі ризиків.

Інфраструктура відкритих ключів

Інфраструктура відкритих ключів (PKI - Public Key Infrastructure) - це технологія автентифікації, що використовує для ідентифікації суб'єктів криптографію з відкритими ключами разом з наступними механізмами:

1. механізм встановлення довіри на базі певної моделі довіри;
2. механізмом присвоєння суб'єктам імен, унікальних в даному середовищі;
3. Механізмом поширення інформації, що характеризує правильність зв'язування певної пари ключів (відкритого та закритого) з певним ім'ям суб'єкта в даному середовищі (така інформація фіксується і надається центром, якому довіряє верифікатор інформації).

PKI на основі асиметричної криптографії використовують для захисту електронного зв'язку для електронної пошти, інтернет-банкінгу, покупок в Інтернеті, а також для спілкування мільйонів користувачів і веб-сайту, до якого вони підключаються за допомогою HTTPS.

Ключові компоненти PKI:

– Політика сертифікатів – це вимоги безпеки, що використовуються для визначення ієрархії і структури середовища PKI. Крім того, політики обмежені обробкою і управлінням ключами, відкликанням, профілями і форматами сертифіката, безпечним сховищем з багатьма іншими деталями.

– Кореневий центр сертифікації (засвідчувальний центр) – це об'єкт, який є «основним коренем довіри» в реалізації PKI і відповідає за автентифікацію особистості в середовищі PKI.

– Підлеглий ЦС: сертифікується через кореневий ЦС для його використання відповідно до визначення, передбаченим політикою сертифікатів. Крім того, цифрові сертифікати підписуються і видаються суб-CA.

– База даних сертифікатів.

– Послуги відкликання – сервери, які публікують оновлені CRL (списки відкликаних сертифікатів) або OCSP (онлайн-протокол статусу сертифіката), які використовують CRLS для відповіді на перевірки пошуку відкликання для всіх пристроїв, які не можуть самостійно обробляти CRL.

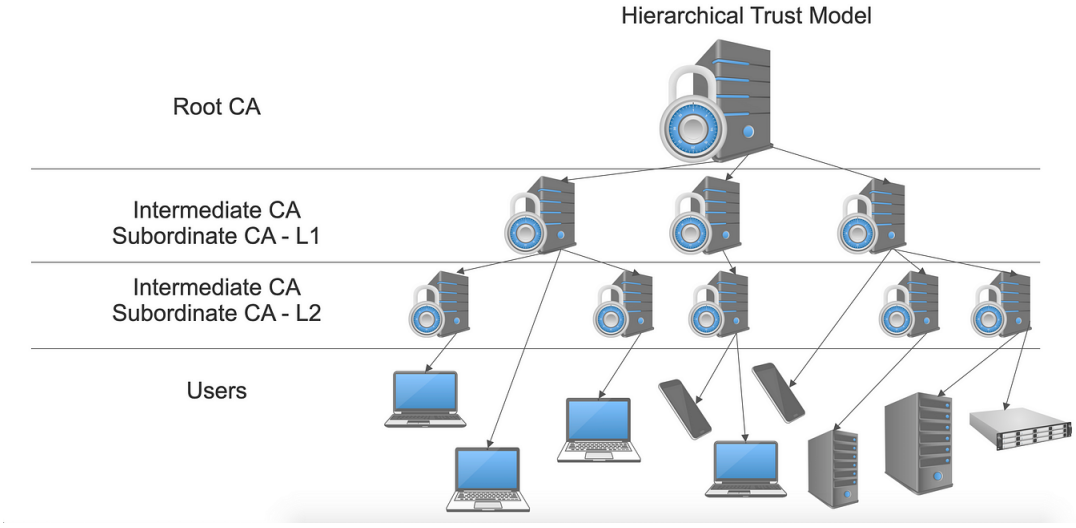
– Користувачі (кінцеві суб'єкти).

У складі PKI повинні функціонувати підсистеми скасування сертифікатів, створення, скасування та відновлення ключів, автоматичного корегування пар ключів, супроводження життєвого циклу ключів, підтримки т.зв. взаємної сертифікації, тощо. Прикладне програмне забезпечення користувачів має взаємодіяти з усіма підсистемами у безпечний, узгоджений та надійний спосіб.

