TLS

Transport Layer Security (**TLS**) este un protocol criptografic conceput pentru a oferi securitate comunicațiilor intr-o rețea de calculatoare. Protocolul este utilizat pe scară largă în aplicații precum e-mail, mesagerie instantanta și Voice over IP, dar utilizarea sa în securizarea HTTPS rămâne cea mai vizibilă public.

Protocolul TLS urmărește în primul rând să ofere securitate, inclusiv Privacy (confidențialitate), integritate și autenticitate prin utilizarea criptografiei, cum ar fi utilizarea certificatelor, între două sau mai multe aplicații care comunică. Acesta rulează în Prezentation Layer și este el însuși compus din două straturi: înregistrarea TLS și protocoalele de Handshake TLS.

DTLS (**Datagram Transport Layer Security**) este strâns legată este un protocol de comunicații care oferă securitate aplicațiilor bazate pe datagrame. În scrierea tehnică, trimiterile la „(**D**) **TLS** ” se fac atunci când se aplică ambelor versiuni.

TLS este un standard Internet Engineering Task Force (IETF), definit pentru prima dată în 1999, iar versiunea actuală este TLS 1.3, definită în august 2018. TLS se bazează pe specificațiile **SSL** (**Secure Sockets Layer**) acum demodate (1994, 1995, 1996) dezvoltat de Netscape Communications pentru adăugarea protocolului HTTPS la browser-ul lor web Navigator.

Descriere

Aplicațiile client-server folosesc protocolul TLS pentru a comunica printr-o rețea într-un mod conceput pentru a preveni interceptarea și manipularea.

Deoarece aplicațiile pot comunica fie cu sau fără TLS (sau SSL), clientul trebuie să solicite serverului să stabilească o conexiune TLS. Una dintre principalele modalități de a realiza acest lucru este utilizarea unui număr de port diferit pentru conexiunile TLS. *Portul 80* este utilizat de obicei pentru traficul HTTP necriptat, în timp ce *portul 443* este portul comun utilizat pentru traficul HTTPS criptat. Un alt mecanism este de a face o solicitare STARTTLS specifică protocolului către server pentru a comuta conexiunea la TLS – de exemplu, atunci când utilizați protocoalele de e-mail și alte protocoale.

Odată ce clientul și serverul au fost de acord să utilizeze TLS, ei negociază o conexiune cu stare, folosind o procedură de Handshake (vezi § Handshake TLS). Protocoalele folosesc Handshake cu un cifru asimetric pentru a stabili nu numai setările de cifrare, ci și o cheie partajată specifică sesiunii, cu care comunicarea ulterioară este criptată folosind un cifru simetric. În timpul acestei Handshake, clientul și serverul convin asupra diferiților parametri utilizați pentru a stabili securitatea conexiunii:

- Handshake-ul începe atunci când un client se conectează la un server compatibil TLS solicitând o conexiune securizată, iar clientul prezintă o listă de suite de cifrare acceptate (funcții de cifrare și hash).
- Din această listă, serverul alege o funcție de criptare și hash pe care o suportă, de asemenea, și notifică clientul de decizie.
- Serverul oferă de obicei identificarea sub forma unui certificat digital. Certificatul conține numele serverului, autoritatea de certificare (CA) de încredere care garantează autenticitatea certificatului și cheia publică de criptare a serverului.
- Clientul confirmă valabilitatea certificatului înainte de a continua.
- Pentru a genera cheile de sesiune utilizate pentru conexiunea securizată, clientul:
    - criptează un număr aleatoriu (*PreMasterSecret*) cu cheia publică a serverului și trimite rezultatul către server (pe care doar serverul ar trebui să-l poată decripta cu cheia sa privată); ambele părți folosesc apoi numărul aleatoriu pentru a genera o cheie unică de sesiune pentru criptarea și decriptarea ulterioară a datelor în timpul sesiunii sau
    - folosește schimbul de chei ***Diffie–Hellman**** (sau varianta sa cu curbă eliptică DH) pentru a genera în siguranță o cheie de sesiune aleatoare și unică pentru criptare și decriptare care are proprietatea suplimentară de secretizare directă - **Forward secrecy** (**FS**) : dacă cheia privată a serverului este dezvăluită în viitor, aceasta nu poate fi folosita pentru a decripta sesiunea curentă, chiar dacă sesiunea este interceptată și înregistrată de o terță parte.

Aceasta încheie Handshake-ul și începe conexiunea securizată, care este criptată/ decriptată cu cheia de sesiune până la închiderea conexiunii. Dacă oricare dintre pașii de mai sus nu reușește, atunci Handshake-ul TLS eșuează și conexiunea nu este creată.

TLS și SSL nu se potrivesc perfect în nici un singur strat al modelului OSI** sau al modelului TCP/IP. TLS rulează deasupra unui protocol de transport fiabil (de exemplu, TCP), ceea ce ar implica că este deasupra Transport Layer. Servește criptarea straturilor superioare, care este în mod normal funcția Presentation Layer. Cu toate acestea, aplicațiile folosesc în general TLS ca și cum ar fi un Transport Layer, chiar dacă aplicațiile care folosesc TLS trebuie să controleze în mod activ inițierea acordurilor de mână TLS și gestionarea certificatelor de autentificare schimbate.

Când sunt securizate de TLS, conexiunile dintre un client (de exemplu, un browser web) și un vor avea toate următoarele proprietăți:

- Conexiunea este *privată* (sau are *confidențialitate*) deoarece se folosește un algoritm cu cheie simetrică pentru a cripta datele transmise. Cheile pentru această criptare simetrică sunt generate în mod unic pentru fiecare conexiune și se bazează pe un secret partajat care a fost negociat la începutul sesiunii. Serverul și clientul negociază detaliile despre ce algoritm de criptare și cheile criptografice să fie utilizate înainte ca primul octet de date să fie transmis (vezi mai jos). Negocierea unui secret partajat este atât sigură (secretul negociat este indisponibil pentru interceptători și nu poate fi obținut, chiar și de către un atacator care se plasează în mijlocul conexiunii), cât și de încredere (niciun atacator nu poate modifica comunicațiile în timpul negocierii fără a fi detectat).
- Identitatea părților care comunică poate fi *autentificată* folosind criptografia cu cheie publică. Această autentificare este necesară pentru server și opțională pentru client.
- Conexiunea este *fiabilă* (sau are *integritate*) deoarece fiecare mesaj transmis include o verificare a integrității mesajului folosind un cod de autentificare a mesajului pentru a preveni pierderea sau alterarea nedetectată a datelor în timpul transmisiei.

TLS acceptă multe metode diferite pentru schimbul de chei, criptarea datelor și autentificarea integrității mesajelor. Ca rezultat, configurarea securizată a TLS implică mulți parametri configurabili și nu toate opțiunile oferă toate proprietățile legate de confidențialitate descrise în lista de mai.

Au fost făcute încercări de a submina aspecte ale securității comunicațiilor pe care TLS încearcă să le ofere, iar protocolul a fost revizuit de mai multe ori pentru a aborda aceste amenințări de securitate. Dezvoltatorii de browsere web și-au revizuit în mod repetat produsele pentru a se apăra împotriva potențialelor deficiențe de securitate după ce acestea au fost descoperite (consultați istoricul suportului TLS/SSL al browserelor web).

Datagram Transport Layer Security

Datagram Transport Layer Security, abreviat DTLS, este un protocol de comunicații înrudit care ofera securitate aplicațiilor bazate pe datagrame, permițându-le să comunice într-un mod conceput pentru a preveni interceptarea sau falsificarea mesajelor. Protocolul DTLS se bazează pe protocolul Transport Layer Security (TLS) orientat spre flux și are scopul de a oferi garanții de securitate similare. Cu toate acestea, spre deosebire de TLS, acesta poate fi utilizat cu majoritatea protocoalelor orientate pe datagramă, inclusiv Protocolul de datagramă utilizator (UDP), Protocolul de control al congestionării datelor (DCCP), Controlul și furnizarea punctelor de acces fără fir (CAPWAP), încapsularea Protocolului de transmisie de control al fluxului (SCTP), și Secure Real-time Transport Protocol (SRTP).

