НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикуму

ДОСЛІДЖЕННЯ РЕАЛІЗАЦІЙ ПРОТОКОЛУ SSL/TLS

Виконали студенти
групи ФІ-32мн
Мельник Ілля,
Міснік Аліна

Перевірила:
Селюх П.В.

Київ — 2024

Мета роботи: Дослідження особливостей реалізації криптографічних механізмів протоколу SSL/TLS

1 ХІД РОБОТИ

1.1 SSL протокол

SSL або secure socket layer — це протокол, що шифрує і захищає дані під час їх передачі в мережі, використовуючи спеціальні криптографічні ключі.

В моделі TCP/IP протокол знаходиться на прикладному рівні, а в OSI — між транспортним і прикладним. Сьогодні SSL вже використовується рідко внаслідок заміни більш безпечнішим і новішим TLS протоколом.

В загальному роботу протоколу можна описати наступним чином: коли ви заходите на сайт, що захищений SSL-протоколом, браузер встановлює безпечне з'єднання із сервером сайту. Таке з'єднання необхідно, щоб можна було без страху передавати конфіденційну інформацію такі, як паролі чи чані банківських карток. Як тільки з'єднання буде налаштовано, браузер використовуючи криптографічний ключ зашифрує повідомлення і відправить серверу. Сервер в свою чергу отримає це повідомлення і розшифрує з допомогою власного ключа. Ця сесія відбуватиметься доки буде відкрита вкладка.

Можемо поділити роботу SSL на дві задачі: перевірку дійсності сайту та встановити з ним захищений зв'язок. Зазвичай сайт використовує протокол HTTP для передачі даних між клієнтом і сервером. Але HTTP є незахищеним протоколом, через що він стає вразливим. Тому з'явився новий стандарт HTTPS — протокол, що дуже схожий на свого попередника, але містить шифрування, яке якраз виконує SSL.

Таким чином SSL дозволяє:

– забезпечити цілісність даних (зміна даних);

– захиститися від атаки MitM;

– перевіряти сайти.

Також варто зазначити, що по суті SSL містить у собі дві сутності: SSL протокол та SSL сертифікат.

SSL протокол — визначає порядок дій, що необхідні для встановлення захищеного з'єднання між клієнтами й серверами. **SSL сертифікат** — це елетронний документ, що містить інформацію про сайт, яка необхідна для встановлення безпечного з'єднання, а саме туди входить:

– дані про власника сертифіката;

– публічний ключ для шифрування;

– дійсний підпис.

Формує і видає такі сертифікати Центр сертифікації. В кожній країні центром сертифікації можуть виступати різні органи.

1.1.1 Алгоритми шифрування

Для симетричного шифрування використовували різні алгоритми. Першим був блоковий шифр DES, розроблений компанією IBM. У США його затвердили як стандарт у 70-х роках. В основі алгоритму лежить мережа Фейстеля з 16-ма циклами. Довжина ключа становить 56 біт, а блоку даних — 64. Розвитком DES є алгоритм 3DES. Він створювався з метою вдосконалення короткого ключа. Розмір ключа і кількість циклів шифрування збільшилася втричі, що знизило швидкість роботи, але підвищило надійність. Ще був блоковий шифр RC2 зі змінною довжиною ключа, який працював швидше за DES, а його 128-бітний ключ можна було порівняти з 3DES за надійністю. Потоковий шифр RC4 був набагато швидшим за блокові й будувався на основі генератора псевдовипадкових бітів. Але сьогодні всі ці алгоритми вважаються небезпечними або застарілими.

Наразі прийнятим стандартом є алгоритм AES.

Що стосується асиметричного шифрування, то воно найчастіше будується на базі таких алгоритмів, як RSA, DSA або ECC. RSA використовується і для шифрування, і для цифрового підпису. Алгоритм заснований на складності факторизації великих чисел і підтримує всі типи SSL-сертифікатів.

DSA (Digital Signature Algorithm) використовується тільки для створення цифрового підпису і заснований на обчислювальній складності взяття логарифмів у кінцевих полях. За безпекою і продуктивністю повністю можна порівняти з RSA.

ECC (Elliptic Curve Cryptography) визначає пару ключів за допомогою точок на кривій і використовується тільки для цифрового підпису. Основною перевагою алгоритму є вищий рівень надійності за меншої довжини ключа (256-бітний ECC-ключ можна порівняти за надійністю з 3072-бітовим RSA-ключем.