Для різних користувачів довірчою стороною можуть виступати різні центри сертифікації, або навіть користувачі з відповідними повноваженнями. Таким чином, PKI має гарантувати перевіряючому, що він може довіряти іншим центрам сертифікації чи уповноваженим користувачам.

Для реалізації цієї вимоги прийнято концепцію шляхів довіри. Шлях довіри — це ланцюжок сертифікатів, що дозволяє простежити статус довіри між центрами сертифікації: своїм та відправника. Якщо центри сертифікації довіряють одне одному, то перевіряючій може довіряти відповідному користувачу. У сертифікатах передбачено наявність інформації, що дозволяє пересуватися ланцюжком сертифікатів та перевіряти дійсність сертифікату на відповідному кроці.

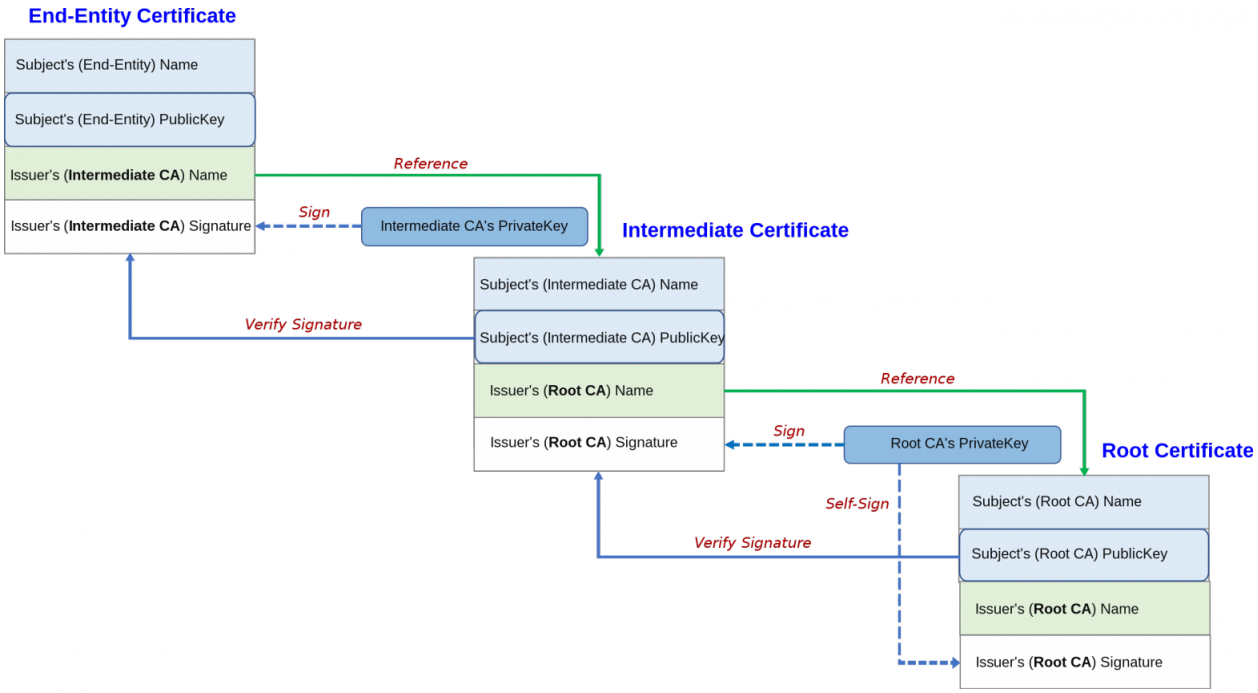
Основна модель архітектури PKI – ієрархічна модель: в такій системі повинен існувати корінь (власне, кореневий ЦС). Він здійснює реєстрацію підлеглих ЦС і формує для них кореневі сертифікати, підлеглі ж ЦС можуть випускати власні сертифікати кінцевим користувачам (або ЦС ще нижчого рівня).