Deoarece datagrama protocolului DTLS păstrează semantica transportului subiacent - aplicația nu suferă de întârzierile asociate cu protocoalele de flux, totuși aplicația trebuie să se ocupe de reordonarea pachetelor, pierderea de datagramă și de date mai mari decât dimensiunea unei packet datagramă de retea. Deoarece DTLS folosește UDP sau SCTP mai degrabă decât TCP, evită „problema de topire a TCP”, atunci când este folosit pentru a crea un tunel VPN.

Lansarea originală din 2006 a DTLS versiunea 1.0 nu a fost un document independent. A fost dat ca o serie de delte la TLS 1.1. În mod similar, versiunea ulterioară din 2012 a DTLS este o deltă față de TLS 1.2. I s-a dat numărul versiunii DTLS 1.2 pentru a se potrivi cu versiunea sa TLS. În cele din urmă, 2022 DTLS 1.3 este o deltă față de TLS 1.3. La fel ca cele două versiuni anterioare, DTLS 1.3 este menit să ofere „garanții de securitate echivalente [cu TLS 1.3], cu excepția protecției comenzilor/nerejucabilitatea”.

Mulți clienți VPN, inclusiv Cisco AnyConnect și InterCloud Fabric, OpenConnect, tunel ZScaler, F5 Networks Edge VPN Client, și Citrix Systems NetScaler folosesc DTLS pentru a securiza traficul UDP. În plus, toate browserele web moderne acceptă DTLS-SRTP pentru WebRTC.

Istoric

Protocoale SSL și TLS		
Protocol	**Publicat**	**Status**
SSL 1.0	Nepublicat	Nepublicat
SSL 2.0	1995	Învechit în 2011 (RFC 6176)
SSL 3.0	1996	Învechit în 2015 (RFC 7568)
TLS 1.0	1999	Depreciat în 2021 (RFC 8996) [20] [21] [22]
TLS 1.1	2006	Depreciat în 2021 (RFC 8996) [20] [21] [22]
TLS 1.2	2008	În uz din 2008 [23] [24]
TLS 1.3	2018	În uz din 2018 [24] [25]

Sistem de rețea de date securizat

Protocolul TLS, împreună cu alte câteva platforme de bază de securitate a rețelei, a fost dezvoltat printr-o inițiativă comună începută în august 1986, între Agenția Națională de Securitate, Biroul Național de Standarde, Agenția de Comunicații pentru Apărare și douăsprezece comunicații și comunicații. corporații informatice care au inițiat un proiect special numit Secure Data Network System (SDNS). Programul a fost descris în septembrie 1987 la a 10-a Conferință Națională de Securitate a Calculatoarelor într-un set extins de lucrări publicate. Programul inovator de cercetare s-a concentrat pe proiectarea următoarei generații de rețele de comunicații computerizate securizate și specificații de produs care să fie implementate pentru aplicații pe internetul public și privat. Acesta a fost menit să completeze noile standarde de internet OSI care au apărut rapid, care avansează atât în profilurile GOSIP ale guvernului SUA, cât și în uriașul efort de internet ITU-ISO JTC1 la nivel internațional. Cunoscut inițial ca protocol SP4, a fost redenumit TLS și ulterior publicat în 1995 ca standard internațional ITU-T X.274|ISO/IEC 10736:1995.

Programare în rețea securizată

Eforturile timpurii de cercetare pentru Transport Layer Security au inclus interfața de programare a aplicațiilor (API) Secure Network Programming (SNP), care în 1993 a explorat abordarea de a avea un API de nivel de transport securizat care seamănă foarte mult cu socket-urile Berkeley, pentru a facilita adaptarea aplicațiilor de rețea preexistente cu securitate. Măsuri.

SSL 1.0, 2.0 și 3.0

Netscape a dezvoltat protocoalele SSL originale, iar Taher Elgamal, om de știință șef la Netscape Communications din 1995 până în 1998, a fost descris drept „părintele SSL”. SSL versiunea 1.0 nu a fost niciodată lansată public din cauza unor defecte serioase de securitate în protocol. Versiunea 2.0, după ce a fost lansată în februarie 1995, sa descoperit rapid că conține o serie de defecte de securitate și de utilizare. A folosit aceleași chei criptografice pentru autentificarea și criptarea mesajelor. Avea o construcție MAC slabă care folosea funcția hash MD5 cu un prefix secret, făcându-l vulnerabil la atacurile de extensie de lungime. De asemenea, nu a oferit nicio protecție nici pentru deschiderea Handshake, nici pentru un mesaj explicit de închidere, ambele însemnând că atacurile de tip om-in-the-middle ar putea rămâne nedetectate. Mai mult, SSL 2.0 presupunea un singur serviciu și un certificat de domeniu fix, în conflict cu caracteristica larg utilizată de găzduire virtuală în serverele Web, astfel încât majoritatea site-urilor web au fost efectiv afectate de utilizarea SSL.

Aceste defecte au necesitat reproiectarea completă a protocolului la versiunea SSL 3.0. Lansat în 1996, a fost produs de Paul Kocher, care lucrează cu inginerii Netscape Phil Karlton și Alan Freier, cu o implementare de

referință de către Christopher Allen și Tim Dierks de la Certicom. Versiunile mai noi de SSL/TLS se bazează pe SSL 3.0. Schița din 1996 a SSL 3.0 a fost publicată de IETF ca document istoric în RFC 6101.

SSL 2.0 a fost depreciat în 2011 de RFC 6176. În 2014, SSL 3.0 s-a dovedit a fi vulnerabil la atacul POODLE care afectează toate cifrurile bloc din SSL; RC4, singurul cifru non-block acceptat de SSL 3.0, este, de asemenea, fezabil spart așa cum este utilizat în SSL 3.0. SSL 3.0 a fost depreciat în iunie 2015 de RFC 7568.

TLS 1.0

TLS 1.0 a fost definit pentru prima dată în RFC 2246 în ianuarie 1999 ca o actualizare a SSL Versiunea 3.0 și scris de Christopher Allen și Tim Dierks de la Certicom. După cum se precizează în RFC, „diferențele dintre acest protocol și SSL 3.0 nu sunt dramatice, dar sunt suficient de semnificative pentru a exclude interoperabilitatea între TLS 1.0 și SSL 3.0". Tim Dierks a scris mai târziu că aceste modificări, precum și redenumirea de la „SSL" la „TLS", au fost un gest de salvare a feței pentru Microsoft, „deci nu ar părea [ca] IETF doar a marcat protocolul Netscape".

Consiliul PCI a sugerat ca organizațiile să migreze de la TLS 1.0 la TLS 1.1 sau o versiune ulterioară înainte de 30 iunie 2018. În octombrie 2018, Apple, Google, Microsoft și Mozilla au anunțat împreună că vor renunța la TLS 1.0 și 1.1 în martie. 2020. TLS 1.0 și 1.1 au fost depreciate oficial în RFC 8996 în martie 2021.

TLS 1.1

TLS 1.1 a fost definit în RFC 4346 în aprilie 2006. Este o actualizare de la versiunea TLS 1.0. Diferențele semnificative în această versiune includ:

- Protecție adăugată împotriva atacurilor de tip cipher-block chaining (CBC).
    - Vectorul de inițializare implicită (IV) a fost înlocuit cu un IV explicit.
    - Modificare în gestionarea erorilor de padding.
- Suport pentru înregistrarea IANA a parametrilor.

Suportul pentru versiunile TLS 1.0 și 1.1 a fost larg depreciat de site-urile web în jurul anului 2020, dezactivând accesul la versiunile Firefox și la browserele bazate pe Chromium.