Коротший ключ також впливає на час обробки даних, який помітно скорочується. Цей факт і те, що алгоритм ефективно обробляє велику кількість підключень, зробили його зручним інструментом для роботи з мобільним зв'язком. У SSL-сертифікатах можна використовувати відразу кілька методів шифрування для більшого захисту.

Окрім цього варто не забувати про хеш-функції та MAC, що також приймають активну участь у роботі SSL.

Мета хеш-алгоритму - перетворювати весь вміст SSL-сертифіката в бітовий рядок фіксованої довжини. Для шифрування значення хешу застосовується закритий ключ центру сертифікації, який включається в сертифікат як підпис.

Хеш-алгоритм також використовує величину, необхідну для перевірки цілісності переданих даних - MAC (message authentication code). MAC використовує функцію відображення, щоб представляти дані повідомлення як фіксоване значення довжини, а потім хешує повідомлення.

У протоколі TLS застосовується HMAC (hashed message authentication code), який використовує хеш-алгоритм відразу зі спільним секретним ключем. Тут ключ прикріплюється до даних, і для підтвердження їхньої автентичності обидві сторони мають використовувати однакові секретні ключі, що забезпечує більшу безпеку.

Усі алгоритми шифрування сьогодні підтримують алгоритм хешування SHA2, найчастіше саме SHA-256. SHA-512 має схожу структуру,

але в ньому довжина слова дорівнює 64 біти (замість 32), кількість раундів у циклі дорівнює 80 (замість 64), а повідомлення розбивається на блоки по 1024 біти (замість 512 біт). Раніше для тих самих цілей застосовували алгоритм SHA1 і MD5, але сьогодні їх вважають уразливими.

1.1.2 Види сертифікатів

Domain Validation, або сертифікати з перевіркою домену, підходять для некомерційних сайтів, оскільки вони підтверджують тільки веб-сервер, який обслуговує певний сайт, на який було здійснено перехід. Цей вид сертифіката найдешевший і найпопулярніший, але не може вважатися повністю безпечним, оскільки містить тільки інформацію про зареєстроване доменне ім'я.

Organization Validation, або сертифікати з перевіркою організації, є більш надійними, оскільки підтверджують ще реєстраційні дані компанії-власника. Цю інформацію юридична особа зобов'язана надати під час купівлі сертифіката, а засвідчувальний центр може зв'язатися безпосередньо з компанією для підтвердження цієї інформації. Сертифікат відповідає стандартам RFC і містить інформацію про те, хто його підтвердив, але дані про власника не відображаються.

Extended Validation, або сертифікат із розширеною перевіркою, вважається найнадійнішим. Власне, зелений замочок або ярлик у браузері означає якраз те, що у сайту є саме такий сертифікат.

Крім того, сертифікати можуть відрізнятися залежно від кількості доменів, на які вони були видані. Однодоменні сертифікати (Single Certificate) прив'язуються до одного домену, який вказується під час купівлі. Мультидоменні сертифікати (типу Subject Alternative Name, Unified Communications Certificate, Multi Domain Certificate) діятимуть уже для більшої кількості доменних імен і серверів, які також визначаються під час замовлення. Однак за включення додаткових доменів, понад визначену норму, потрібно буде платити окремо.

Ще існують піддоменні сертифікати (типу WildCard), які охоплюють усі піддомени зазначеного під час реєстрації доменного імені. Іноді можуть знадобитися сертифікати, які будуть одночасно охоплювати не тільки кілька доменів, а й піддомени. У таких випадках можна придбати сертифікати типу Comodo PositiveSSL Multi-Domain Wildcard і Comodo Multi-Domain Wildcard SSL.

1.2 Історія протоколів

1) **SSL 1.0** — перша версія, що ніколи не була у публічному використанні через ряд технічних проблем із вразливістю.

2) **SSL 2.0** — друга версія, але перша публічна, випущена у 1995 році. Дозволяла використовувати криптографію для шифрування трафіку, містила геш-функції та алгоритми шифрування (слабкі). Але містила ще ряд проблем з вразливістю до атак, одною з яких є MitM.

3) **SSL 3.0** — стала покращенням попередньої версії. Було додано можливість використання HMAC, алгоритми cipher suites(алгоритмів через механізм узгодження параметрів). Але все ще була вразлива до деяких атак, одна з них це POODLE (Padding Oracle On Downgraded Legacy Encryption).

4) **TLS 1.0** — хоча протокол має вже іншу назву, по суті є продовженням покращення попереднього алгоритму SSL 3.0. Має більший захист від атак на хеш-фукції (через HMAC), підтримує Forward Secrecy за Diffie-Hellman та включав у себе більше нових алгоритмів шифрування. Однак, він мав вразливість до атаки BEAST (блокові шифри) та використовував слабку SHA-1.