В класичній ієрархічній схемі кореневими сертифікатами являються ті сертифікати, які, очевидно, випускаються кореневим ЦС. В нашому ж світі існують різні ЦС, кожен з яких може одночасно виступати коренем деякого дерева, тому правильним буде формулювання, що кореневий сертифікат – це самопідписаний сертифікат деякого ЦС. Відповідно, якщо деякий кореневий ЦС буде скомпрометовано, то всі сертифікати на всіх рівнях з коренем у вигляді цього ЦС будуть також визнані скомпрометованими.

Організація роботи з кореневими сертифікатами

Отже, наразі більшість сайтів отримують від деякого ЦС сертифікат, який можуть надсилати користувачу та встановлюти завдяки ньому безпечне з'єднання, але виникає питання: яким чином користувач може впевнитись, що цьому сайту з цим сертифікатом можна вірити? Для цього було створено так звані кореневі сховища сертифікатів (root store) – це сховища довірених кореневих центрів сертифікації.



Майже кожна операційна система містить у собі таке сховище – Windows, MacOS, Android, IOS, і, зазвичай, браузери звертаються до системного сховища (Safari, Chrome та усі Chromium-браузери – Opera, Edge тощо). Виключенням є Firefox – компанія Mozilla володіє власним сховищем кореневих сертифікатів, які використовуються її браузером, а також ОС Linux.

Очевидно, що сховище кореневих сертифікатів потрібно завжди тримати в актуальному стані, а оновити їх можна оновленням ОС та/або браузера, тому нехтувати безпековими оновленнями не варто. Звісно, ніхто не забороняє самостійно редагувати (добавляти/видаляти) якісь кореневі сертифікати, але за усі наслідки, в такому випадку, відповідальність нестиме лише користувач.

Таким чином, певний вебсайт отримує сертифікат від деякого ЦС (тобто публічний та приватний ключ), публічний ключ буде надісланий у складі TLS-сертифікату, а приватний буде зберігатися в оперативній пам'яті, щоб його не можна було дістати.

Утворення спільного ключа

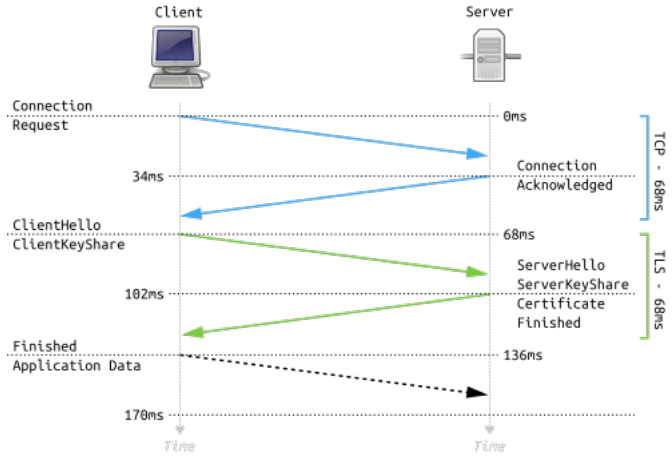
Вище вже було детальну схему роботи протоколу, але опишемо ще раз процедуру рукостискання (для версії 1.3):

1. ClientHello: клієнт надсилає повідомлення "Hello яке містить версію протоколу, деяке псевдовипадкове число (premaster secret) і список шифрів, які він підтримує. Фактично, клієнт

допускає, що сервер підтримує нову версію TLS, і хоча б один шифр (список яких був і так сильно скорочений) сервер може забезпечити.

2. ServerHello: сервер отримав псевдовипадкове значення від клієнта, генерує своє, і тепер може сформувати спільний ключ на своїй стороні. Тепер він надсилає сертифікат сервера, цифровий підпис сертифікату, згенероване псевдовипадкове значення та обраний набір шифрів.

3. Finished: клієнт перервяє підпис і сертифікат, також генерує спільний ключ за допомогою надісланого псевдовипадкового premaster secret-a сервера і надсилає повідомлення "Finished".