TLS 1.2

TLS 1.2 a fost definit în RFC 5246 în august 2008. Se bazează pe specificația anterioară TLS 1.1. Diferențele majore includ:

- Combinația MD5 și SHA-1 în funcția pseudoaleatorie (PRF) a fost înlocuită cu SHA-256, cu o opțiune de a utiliza (pseudorandom function) PRF-uri specificate în suita de criptare.
- Combinația MD5 și SHA-1 din hash -ul mesajului final a fost înlocuită cu SHA-256, cu o opțiune de a utiliza algoritmi de hash specifici suitei de cifrare. Cu toate acestea, dimensiunea hash-ului din mesajul final trebuie să fie încă de cel puțin 96 de biți.
- Combinația MD5 și SHA-1 din elementul semnat digital a fost înlocuită cu un singur hash negociat în timpul Handshake, care este implicit SHA-1.
- Îmbunătățirea capacității clientului și a serverului de a specifica ce hashuri și algoritmi de semnătură acceptă.
- Extinderea suportului pentru cifrurile de criptare autentificate, utilizate în principal pentru modul Galois/Counter Mode (GCM) și modul CCM de criptare Advanced Encryption Standard (AES).
- Au fost adăugate definiția extensiilor TLS și suitele de criptare AES.

Toate versiunile TLS au fost îmbunătățite în continuare în RFC 6176 în martie 2011, eliminându-le compatibilitatea cu SSL, astfel încât sesiunile TLS să nu negocieze niciodată utilizarea Secure Sockets Layer (SSL) versiunea 2.0. În prezent, nu există o dată oficială pentru ca TLS 1.2 să fie depreciat.

TLS 1.3

TLS 1.3 a fost definit în RFC 8446 în august 2018. Se bazează pe specificația anterioară TLS 1.2. Diferențele majore față de TLS 1.2 include:

- Separarea acordului de cheie și a algoritmilor de autentificare de suitele de criptare
- Eliminarea suportului pentru curbele eliptice numite slabe și mai puțin utilizate
- Eliminarea suportului pentru funcțiile hash criptografice MD5 și SHA-224
- Necesită semnături digitale chiar și atunci când este utilizată o configurație anterioară
- Integrarea HKDF și propunerea semi-efemeră DH
- Înlocuirea reluării cu PSK (pre-shared key) și tickete
- Suport 1- RTT (round trip time) handshake și suport inițial pentru 0- RTT

- Obligarea Forward Secrecy, prin utilizarea cheilor efemere în timpul acordului de cheie (EC)DH
- Renunțarea suportului pentru multe funcții nesigure sau învechite, inclusiv compresie, renegociere, cifruri non- AEAD, schimb de chei non- PFS (printre care sunt schimburile de chei RSA statice și DH statice), grupuri DHE personalizate, negociere cu format de punct EC, protocol Change Cipher Spec, Bună ziua mesajul UNIX, și câmpul de lungime AD introdus la cifrurile AEAD - authenticated encryption with associated data
- Interzicerea negocierii SSL sau RC4 pentru compatibilitate cu versiunea inversă
- Integrarea utilizării hash-ului sesiunii
- Renunțarea la utilizarea numărului de versiune a stratului de înregistrare și înghețarea numărului pentru o compatibilitate îmbunătățită
- Mutarea unor detalii ale algoritmului de securitate dintr-un apendice la specificație și relevarea ClientKeyShare într-o anexă
- Adăugarea cifrului de flux ChaCha20 cu codul de autentificare a mesajului Poly1305
- Adăugarea algoritmilor de semnătură digitală Ed25519 și Ed448
- Adăugarea protocoalelor de schimb de chei x25519 și x448
- Adăugarea suportului pentru trimiterea mai multor răspunsuri OCSP
- Criptarea tuturor mesajelor de Handshake după ServerHello

Network Security Services (NSS), biblioteca de criptografie dezvoltată de Mozilla și utilizată de browserul său web Firefox, a activat implicit TLS 1.3 în februarie 2017. Ulterior a fost adăugat suport pentru TLS 1.3 – dar din cauza problemelor de compatibilitate pentru un număr mic de utilizatori, neactivați automat — la Firefox 52.0, care a fost lansat în martie 2017. TLS 1.3 a fost activat implicit în mai 2018 odată cu lansarea Firefox 60.0.

Google Chrome a stabilit TLS 1.3 ca versiune implicită pentru o perioadă scurtă de timp în 2017. Apoi l-a eliminat ca implicit, din cauza casetelor intermediare incompatibile, cum ar fi proxy-urile web Blue Coat.

Intoleranța noii versiuni de TLS a fost osificarea protocolului ; middlebox-urile osificaseră parametrul de versiune a protocolului. Ca rezultat, versiunea 1.3 imită imaginea versiunii 1.2. Această modificare a avut loc foarte târziu în procesul de proiectare, fiind descoperită doar în timpul implementării browserului. Descoperirea acestei intoleranțe a dus și la strategia de negociere a versiunii anterioare, unde a fost aleasă versiunea cea mai potrivită, fiind abandonată din cauza nivelurilor de osificare imposibil de realizat. „ Ungerea ” unui punct de extensie, în care un participant la protocol pretinde sprijin pentru extensii inexistente pentru a se asigura că sunt tolerate extensiile nerecunoscute, dar existente, și astfel pentru a rezista osificării, a fost conceput inițial pentru TLS.

În timpul IETF 100 Hackathon, care a avut loc în Singapore în 2017, Grupul TLS a lucrat la adaptarea aplicațiilor open-source pentru a utiliza TLS 1.3. Grupul TLS a fost format din persoane din Japonia, Regatul Unit și Mauritius prin intermediul echipei cyberstorm. Această activitate a fost continuată în IETF 101 Hackathon din Londra, și IETF 102 Hackathon din Montreal.

Libraria wolfSSL a permis utilizarea TLS 1.3 începând cu versiunea 3.11.1, lansată în mai 2017. Ca prima implementare comercială TLS 1.3, wolfSSL 3.11.1 a acceptat Draft 18 și acum acceptă Draft 28, versiunea finală, precum și multe versiuni mai vechi. Au fost publicate o serie de bloguri despre diferența de performanță dintre TLS 1.2 și 1.3.

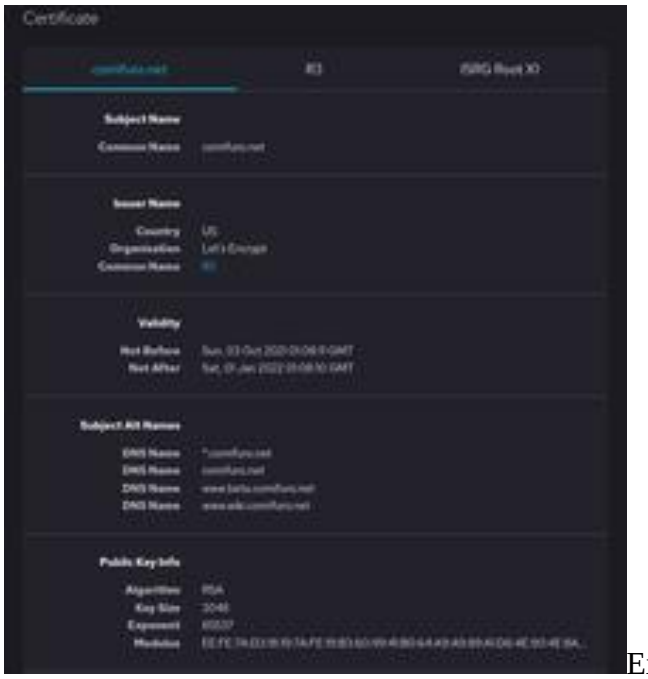
Înseptembrie 2018, popularul proiect OpenSSL a lansat versiunea 1.1.1 a bibliotecii sale, în care suportul pentru TLS 1.3 era „noua caracteristică principală”.

Suportul pentru TLS 1.3 a fost adăugat pentru prima dată la Schannel cu Windows 11 și Windows Server 2022.