5) **TLS 1.1** — був покращений захист (автентифікація, захист від повторних атак) а також усунуто вразливості з атакою BEAST. Однак, все ще використовував SHA-1.

6) **TLS 1.2** — було накінець впроваджено стійку геш-функцію SHA-256. Окрім цього була додана підтримка AEAD алгоритмів (шифрування + автентифікація, AES-GCM). Однак мав деякі проблеми з технічної сторони,

помилки у конфігурації і потребував кращого захисту від атак на зниження рівня безпеки.

7) **TLS 1.3** — остання версія протоколу, що була випущена у 2018 році. З цієї версії прибрали усі застарілі й вразливі алгоритми. Також було впроваджено ряд технічних покращень процесу з'єднання (зменшення кількості раундів до одного для повторного з'єднання, шифрування трафіку починається раніше). Основною проблемою нового протоколу є поки несумісність з усіма системами (в основному зі старими).

1.3 TLS протокол

Оскільки процедура і сертифікати не сильно видозмінені, ми пройдемося по основним моментам, що містить TLS. І зосередимося ми на останній версії TLS 1.3 (RFC 8446), сьомій ітерації протоколу SSL / TLS, що десять років покращувалася, щоб замінити TLS 1.2 у зв'язку з намаганням подолати різні види можливих атак. Але зрештою TLS 1.3 було випущено, і він отримав низку значних поліпшень, що забезпечують неперевершену продуктивність і конфіденційність порівняно з попередніми версіями. Крім того, починаючи з найпершої версії TLS 1.3, випущеної 17 квітня 2014 року, аж до 28-ї й до фінальної версії, всі проєкти були протестовані та перевірені технічними гігантами, такими як Google, Mozilla, Cloudflare і багатьма іншими.

TLS 1.3 значно поліпшив свою роботу, внісши зміни в протокол рукостискання. У TLS 1.3 потрібна всього одна передача в обидва кінці, щоб прискорити підключення до сайту. Оскільки кількість переговорів між клієнтом і сервером скоротилася з чотирьох до двох, обмін ключами і схема цифрового підпису через розширення більше не потрібні. Оскільки все відбувається в мілісекундах, це не помічається так швидко, але це має значення. Ба більше, TLS 1.3 пішов далі і зробив свій крок уперед, щоб увімкнути рукостискання 0-RTT ще до того, як клієнт і сервер зустрілися, через що для встановлення рукостискання потрібно нуль циклів, що

призводить до зниження затримки, це більше схоже на етап, тому що завдяки виконанню 0-RTT Resuration клієнт може під'єднатися до сервера, навіть до того, як TLS 1.3 дозволить квітування з нульовим циклом. Зазвичай це виконується шляхом зберігання секретної інформації, такої як ідентифікатор сеансу або квитки попередніх сеансів, і використання їх для майбутнього використання щоразу, коли обидві сторони підключаються.

1) ClientHello, Supported Cipher Suites, Guessing of Key Agreement Protocol, Key Share Починається з повідомлення «ClientHello», клієнт також відправляє список підтримуваних комплектів шифрів, вгадуючи, який протокол угоди про ключі буде обраний сервером. Після, клієнт також надсилає свій ключ для цього протоколу угоди.

2) ServerHello, Key Agreement Protocol, Key Share, Server Finished Сервер відповідає «ServerHello», повідомляє обраний протокол угоди, ключ сервера як сертифікат для спілкування і, після, повідомлення «ServerFinished».

3) Checks Certificate, Generate Keys, Client Finished Далі, клієнт перевіряє сертифікат сервера, генерує свої ключі зі спільного ключа сервера і надсилає повідомлення «Client Finished», що означає, що симетричне шифрування даних може бути розпочато.

Багато застарілих шифрів було видалено із нової версії протоколу. Для кращого розуміння наведено відповідно алгоритми, що включав TLS 1.2: А

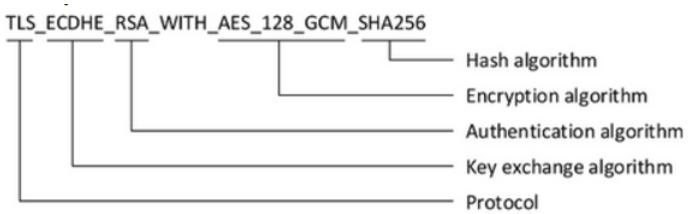

Рисунок 1.1 – TLS 1.2

також новий TLS 1.3:

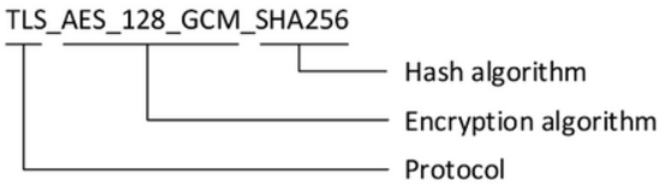

Рисунок 1.2 – TLS 1.3

Набори шифрів реєструються і зберігаються в Реєстрі наборів шифрів TLS IANA, даючи кожному комплекту шифрів свій унікальний номер для ідентифікації. Таким чином, набори шифрів, визначені для TLS 1.3, не можуть використовуватися з TLS 1.2 і навпаки, навіть якщо вони використовують одні й ті самі набори шифрів. Крім того, визначений алгоритм шифрування набору шифрів TLS 1.3 має бути алгоритмом AEAD (Authenticated Encryption with Additional Data), оскільки він забезпечує як конфіденційність, так і автентифікацію повідомлень в одному криптографічному алгоритмі. Тут основна концепція функцій цілісності та шифрування була змінена за допомогою алгоритму AEAD. У зв'язку з введенням AEAD і скороченням підтримуваного шифру не буде можливості надсилати незашифровані дані через TLS 1.3, який був опцією в TLS 1.2 з використанням наборів шифрів NULL.

Наведемо нижче конфігурації алгоритмів:

– $TLS\_AES\_256\_GCM\_SHA384$;

– $TLS\_CHACHA20\_POLY1305\_SHA256$;

– $TLS\_AES\_128\_GCM\_SHA256$;

– $TLS\_AES\_128\_CCM\_8\_SHA256$;

– $TLS\_AES\_128\_CCM\_SHA256$.

1.4 Відомі атаки

Історія протоколів бачила безліч атак, величезну кількість яких вдалося побороти. Наведемо деякі досить відомі атаки на SSL та TLS:

– **SSLstrip.** Ідея полягає в тому, що жертва і зловмисник спілкуються через HTTP, а зловмисник і сервер спілкуються через HTTPS із сертифікатом сервера. Таким чином, зловмисник може бачити весь відкритий текстовий трафік жертви.

– **BEAST.** Під час атаки «поблочно з обраним відкритим текстом» зловмисник може зашифрувати будь-який відкритий текст за своїм вибором у блоках. Використовувана ентропія потім може бути виведена з відомого йому відкритого тексту і спостережуваного ключового тексту. Це значно знижує зусилля, необхідні для злому шифрування.

– **CRIME.** Сутність атаки полягає у розшифруванні файлів cookie, що мають відношення до https-сесії. Основними факторами застосовності атаки є SSL, стиснення даних та SPDY — http-подібний протокол, розроблений Google. Реалізація всіх трьох факторів створює можливість використання експлоїту.

– **BREACH.** Хакер повинен перехоплювати весь користувацький трафік. Додаючи в нього свої дані, він надсилає запити на сервер. Алгоритм стиснення при повторенні частини тексту стискає його. Після надсилання даних на сервер, хакер дивиться за довжиною повідомлення, і якщо вона зменшилася, значить підібрано правильну послідовність. Тим самим можна підбирати окремі рядки і параметри в запиті, наприклад, значення кукі. Але є обмеження - частину даних потрібно знати, щоб від неї почати підбирати решту інформації.

– **POODLE.** Вид атаки в комп'ютерній безпеці типу «людина посередині», коли зловмисник шляхом блокування TLS 1.0 і збільшення кількості спроб з'єднання спричиняє примусове використання інтернет-клієнтами та користувачами захисного програмного забезпечення SSL версії 3.0. Після того як було зроблено відкат системи до SSL 3.0, зловмисник використовує атаку Padding Oracle.

– **FREAK.** Уразливість полягає в недостатній перевірці при виконанні TLS Handshake на стороні клієнта, що призводить до можливості знизити шифрування під час виконання атаки «людина посередині» до використання

512-бітних ключів RSA (RSA_EXPORT ключі), які можуть бути підібрані зловмисником протягом декількох годин. Уразливість була знайдена в 2015 році, а сама помилка існувала з 1990-х

ВИСНОВКИ

В даній роботі ми розглянули різні протоколи захисту трафіку та шифрування в мережі. Можна вказати, що варто користуватися останньою версією TLS 1.3, що є найбезпечнішою серед усіх. Також варто слідкувати за оновленнями постійно, оскільки технології покращуються разом з атаками, що можуть скомпрометувати систему яка була ще досить захищеною вчора.