Метод ARP спуфінгу та його застосування для атаки sslstrip

Атака спуфінгу протоколу вирішення адрес (ARP) або отруєння ARP - це форма атаки спуфінгу, яку хакери використовують для перехоплення даних. Хакер вчиняє атаку спуфінгу ARP, обманюючи один пристрій, щоб він відправляв повідомлення хакерові замість заданого отримувача. Таким чином, хакер отримує доступ до комунікацій вашого пристрою, включаючи чутливі дані, такі як паролі та інформація про кредитну картку. На щастя, ви можете захистити себе від таких атак за допомогою кількох способів.

Протокол ARP і спуфінг ARP

Атаки спуфінгу ARP відбуваються в локальній мережі (LAN) за допомогою протоколу ARP. ARP - це протокол обміну інформацією, який з'єднує динамічну адресу інтернет-протоколу (IP) з фізичною адресою пристрою. Останню іноді називають адресою керування доступом до мережі (MAC). Протокол ARP керує комунікацією в мережі LAN.

Кожен мережевий пристрій має як IP-адресу, так і MAC-адресу. Щоб відправляти та отримувати повідомлення, хости в мережі повинні знати адреси інших пристроїв в цій мережі. Для цього хост пов'язує динамічну IP-адресу з фізичною MAC-адресою.

Наприклад, хост A в комп'ютерній мережі бажає пов'язати свою IP-адресу з MAC-адресою хоста B. Для цього він надсилає запит ARP всім іншим хостам в LAN. Після цього запиту він отримує відповідь ARP від хоста B із його MAC-адресою. По завершенні запитування хост зберігає цю адресу в своєму кеші ARP, схожому на список контактів. Цей кеш іноді називається таблицею ARP, оскільки адреси зберігаються у вигляді таблиці.

Спуфінг ARP означає, що атакувач з доступом до LAN претендує на роль хоста B. Атакувач відправляє повідомлення хосту A з метою обману хоста A і збереження адреси атакувача як адреси хоста B. В результаті хост A відправлятиме комунікації, призначені для хоста B, атакувачеві замість цього. Як тільки атакувач стає проміжним ланцюжком, кожного разу, коли хост A спілкується з хостом B, насправді цей хост спілкується спочатку з атакувачем. Хост B, як правило, є маршрутизатором за замовчуванням.

Мета атаки спуфінгу ARP

Атаки спуфінгу ARP можуть мати кілька цілей. Атаки можуть бути використані для шпигунства, атак середньої людини або додаткових кібератак, таких як атаки відмови в обслуговуванні (DoS).

Що таке атаки шпигунства та атаки DoS?

Шпигунство відбувається, коли атакувач не модифікує комунікацію між хостами A та B, але лише читає комунікацію та пересилає її заданому отримувачеві. У випадку атаки середньої людини атакувач модифікує інформацію перед її надсиланням заданому отримувачеві.

Проте такі дії шпигунства та модифікації повідомлень часто є частиною більш складної кібератаки, яка включає в себе рух в бічному напрямку. Однією з таких атак є атака відмови в обслуговуванні, в якій атакувач заважає надсиланню комунікації між двома або більше хостами. Атакувачі також можуть надсилати зловмисні файли між хостами.

Хто використовує спуфінг ARP?

Хакери використовують спуфінг ARP з 1980-х років. Атаки хакерів можуть бути плановими або спонтанними. Планові атаки включають атаки відмови в обслуговуванні, в той час як крадіжка інформації з громадської мережі WI-FI - це приклад спонтанних атак. Хоча такі атаки можуть бути запобіжними, їх все ще часто використовують, оскільки вони легко виконуватися як з фінансової, так і з технічної точки зору.

Однак спуфінг ARP також може бути виконаний із доброї метою. Розробники також використовують спуфінг ARP для налагодження мережевого трафіку, наміреного вставити проміжного ланцюжка між двома хостами. Етичні хакери також можуть симулювати атаки на отруєння кешу ARP, щоб переконатися, що мережі захищені від таких атак.