Enterprise Transport Security

Electronic Frontier Foundation a lăudat TLS 1.3 și și-a exprimat îngrijorarea cu privire la varianta protocolului Enterprise Transport Security (ETS) care dezactivează în mod intenționat măsurile de securitate importante în TLS 1.3. Denumit inițial Enterprise TLS (eTLS), ETS este un standard publicat cunoscut sub numele de „ ETSI TS103523-3”, „Middlebox Security Protocol, Part3: Enterprise Transport Security”. Este destinat utilizării în întregime în cadrul rețelelor proprietare, cum ar fi sistemele bancare. ETS nu acceptă secretizarea directă, astfel încât să permită organizațiilor terțe conectate la rețelele proprietare să își poată folosi cheia privată pentru a monitoriza traficul de rețea pentru detectarea programelor malware și pentru a facilita efectuarea auditurilor. În ciuda beneficiilor pretinse, EFF a avertizat că pierderea secretului de transmitere ar putea face mai ușoară expunerea datelor și a spus că există modalități mai bune de a analiza traficul.

Certificate digitale



Exemplu de site web cu certificat digital

Un certificat digital certifică proprietarul unei chei publice și indică anumite utilizări așteptate ale acelei chei. Acest lucru permite altora să se bazeze pe semnăturile sau pe cheia privată care corespunde cheii publice certificate. Depozitele de chei de încredere pot fi în diferite formate, cum ar fi.pem,.crt,.pfx și.jks.

Autorități de certificare

TLS se bazează de obicei pe un set de autorități de certificare terțe de încredere pentru a stabili autenticitatea certificatelor. Încrederea este de obicei ancorată într-o listă de certificate distribuite cu software-ul agent de utilizator, și poate fi modificată de partea care se bazează.

Potrivit Netcraft, care monitorizează certificatele TLS active, autoritatea de certificare (CA) lider pe piață a fost Symantec de la începutul sondajului lor (sau VeriSign înainte ca unitatea de afaceri de servicii de autentificare să fie achiziționată de Symantec). Începând cu 2015, Symantec a reprezentat puțin mai puțin de o treime din toate certificatele și 44% din certificatele valabile utilizate de cele mai aglomerate site-uri web de 1 milion, calculate de Netcraft. În 2017, Symantec și-a vândut afacerea TLS/SSL către DigiCert. Într-un raport actualizat, sa arătat că IdenTrust, DigiCert și Sectigo sunt primele 3 autorități de certificare în ceea ce privește cota de piață din mai 2019.

Ca o consecință a alegerii certificatelor X.509, autoritățile de certificare și o infrastructură de chei publice sunt necesare pentru a verifica relația dintre un certificat și proprietarul acestuia, precum și pentru a genera, semna și administra valabilitatea certificatelor. Deși acest lucru poate fi mai convenabil decât verificarea identităților printr-o rețea de încredere, dezvăluirile de supraveghere în masă din 2013 au făcut cunoscut mai mult faptul că autoritățile de certificare sunt un punct slab din punct de vedere al securității, permițând atacuri „man-in-the-middle” (MITM) dacă autoritatea de certificare cooperează (sau este compromisă).

Importanța certificatelor SSL

- **Criptare**: certificatele SSL criptează datele trimise între un server web și browserul unui utilizator, asigurând că informațiile sensibile sunt protejate pe tot parcursul transmisiei. Această tehnologie de criptare împiedică părțile neautorizate să intercepteze și să interpreteze datele, protejându-le astfel de posibile riscuri, cum ar fi hacking sau încălcarea datelor.
- **Autentificare**: certificatele SSL oferă, de asemenea, autentificare, care atestă integritatea unui site web și că vizitatorii se conectează la serverul corect, mai degrabă decât la un impostor rău intenționat. Această metodă de autentificare ajută consumatorii să câștige încredere, asigurându-se că au de-a face cu un site web de încredere și sigur.
- **Integritate**: Un alt rol important al certificatelor SSL este acela de a asigura integritatea datelor. SSL folosește tehnici criptografice pentru a verifica dacă datele comunicate între server și browser sunt intacte și nemodificate în timpul tranzitului. Acest lucru îi împiedică pe actorii răuvoitori să interfereze cu datele, asigurând integritatea și credibilitatea acestora.

Algoritmi

Schimb de chei

Înainte ca un client și un server să poată începe să facă schimb de informații protejate de TLS, aceștia trebuie să schimbe în siguranță sau să convină asupra unei chei de criptare și a unui cifru pe care să le folosească la criptarea datelor. Printre metodele utilizate pentru schimbul/acordul de chei se numără: cheile publice și private generate cu RSA (denotat TLS_RSA în protocolul de Handshake TLS), Diffie–Hellman (TLS_DH), Diffie–Hellman efemer (TLS_DHE), Diffie–Hellman cu curbă eliptică (TLS_ECDH), Diffie–Hellman cu curbă eliptică efemeră (TLS_ECDHE), Diffie–Hellman anonim (TLS_DH_anon), cheie pre-partajată (TLS_PSK) și Parolă securizată la distanță (TLS_SRP).

Metodele de acord cu cheile TLS_DH_anon și TLS_ECDH_anon nu autentifică serverul sau utilizatorul și, prin urmare, sunt rareori utilizate, deoarece acestea sunt vulnerabile la atacurile man-in-the-middle. Doar TLS_DHE și TLS_ECDHE asigură Forward secrecy.

Certificatele de chei publice utilizate în timpul schimbului/acordului variază, de asemenea, în ceea ce privește dimensiunea cheilor de criptare publice/private utilizate în timpul schimbului și, prin urmare, robustețea securității furnizate. În iulie 2013, Google a anunțat că nu va mai folosi chei publice de 1024 de biți și că va trece în schimb la chei de 2048 de biți pentru a crește securitatea criptării TLS pe care o oferă utilizatorilor săi, deoarece puterea de criptare este direct legată de dimensiunea cheii.

Schimb de chei/acord și autentificare

Algoritm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	stare
RSA	da	da	da	da	da	Nu	Definit pentru TLS 1.2 în RFC-uri
DH - RSA	Nu	da	da	da	da	Nu	
DHE - RSA (Forward secrecy)	Nu	da	da	da	da	da	
ECDH - RSA	Nu	Nu	da	da	da	Nu	
ECDHE - RSA (Forward secrecy)	Nu	Nu	da	da	da	da	
DH - DSS	Nu	da	da	da	da	Nu	
DHE - DSS (Forward secrecy)	Nu	da	da	da	da	Nu [73]	
DHE - ECDSA (Forward secrecy)	Nu	Nu	Nu	Nu	Nu	da	
ECDH - ECDSA	Nu	Nu	da	da	da	Nu	

ECDHE - ECDSA (Forward secrecy)	Nu	Nu	da	da	da	da	
DHE - EdDSA (Forward secrecy)	Nu	Nu	Nu	Nu	Nu	da	
ECDH - EdDSA	Nu	Nu	da	da	da	Nu	
ECDHE - EdDSA (Forward secrecy)	Nu	Nu	da	da	da	da	
PSK	Nu	Nu	da	da	da	da	
RSA - PSK	Nu	Nu	da	da	da	Nu	
DHE - PSK (Forward secrecy)	Nu	Nu	da	da	da	da	
ECDHE - PSK (Forward secrecy)	Nu	Nu	da	da	da	da	
SRP	Nu	Nu	da	da	da	Nu	
SRP - DSS	Nu	Nu	da	da	da	Nu	
SRP - RSA	Nu	Nu	da	da	da	Nu	
Kerberos	Nu	Nu	da	da	da	?	
DH -ANON (nesigur)	Nu	da	da	da	da	Nu	
ECDH -ANON (nesigur)	Nu	Nu	da	da	da	Nu	
GOST R 34.10-2012	Nu	Nu	Nu	Nu	da	da	Definit pentru TLS 1.2 și pentru TLS 1.3 în RFC 9189, 9367.

Cifrare

Securitate cifrată împotriva atacurilor fezabile cunoscute public

Cifru			Versiunea protocolului						Status
Tip	Algoritm	Forța nominală (biți)	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	
Cifra bloc cu modul de operare	AES GCM	256, 128	—	—	—	—	Sigur	Sigur	Definit pentru TLS 1.2 în RFC-uri
	AES CCM		—	—	—	—	Sigur	Sigur	
	AES CBC		—	Nesigur	Depinde de atenuări	Depinde de atenuări	Depinde de atenuări	—	
	Camellia GCM	256, 128	—	—	—	—	Sigur	—	
	Camellia CBC		—	Nesigur	Depinde de atenuări	Depinde de atenuări	Depinde de atenuări	—	
	ARIA GCM	256, 128	—	—	—	—	Sigur	—	
	ARIA CBC		—	—	Depinde de atenuări	Depinde de atenuări	Depinde de atenuări	—	
	SEED CBC	128	—	Nesigur	Depinde de atenuări	Depinde de atenuări	Depinde de atenuări	—	
	3DES EDE CBC	112	Nesigur	Nesigur	Nesigur	Nesigur	Nesigur	—	
	GOST R 34.12-2015 Magma CTR	256	—	—	Nesigur	Nesigur	Nesigur	—	Definit în RFC 4357, 9189
	GOST R 34.12-2015 Kuznyechik CTR	256	—	—	—	—	Sigur	—	Definit în RFC 9189

	GOST R 34.12-2015 Magma MGM	256	—	—	—	—	—	Nesigur	Definit în RFC 9367
	GOST R 34.12-2015 Kuznyechik MGM	256	—	—	—	—	—	Sigur	Definit în RFC 9367
	IDEA CB[]	128	Nesigur	Nesigur	Nesigur	Nesigur	—	—	Eliminat din TLS 1.2
	DES CBC	56	Nesigur	Nesigur	Nesigur	Nesigur	—	—	
		40	Nesigur	Nesigur	Nesigur	—	—	—	Interzis în TLS 1.1 și versiuni ulterioare
	RC2 CBC	40	Nesigur	Nesigur	Nesigur	—	—	—	
Cifrarea fluxului	**ChaCha20 - Poly1305**	256	—	—	—	—	Sigur	Sigur	Definit pentru TLS 1.2 în RFC-uri
	RC4	128	Nesigur	Nesigur	Nesigur	Nesigur	Nesigur	—	Interzis în toate versiunile de TLS prin RFC 7465
		40	Nesigur	Nesigur	Nesigur	—	—	—	
Niciunul	**Nulă**	–	Nesigur	Nesigur	Nesigur	Nesigur	Nesigur	—	Definit pentru TLS 1.2 în RFC-uri

Note

1. RFC5746 trebuie implementat pentru a remedia un defect de renegociere care altfel ar sparge acest protocol.
2. Dacă bibliotecile implementează remedieri enumerate în RFC 5746, aceasta încalcă specificația SSL 3.0, pe care IETF nu o poate modifica spre deosebire de TLS. Majoritatea bibliotecilor actuale implementează remedierea și ignoră încălcarea pe care aceasta o provoacă.
3. Atacul BEAST sparge toate cifrurile bloc (cifrele CBC) utilizate în SSL 3.0 și TLS 1.0, dacă nu sunt protejate de client și/sau server.
4. Atacul POODLE sparge toate cifrurile bloc (CBC) utilizate în SSL 3.0, dacă nu sunt atenuate de client și/sau server.
5. Cifrurile AEAD (cum ar fi GCM și CCM) pot fi utilizate numai în TLS 1.2 sau o versiune ulterioară.
6. Cifrurile CBC pot fi sparte cu atacul Lucky Thirteen dacă biblioteca nu este scrisă cu atenție pentru a elimina canalele laterale de sincronizare.
7. AtaculSweet32 sparge criptarile bloc cu o dimensiune a blocului de 64 de biți.

8. Deși lungimea cheii a 3DES este de 168 de biți, puterea efectivă de securitate a 3DES este de numai 112 biți, care este sub minimul recomandat de 128 de biți.
9. IDEA și DES au fost eliminate din TLS 1.2.
10. Suitele de criptare cu putere de 40 de biți au fost proiectate intenționat cu lungimi de chei reduse pentru a se conforma reglementărilor americane anulate de atunci, care interziceau exportul de software criptografic care conține anumiți algoritmi de criptare puternici. Aceste suite slabe sunt interzise în TLS 1.1 și versiuni ulterioare.
11. Utilizarea RC4 în toate versiunile de TLS este interzisă de RFC 7465 (deoarece atacurile pe RC4 slăbesc sau rup RC4 utilizat în SSL/TLS).

Integritatea datelor

Un cod de autentificare a mesajelor (MAC) este utilizat pentru integritatea datelor. HMAC este utilizat pentru modul CBC al cifrurilor bloc. Criptarea autentificată (AEAD), cum ar fi modul GCM și CCM, utilizează MAC integrat în AEAD și nu utilizează HMAC. PRF bazat pe HMAC sau HKDF este folosit pentru Handshake TLS.

Integritatea datelor

Algoritm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	stare
HMAC - MD5	da	da	da	da	da	Nu	Definit pentru TLS 1.2 în RFC-uri
HMAC - SHA1	Nu	da	da	da	da	Nu	
HMAC - SHA256/384	Nu	Nu	Nu	Nu	da	Nu	
AEAD	Nu	Nu	Nu	Nu	da	da	
GOST 28147-89 IMIT	Nu	Nu	Nu	Nu	da	Nu	Definit pentru TLS 1.2 în RFC 9189.
GOST R 34.12-2015 AEAD	Nu	Nu	Nu	Nu	Nu	da	Definit pentru TLS 1.3 în RFC 9367.

Aplicatii

În proiectarea aplicațiilor, TLS este de obicei implementat pe deasupra protocoalelor Transport Layer, criptând toate datele legate de protocol ale protocoalelor, cum ar fi HTTP, FTP, SMTP, NNTP și XMPP.

Din punct de vedere istoric, TLS a fost utilizat în principal cu protocoale de transport fiabile, cum ar fi Transmission Control Protocol (TCP). Cu toate acestea, a fost implementat și cu protocoale de transport orientate pe datagrame, cum ar fi User Datagram Protocol (UDP) și Datagram Congestion Control Protocol (DCCP), a căror utilizare a fost standardizată independent folosind termenul Datagram Transport Layer Security (DTLS).

Site-uri web

O utilizare principală a TLS este securizarea traficului World Wide Web între un site web și un browser web codificat cu protocolul HTTP. Această utilizare a TLS pentru a securiza traficul HTTP constituie protocolul HTTPS.

Suport pentru protocolul site-ului web (sept 2023)

Versiunea protocolului	Asistență pentru site-uri web	Securitate
SSL 2.0	0,2%	Nesigur
SSL 3.0	1,7%	Nesigur
TLS 1.0	30,1%	Învechit
TLS 1.1	32,5%	Învechit
TLS 1.2	99,9%	Depinde de cifrul și atenuările clientului
TLS 1.3	64,8%	Sigur

Browsere web

Începând cu aprilie 2016, cele mai recente versiuni ale tuturor browserelor web majore acceptă TLS 1.0, 1.1 și 1.2 și le au activate implicit. Cu toate acestea, nu toate sistemele de operare Microsoft acceptate acceptă cea mai recentă versiune de IE. În plus, multe sisteme de operare Microsoft acceptă în prezent mai multe versiuni de IE, dar acest lucru s-a schimbat conform întrebărilor frecvente ale Politicii privind ciclul de viață a suportului pentru Internet Explorer de la Microsoft, „începând cu 12 ianuarie 2016, va primi doar cea mai recentă versiune de Internet Explorer disponibilă pentru un sistem de operare acceptat, asistență tehnică și actualizări de securitate.” Pagina continuă apoi cu cea mai recentă versiune de IE acceptată la acea dată pentru fiecare sistem de operare. Următoarea dată critică ar fi atunci când un sistem de operare ajunge la stadiul de sfârșit de viață. Din 15 iunie 2022, Internet Explorer 11 a renunțat la suportul pentru edițiile Windows 10 care urmează Politica modernă a ciclului de viață a Microsoft.