Ефекти спуфінгу ARP

Спуфінг ARP має схожі ефекти з іншими формами спуфінгу, такими як спуфінг електронної пошти. Явні ефекти спуфінгу ARP можуть варіюватися від відсутності до повного втрати зв'язку в мережі, якщо атакувач використовує атаку відмови в обслуговуванні. Чи видимі ефекти атаки залежать від цілей хакера. Якщо їхня мета - лише шпигунство або модифікація інформації між хостами, це може залишитися непоміченим. Якщо вони намагаються провести подальшу атаку, як це часто буває при атаках спуфінгу ARP, їм також потрібно залишитися непоміченими. Проте такі атаки стануть очевидними, коли буде досягнуто кінцеву мету. Наприклад, вашу систему може перенапружити комунікація від зловмисних програм або ваш комп'ютер може бути інфікований вимагальним програмним забезпеченням.

Чому спуфінг ARP є небезпечним

Спуфінг ARP може бути небезпечним з багатьох причин. Найважливіше, він надає хакерам несанкціонований доступ до особистої інформації. Хакер може використовувати цей доступ для численних зловмисних цілей, таких як отримання паролів, ідентифікаційної інформації або інформації про кредитну картку. Ці атаки також можуть використовуватися для введення зловмисного програмного забезпечення, такого як вимагальники викупу.

Як визначити, що хтось використовує спуфінг ARP

Щоб визначити, чи вас обманюють, перевірте свою програму автоматизації завдань та управління конфігурацією. Тут ви зможете знайти ваш кеш ARP. Якщо є дві IP-адреси з однаковою MAC-адресою, ви можли могли стати жертвою атаки. Хакери, як правило, використовують програмне забезпечення спуфінгу, яке відправляє повідомлення, стверджуючи, що це адреса за замовчуванням шлюзу.

Однак також можливо, що це програмне забезпечення обманює свою жертву і перезаписує MAC-адресу шлюза за замовчуванням своєю. У цьому випадку вам слід ретельно переглянути трафік ARP на наявність незвичайних даних. Незвичайний трафік, як правило, представлений незапитаними повідомленнями, які стверджують, що вони володіють або IP-адресою, або MAC-адресою маршрутизатора. Такі незапитані повідомлення можуть свідчити про атаку спуфінгу ARP.

Як атакуючі намагаються залишити себе непоміченими

Атакуючі, які використовують спуфінг ARP, часто бажають залишити себе непоміченими. Таким чином, вони можуть збирати інформацію або подальше проникнення в мережу з метою запуску більш значущої атаки. Якщо хости в мережі розуміли, що отримані ними запити ARP є хибними, то спуфінг був би знижений.

Зазвичай потерпілі не виявляють зіткненого кешу ARP; вони продовжують отримувати комунікації, як зазвичай. Проте атакувач перехоплює ці комунікації, які відправляються за допомогою протоколу ARP, такі як імена користувачів і паролі потерпілих.

Як захистити свої системи від спуфінгу ARP

На щастя, ви можете виконати ряд процедур та інструментів як частину вашого плану реагування на інциденти для запобігання спуфінгу ARP. Ці методи включають сертифікацію запитів ARP, фільтрацію пакетів, анти-ARP-спуфінгове програмне забезпечення та шифрування.

Інструменти для перевірки запитів ARP

Статичні записи ARP представляють найпростіший спосіб перевірки IP- та MAC-адрес. Статичний запис ARP вводиться або приймається вручну, виключаючи можливість автоматичного зміщення кешу ARP відповідно до протоколу ARP. Це можливо для деяких записів (наприклад, адрес за замовчуванням шлюзу), тоді як інші залишаються динамічними. Ймовірно, більшість адрес повинні залишитися динамічними, оскільки обслуговування кешу на багатьох мережах було б надто вимогливим.

Існує також програмне забезпечення для перевірки запитів ARP, яке допомагає захистити вас від спуфінгу ARP. Це програмне забезпечення сертифікує IP- та MAC-адреси та активно блокує несертифіковані відповіді. Існує інше програмне забезпечення, яке повідомляє хоста, коли запис в таблиці ARP було змінено. Існує безліч варіантів програмного забезпечення, деякі з них працюють

на рівні ядра, тоді як інші можуть бути частиною протоколу динамічної конфігурації хоста або мережевого комутатора.