Atenuările împotriva atacurilor cunoscute nu sunt încă suficiente:

- Atenuări împotriva atacului POODLE: unele browsere împiedică deja recursul la SSL 3.0; totuși, această atenuare trebuie să fie susținută nu numai de clienți, ci și de servere. Este necesară dezactivarea SSL 3.0 în sine, implementarea „divizării înregistrărilor anti-POODLE” sau refuzul cifrurilor CBC în SSL 3.0.
    - *Google Chrome*: complet (TLS_FALLBACK_SCSV este implementat începând cu versiunea 33, alternativa la SSL 3.0 este dezactivată începând cu versiunea 39, SSL 3.0 în sine este dezactivată în mod implicit începând cu versiunea 40. Suportul pentru SSL 3.0 în sine a fost renunțat de la versiunea 44.)
    - *Mozilla Firefox*: complet (suportul pentru SSL 3.0 în sine este renunțat la versiunea 39. SSL 3.0 în sine este dezactivat în mod implicit, iar alternativa la SSL 3.0 este dezactivată începând cu versiunea 34, TLS_FALLBACK_SCSV este implementat începând cu versiunea 35. În ESR, SSL 3.0 este dezactivat singur implicit și TLS_FALLBACK_SCSV este implementat începând cu ESR 31.3.0.)
    - *Internet Explorer*: parțial (numai în versiunea 11, SSL 3.0 este dezactivat implicit din aprilie 2015. Versiunea 10 și mai veche sunt încă vulnerabile împotriva POODLE.)
    - *Opera:* completă (TLS_FALLBACK_SCSV este implementat începând cu versiunea 20, „divizarea înregistrărilor anti-POODLE”, care este eficientă numai cu implementarea pe partea clientului, este implementată începând cu versiunea 25, SSL 3.0 în sine este dezactivat implicit din versiunea 27. Suport pentru SSL 3.0 în sine va fi renunțat la versiunea 31.)
    - *Safari*: complet (numai pe OS X 10.8 și versiuni ulterioare și iOS 8, cifrurile CBC în timpul retragerii la SSL 3.0 sunt refuzate, dar asta înseamnă că va folosi RC4, ceea ce nu este

recomandat de asemenea. Suportul pentru SSL 3.0 în sine este renunțat la OS X 10.11 și versiuni ulterioare și iOS 9.)

- Atenuarea atacurilor RC4:
    - Google Chrome a dezactivat RC4, cu excepția cazului de rezervă începând cu versiunea 43. RC4 este dezactivat începând cu Chrome 48.
    - Firefox a dezactivat RC4, cu excepția cazului de rezervă începând cu versiunea 36. Firefox 44 a dezactivat RC4 în mod implicit.
    - Opera a dezactivat RC4, cu excepția cazului de rezervă începând cu versiunea 30. RC4 este dezactivat de la Opera 35.
    - Internet Explorer pentru Windows 7 /Server 2008 R2 și pentru Windows 8 /Server 2012 au setat prioritatea RC4 la cea mai mică și poate, de asemenea, să dezactiveze RC4, cu excepția cazului de rezervă prin setările de registry. Internet Explorer 11 Mobile 11 pentru Windows Phone 8.1 dezactivează RC4, cu excepția cazului în care nu funcționează niciun alt algoritm activat. Edge și IE 11 dezactivează complet RC4 în august 2016.
- Atenuare împotriva atacului FREAK:
    - Browserul Android inclus cu Android 4.0 și mai vechi este încă vulnerabil la atacul FREAK.
    - Internet Explorer 11 Mobile este încă vulnerabil la atacul FREAK.
    - Google Chrome, Internet Explorer (desktop), Safari (desktop și mobil) și Opera (mobil) au atenuări FREAK.
    - Mozilla Firefox pe toate platformele și Google Chrome pe Windows nu au fost afectate de FREAK.

Biblioteci

Majoritatea bibliotecilor de programare SSL și TLS sunt software gratuite și open-source.

- BoringSSL, un furk al OpenSSL pentru Chrome/Chromium și Android, precum și alte aplicații Google.
- Botan, o bibliotecă criptografică cu licență BSD scrisă în C++.
- BSAFE Micro Edition Suite: o implementare multiplatformă a TLS scrisă în C folosind un modul criptografic validat de FIPS
- BSAFE SSL-J: o bibliotecă TLS care oferă atât un API proprietar, cât și un API JSSE, folosind un modul criptografic validat de FIPS
- cryptlib: o bibliotecă portabilă de criptare open source (include implementarea TLS/SSL)
- Programatorii Delphi pot folosi o bibliotecă numită Indy care utilizează OpenSSL sau, alternativ, ICS care acceptă TLS 1.3 acum.
- GnuTLS: o implementare gratuită (licență LGPL)
- Java Secure Socket Extension (JSSE): API-ul Java și implementarea furnizorului (numit SunJSSE) [93]
- LibreSSL: un furk al OpenSSL prin proiectul OpenBSD.
- MatrixSSL: o implementare cu două licențe
- Mbed TLS (anterior PolarSSL): O implementare mică de bibliotecă SSL pentru dispozitive încorporate, care este concepută pentru ușurință în utilizare
- Servicii de securitate de rețea: bibliotecă open source validată FIPS 140
- OpenSSL: o implementare gratuită (licență BSD cu unele extensii)
- Schannel: o implementare a SSL și TLS Microsoft Windows ca parte a pachetului său.
- Transport securizat: o implementare a SSL și TLS utilizată în OS X și iOS ca parte a pachetelor lor.
- wolfSSL (anterior CyaSSL): Bibliotecă SSL/TLS încorporată cu un accent puternic pe viteză și dimensiune.

O lucrare prezentată la conferința ACM din 2012 privind securitatea computerelor și a comunicațiilor a arătat că multe aplicații au folosit incorect unele dintre aceste biblioteci SSL, ceea ce duce la vulnerabilități. Potrivit autorilor:
„Cauza principală a majorității acestor vulnerabilități este designul teribil al API-urilor pentru bibliotecile SSL subiacente. În loc să exprime proprietățile de securitate la nivel înalt ale tunelurilor de rețea, cum ar fi confidențialitatea și autentificarea, aceste API-uri expun detalii de nivel scăzut ale protocolului SSL. pentru dezvoltatorii de aplicații. În consecință, dezvoltatorii folosesc adesea incorect API-urile SSL, interpretând greșit și înțelegând greșit numeroșii parametri, opțiuni, efecte secundare și valorile returnate.”

Alte utilizări

Protocolul simplu de transfer de e-mail (SMTP) poate fi, de asemenea, protejat prin TLS. Aceste aplicații folosesc certificate de cheie publică pentru a verifica identitatea punctelor finale.

TLS poate fi folosit și pentru tunelarea unei întregi stive de rețea pentru a crea un VPN, ceea ce este cazul cu OpenVPN și OpenConnect. Mulți furnizori au căsătorit până acum capacitățile de criptare și autentificare ale TLS cu autorizare. A existat, de asemenea, o dezvoltare substanțială de la sfârșitul anilor 1990 în crearea tehnologiei client în afara browserelor Web, pentru a permite suportul pentru aplicațiile client/server. În comparație cu tehnologiile VPN IPsec tradiționale, TLS are câteva avantaje inerente în firewall și traversarea NAT, care fac mai ușor de administrat pentru populații mari cu acces la distanță.

TLS este, de asemenea, o metodă standard pentru protejarea semnalizării aplicației Session Initiation Protocol (SIP). TLS poate fi utilizat pentru furnizarea de autentificare și criptare a semnalizării SIP asociate cu VoIP și alte aplicații bazate pe SIP.