Інструменти для запобігання спуфінгу ARP

Досить простий спосіб захисту від спуфінгу ARP - це використовувати пакетно-фільтруючий брандмауер. Пакетно-фільтруючі брандмауери позначають пакети даних з адреси, яка знаходиться двічі в мережі, оскільки таке подвоєння свідчить про присутність когось, хто прикидається іншим хостом.

Проте шифрування, ймовірно, є найважливішим способом захисту від атаки ARP. Завдяки широкому використанню шифрованих протоколів комунікації сьогодні спуфінг ARP став набагато складнішим. Наприклад, більшість трафіку на веб-сайтах зараз шифрується за допомогою HTTPS, що заважає хакерам читати повідомлення після їх перехоплення. Таким чином, найважливішим способом запобігти таким атакам є уникнення надсилання нешифрованих даних.

Також програмне забезпечення може допомогти забезпечити захист. Багато програм для захисту кібербезпеки повідомляють користувачів, якщо вони знаходяться на сайті, де дані не шифруються.

Іншим інструментом шифрування, який варто відзначити, є віртуальна приватна мережа (VPN). Підключившись до VPN, весь ваш трафік пройде через зашифрований тунель.

Зрештою, хоча це не є "інструментом"в прямому розумінні слова, ви можете симулювати спуфінг ARP у своїй власній мережі, щоб знайти потенційні слабкі місця в системі кібербезпеки. Фактично більшість інструментів, що використовуються для ініціації цих атак, широко доступні і легко використовувані, що робить етичний хакінг виправданим підходом до захисту від таких атак.

Що таке SSL стріппінг?

SSL стріппінг - це загроза кібербезпеці, яка призводить до переходу з безпечного з'єднання HTTPS до менш безпечного зашифрованого з'єднання HTTP, що призводить до того, що весь веб-зв'язок більше не зашифрований.

Ідеально, Gmail хоче, щоб користувачі використовували HTTPS, тому він надсилає сторінку входу, зашифровану за допомогою SSL, але через те, що хакер є посередником, він може "позбавити"(тобто видалити) SSL перед пересиланням його користувачеві. Не маючи жодного підозри, користувач вводить свій пароль і натискає "Увійти не усвідомлюючи, що надсилає його чітким текстом безпосередньо хакерові. Після цього хакер додає назад шифрування SSL перед пересиланням на Gmail, успішно виконуючи свою місію зі зловмисними цілями вкрасти конфіденційну інформацію.

Історія атак спуфінгу SSL

Атаки спуфінгу SSL досить відомі серед професіоналів у галузі кібербезпеки. Вперше вони були представлені на конференції Black Hat 2009 року в Вашингтоні, округ Колумбія, Моксі Марлінспайком, краще відомим як геній з кібербезпеки зашифрованого чат-додатку Signal. Дивно, але атака все ще працює, незважаючи на те, що вона старша за 8 років!

Однак змінилося те, що деякі сайти впровадили новий протокол під назвою HSTS (HTTP Strict Transport Security), створений для запобігання спуфінгу SSL. Сайти, які використовують HSTS, дозволяють браузеру робити запити тільки в HTTPS, а не звичайному HTTP, як той, який спочатку перехоплюється хакером від користувача.

На щастя, спуфінг SSL більше не працює з Facebook або Gmail, оскільки вони повністю перейшли на HTTPS і впровадили HSTS. Проте є ще багато популярних сайтів, таких як Hotmail, Amazon, eBay і Citibank, які ще не покинули HTTP і, отже, не відповідають критеріям для впровадження HSTS.

Як виявити спуфінг SSL

Багато атак спуфінгу SSL залишаються непоміченими. Однак є способи їх виявлення. І найкраща частина полягає в тому, що це не так важко зробити. Ви можете виявити спуфінг SSL за допомогою таких загальних ознак:

Ви були на сайті з HTTPS, перш ніж він перейшов на HTTP. Замок поруч з адресною стрічкою веб-переглядача відкритий і забарвлений червоним. Дизайн сайту виглядає дивно. Багато орфографічних помилок і граматичних помилок на сайті.