Securitate

Atacuri împotriva TLS/SSL

Atacurile semnificative împotriva TLS/SSL sunt enumerate mai jos.

În februarie 2015, IETF a emis un RFC informațional care rezumă diferitele atacuri cunoscute împotriva TLS/SSL.

Atacul de renegociere

O vulnerabilitate a procedurii de renegociere a fost descoperită în august 2009, care poate duce la atacuri de injectare de text simplu împotriva SSL 3.0 și a tuturor versiunilor actuale de TLS. De exemplu, permite unui atacator care poate deturna o conexiune https să-și insereze propriile cereri la începutul conversației pe care clientul o are cu serverul web. Atacatorul nu poate decripta comunicarea client-server, deci este diferită de un atac tipic de tip man-in-the-middle. O remediere pe termen scurt este ca serverele web să nu mai permită renegocierea, care de obicei nu va necesita alte modificări *decât dacă este utilizată autentificarea cu certificat de client*.

Pentru a remedia vulnerabilitatea, a fost propusă o extensie a indicației de renegociere pentru TLS. Va solicita clientului și serverului să includă și să verifice informații despre Handshake-uri anterioare în orice Handshake de renegociere. Această extensie a devenit un standard propus și i s-a atribuit numărul RFC 5746. RFC a fost implementat de mai multe biblioteci.

Atacurile de downgrade: atacul FREAK și Atacul Logjam

Un atac de downgrade a protocolului (numit și un atac de recuperare a versiunii) păcălește un server web să negocieze conexiuni cu versiunile anterioare de TLS (cum ar fi SSLv2) care au fost abandonate de mult timp ca nesigure.

Modificările anterioare aduse protocoalelor originale, cum ar fi **False Start** (adoptat și activat de Google Chrome) sau **Snap Start**, au introdus atacuri limitate de downgrade a protocolului TLS sau au permis modificări la lista suitelor de criptare trimisă de client la server. Procedând astfel, un atacator ar putea reuși să influențeze selecția suitei de criptare în încercarea de a downgrade suita de criptare negociată pentru a utiliza fie un algoritm de criptare simetric mai slab, fie un schimb de chei mai slab.

O lucrare prezentată la o conferință ACM despre securitatea computerelor și a comunicațiilor din 2012 a demonstrat că extensia False Start era în pericol: în anumite circumstanțe ar putea permite unui atacator să recupereze cheile de criptare offline și să acceseze datele criptate.

Atacurile de downgrade de criptare pot forța serverele și clienții să negocieze o conexiune folosind chei slabe din punct de vedere criptografic. În 2014, a fost descoperit un atac de tip „man-in-the-middle”, numit FREAK, care afectează stiva OpenSSL, browserul web implicit Android și unele browsere Safari. Atacul a implicat păcălirea serverelor pentru a negocia o conexiune TLS folosind chei de criptare de 512 biți slabe din punct de vedere criptografic.

Logjam este o exploatare de securitate descoperită în mai 2015, care exploatează opțiunea de utilizare a unor grupuri Diffie–Hellman de 512 biți „export-grade” care datează din anii 1990. Forțează serverele predispuse să treacă la downgrade la grupuri Diffie–Hellman de 512 biți slabe din punct de vedere criptografic. Un atacator poate deduce apoi cheile pe care clientul și serverul le determină folosind schimbul de chei Diffie-Hellman.

Atacuri încrucișate: DROWN

Atacul DROWN este un exploit care atacă serverele care acceptă suitele de protocoale SSL/TLS contemporane prin exploatarea suportului acestora pentru protocolul SSLv2 învechit, nesigur, pentru a exploata un atac asupra conexiunilor folosind protocoale actualizate care altfel ar fi sigure. DROWN exploatează o vulnerabilitate în protocoalele utilizate și configurația serverului, mai degrabă decât orice eroare specifică de implementare. Detaliile complete despre DROWN au fost anunțate în martie 2016, împreună cu un patch pentru

exploit. La acel moment, peste 81.000 dintre cele mai populare 1 milion de site-uri web se numărau printre site-urile web protejate TLS care erau vulnerabile la atacul DROWN.

Atacul BEAST

Pe 23 septembrie 2011, cercetătorii Thai Duong și Juliano Rizzo au demonstrat o dovadă a conceptului numită **BEAST** (**Browser Exploit Against SSL/TLS**) folosind un applet Java pentru a încălca aceleași constrângeri de politică de origine, pentru o înlănțuire de blocuri de cifrare foarte cunoscută (CBC) vulnerabilitate în TLS 1.0: un atacator care observă 2 blocuri de text cifrat consecutive C0, C1 poate testa dacă blocul de text clar P1 este egal cu x prin alegerea următorului bloc de text clar P2 = x ⊕ C0 ⊕ C1 ; conform operației CBC, C2 = E(C1 ⊕ P2) = E(C1 ⊕ x ⊕ C0 ⊕ C1) = E(C0 ⊕ x), care va fi egal cu C1 dacă x = P1. Exploatările practice nu au fost demonstrate anterior pentru această vulnerabilitate, care a fost descoperită inițial de Phillip Rogaway în 2002. Vulnerabilitatea atacului fusese remediată cu TLS 1.1 în 2006, dar TLS 1.1 nu a fost adoptat pe scară largă înainte de acest atac. demonstrație.

RC4 ca cifru de flux este imun la atacul BEAST. Prin urmare, RC4 a fost utilizat pe scară largă ca o modalitate de a atenua atacul BEAST din partea serverului. Cu toate acestea, în 2013, cercetătorii au descoperit mai multe puncte slabe în RC4. După aceea, activarea RC4 pe partea serverului nu a mai fost recomandată.

Chrome și Firefox în sine nu sunt vulnerabile la atacurile BEAST, cu toate acestea, Mozilla și-a actualizat bibliotecile NSS pentru a atenua atacurile de tipul BEAST. NSS este folosit de Mozilla Firefox și Google Chrome pentru a implementa SSL. Ca urmare, unele servere web care au o implementare întrespartă a specificației SSL pot înceta să funcționeze.

Microsoft a lansat Buletinul de securitate MS12-006 pe 10 ianuarie 2012, care a remediat vulnerabilitatea BEAST prin schimbarea modului în care componenta Windows Secure Channel (Schannel) transmite pachete de rețea criptate de la capătul serverului. Utilizatorii Internet Explorer (înainte de versiunea 11) care rulează pe versiuni mai vechi de Windows (Windows 7, Windows 8 și Windows Server 2008 R2) pot restricționa utilizarea TLS la 1.1 sau o versiune ulterioară.

Apple a remediat vulnerabilitatea BEAST prin implementarea 1/n-1 split și activând-o implicit în OS X Mavericks, lansat pe 22 octombrie 2013.

Atacurile CRIME și BREACH

Autorii atacului BEAST sunt, de asemenea, creatorii atacului de mai târziu CRIME, care poate permite unui atacator să recupereze conținutul cookie-urilor web atunci când compresia datelor este utilizată împreună cu TLS. Când este utilizat pentru a recupera conținutul cookie-urilor de autentificare secretă, permite unui atacator să efectueze deturnarea sesiunii pe o sesiune web autentificată.

În timp ce atacul CRIME a fost prezentat ca un atac general care ar putea funcționa eficient împotriva unui număr mare de protocoale, inclusiv, dar fără a se limita la, TLS și protocoale de nivel de aplicație, cum ar fi SPDY sau HTTP, doar exploatările împotriva TLS și SPDY au fost demonstrate și în mare măsură atenuate. în browsere și servere. Exploatarea CRIME împotriva compresiei HTTP nu a fost deloc atenuată, chiar dacă autorii CRIME au avertizat că această vulnerabilitate ar putea fi chiar mai răspândită decât compresia SPDY și TLS combinate. În 2013, a fost anunțată o nouă instanță a atacului CRIME împotriva compresiei HTTP, denumită BREACH. Pe baza atacului CRIME, un atac BREACH poate extrage jetoane de conectare, adrese de e-mail sau alte informații sensibile din traficul web criptat TLS în doar 30 de secunde (în funcție de numărul de octeți care trebuie extrași), cu condiția ca atacatorul să păcălească victima să viziteze un link web rău intenționat sau sa fie capabil să injecteze conținut în paginile valide pe care le vizitează utilizatorul (ex: o rețea fără fir aflată sub controlul atacatorului). Toate versiunile de TLS și SSL sunt expuse riscului, indiferent de algoritmul de criptare sau cifrul utilizat.