Як працює атака спуфінгу SSL

Зазвичай інтернет-користувач спершу підключається до версії HTTP сайту, перш ніж його переадресують до версії HTTPS. У атаках спуфінгу SSL хакер перехоплює з'єднання, перш ніж користувач встигне підключитися до версії HTTPS.

Далі ми розглянемо деякі різновиди атак спуфінгу SSL.

Проксі-сервери

У атаках спуфінгу SSL за допомогою проксі-сервера хакер сидить між потерпілим і сервером. Для перехоплення і крадіжки даних хакер повинен спершу отримати доступ до мережі, до якої підключено пристрій. Потім він може вибрати налаштування проксі-сервера, щоб вони відповідали його потребам, або використовувати комбінацію інших типів атак спуфінгу SSL для отримання доступу до даних, які йому потрібні.

Спуфінг протоколу вирішення адрес (ARP)

Протокол вирішення адреси (ARP) - це налаштування, яке дозволяє мережі підключатися за допомогою IP-адреси та визначати MAC-адреси. MAC-адреса - це глобально унікальний номер, який призначений для кожної мережевої карти і слугує фактичною адресою та ідентифікатором пристрою.

Коли пристрої в мережі повинні спілкуватися між собою, їм потрібно знати MAC-адреси інших пристроїв. Щоб отримати MAC-адресу, пристрої виконують ARP для отримання MAC-адреси.

Під час атаки спуфінгу ARP, або атаки отруєння ARP, хакер надсилає вигадане повідомлення ARP на пристрій потерпілого, щоб отримати його MAC-адресу. Після цього хакер може перевірити всі дані, які передаються потерпілим, перш ніж відправити їх ініціально призначеному пристрою.

Рекомендації для підвищення безпеки використання протоколу TLS

Для уникнення відомих вразливостей і підвищення безпеки використання протоколу TLS (Transport Layer Security), слід дотримуватися наступних рекомендацій:

1. **Оновлюйте версію TLS**: завжди використовуйте найновішу версію TLS, оскільки нові версії зазвичай мають виправлення для відомих вразливостей.

2. **Вимагайте використання сильних шифрів**: налаштуйте сервер і клієнт так, щоб вони вимагали використання сильних криптографічних шифрів і відкидали застарілі або слабкі шифри.

3. **Встановлюйте правильні параметри конфігурації TLS**: налаштуйте параметри TLS, такі як вирішення обміну ключами, кількість переговорів і час життя сесії, для забезпечення оптимального рівня безпеки.

4. **Використовуйте сертифікати від надійних постачальників**: отримуйте SSL/TLS-сертифікати від надійних постачальників і регулярно оновлюйте їх.

5. **Ввести захист від атаки "downgrade attack"(атака зниження)**: захистіть вашу систему від атаки, яка спрямована на "зниження"рівня TLS до менш захищеного протоколу, наприклад, до SSL.

6. **Запускайте регулярні аудиторські перевірки**: проводьте аудиторські перевірки та перевірки безпеки TLS для виявлення потенційних вразливостей.

7. **Використовуйте багаторівневий захист**: розгляньте можливість використання багаторівневого захисту, такого як брандмауери, системи вторгнення та антивірусне програмне забезпечення, для захисту від різних атак.

8. **Встановлюйте систему моніторингу безпеки**: використовуйте систему моніторингу безпеки для виявлення підозрілих або несправедливих активностей в мережі і на серверах.

9. **Забезпечте резервне копіювання і відновлення**: регулярно створюйте резервні копії конфігурації та ключів TLS і забезпечте можливість їх відновлення в разі втрати або пошкодження.

10. **Оновлюйте ваше програмне забезпечення**: використовуйте оновлення для вашого вебсервера та програмного забезпечення TLS для виправлення вразливостей та забезпечення безпеки.

11. **Дотримуйтесь найкращих практик**: досліджуйте і впроваджуйте найкращі практики в галузі кібербезпеки для забезпечення максимальної безпеки TLS.