Spre deosebire de cazurile anterioare de CRIME, care pot fi apărate cu succes prin dezactivarea compresiei TLS sau a compresiei antetului SPDY, BREACH exploatează compresia HTTP care nu poate fi dezactivată în mod realist, deoarece aproape toate serverele web se bazează pe aceasta pentru a îmbunătăți vitezele de transmisie a datelor pentru utilizatorii. Aceasta este o limitare cunoscută a TLS, deoarece este susceptibil la atacuri alese în text clar împotriva datelor din stratul de aplicație pe care trebuia să le protejeze.

Atacurile de cronometrare la padding

Versiunile TLS anterioare erau vulnerabile împotriva atacului Padding Oracle descoperit în 2002. O variantă nouă, numită atacul Lucky Thirteen, a fost publicată în 2013.

Unii experți au recomandat, de asemenea, evitarea Triple DES CBC. Deoarece ultimele cifruri acceptate dezvoltate pentru a susține orice program care utilizează biblioteca SSL/TLS a Windows XP, cum ar fi Internet Explorer pe Windows XP, sunt RC4 și Triple-DES, iar din moment ce RC4 este acum depreciat (vezi discuția despre atacurile RC4), acest lucru face dificilă suportul pentru orice versiune de SSL pentru orice program care utilizează această bibliotecă pe XP.

A fost lansată o remediere ca extensia Encrypt-then-MAC la specificația TLS, lansată ca RFC 7366. Atacul Lucky Thirteen poate fi atenuat în TLS 1.2 folosind doar cifrurile AES_GCM; AES_CBC rămâne vulnerabil. SSL poate

proteja e-mailul, VoIP și alte tipuri de comunicații prin rețele nesigure, în plus față de cazul de utilizare principal al transmiterii securizate de date între un client și server.

Atacul POODLE

Pe 14 octombrie 2014, cercetătorii Google au publicat o vulnerabilitate în designul SSL 3.0, care face ca modul de operare CBC cu SSL 3.0 să fie vulnerabil la un Padding attack (CVE - 2014-3566). Acest atac l-au numit **POODLE** (**Padding Oracle On Downgraded Legacy Encryption**). În medie, atacatorii trebuie să facă doar 256 de solicitări SSL 3.0 pentru a dezvălui un octet de mesaje criptate.

Deși această vulnerabilitate există doar în SSL 3.0 și majoritatea clienților și serverelor acceptă TLS 1.0 și versiuni ulterioare, toate browserele majore trec voluntar la SSL 3.0 dacă Handshake cu versiuni mai noi de TLS eșuează, cu excepția cazului în care oferă opțiunea unui utilizator sau administrator de a dezactiva SSL 3.0. iar utilizatorul sau administratorul face acest lucru. Prin urmare, omul din mijloc poate mai întâi să efectueze un atac de retragere a versiunii și apoi să exploateze această vulnerabilitate.

Pe 8 decembrie 2014, a fost anunțată o variantă de POODLE care are impact asupra implementărilor TLS care nu impun în mod corespunzător cerințele de octeți de completare.

Atacurile RC4

În ciuda existenței unor atacuri asupra RC4 care i-au spart securitatea, suitele de criptare în SSL și TLS care erau bazate pe RC4 erau încă considerate sigure înainte de 2013, pe baza modului în care erau utilizate în SSL și TLS. În 2011, suita RC4 a fost de fapt recomandată ca o soluție pentru atacul BEAST. Noile forme de atac dezvăluite în martie 2013 au demonstrat în mod concludent fezabilitatea ruperii RC4 în TLS, sugerând că nu a fost o soluție bună pentru BEAST. Un scenariu de atac a fost propus de către AlFardan, Bernstein, Paterson, Poettering și Schuldt, care a folosit părtiniri statistice recent descoperite în tabelul de chei RC4 pentru a recupera părți ale textului simplu cu un număr mare de criptări TLS. Un atac asupra RC4 în TLS și SSL care necesită criptări 13×2^{20} pentru a sparge RC4 a fost dezvăluit pe 8 iulie 2013 și mai târziu descris ca „fezabil” în prezentarea însoțitoare la un simpozion de securitate USENIX în august 2013. În iulie 2015, îmbunătățirile ulterioare ale atacului fac din ce în ce mai practic să învingă securitatea TLS criptat cu RC4.

Deoarece multe browsere moderne au fost concepute pentru a învinge atacurile BEAST (cu excepția Safari pentru Mac OS X 10.7 sau o versiune anterioară, pentru iOS 6 sau o versiune anterioară și pentru Windows), RC4 nu mai este o alegere bună pentru TLS 1.0. Cifrurile CBC care au fost afectate de atacul BEAST în trecut au devenit o alegere mai populară pentru protecție. Mozilla și Microsoft recomandă dezactivarea RC4 acolo unde este posibil. RFC 7465 interzice utilizarea suitelor de criptare RC4 în toate versiunile de TLS.

La 1 septembrie 2015, Microsoft, Google și Mozilla au anunțat că suitele de criptare RC4 vor fi dezactivate implicit în browserele lor (Microsoft Edge, Internet Explorer 11 pe Windows 7/8.1/10, Firefox și Chrome) la începutul lui 2016.

Atacul de trunchiere

Un atac de trunchiere TLS (deconectare) blochează solicitările de deconectare ale contului unei victime, astfel încât utilizatorul să rămână conectat fără să știe la un serviciu web. Când este trimisă cererea de deconectare, atacatorul injectează un mesaj TCP FIN necriptat (nu mai sunt date de la expeditor) pentru a închide conexiunea. Prin urmare, serverul nu primește cererea de deconectare și nu este conștient de terminarea anormală.

Publicat în iulie 2013, atacul face ca servicii web precum Gmail și Hotmail să afișeze o pagină care informează utilizatorul că s-a deconectat cu succes, asigurând în același timp că browserul utilizatorului păstrează autorizarea cu serviciul, permițând un atacator cu acces ulterior la browser pentru a accesa și a prelua controlul asupra contului conectat al utilizatorului. Atacul nu se bazează pe instalarea de programe malware pe computerul victimei; atacatorii trebuie doar să se plaseze între victimă și serverul web (de exemplu, prin configurarea unui hotspot fără fir). Această vulnerabilitate necesită și accesul la computerul victimei. O altă posibilitate este că atunci când se utilizează FTP, conexiunea de date poate avea un FIN fals în fluxul de date, iar dacă regulile de protocol pentru schimbul de alerte close_notify nu sunt respectate la un fișier poate fi trunchiat.

Atac în text simplu împotriva DTLS

În februarie 2013, doi cercetători de la Royal Holloway, Universitatea din Londra au descoperit un atac de sincronizare care le-a permis să recupereze (părți din) text simplu dintr-o conexiune DTLS utilizând implementarea OpenSSL sau GnuTLS a DTLS atunci când a fost utilizată criptarea modului Cipher Block Chaining.

Atacul Unholy PAC

Acest atac, descoperit la jumătatea anului 2016, exploatează punctele slabe ale protocolului Web Proxy Autodiscovery (WPAD) pentru a expune adresa URL la care un utilizator web încearcă să o acceseze printr-un link web activat pentru TLS. Dezvăluirea unei adrese URL poate încălca confidențialitatea unui utilizator, nu numai din cauza site-ului web accesat, ci și pentru că adresele URL sunt uneori folosite pentru autentificarea utilizatorilor. Serviciile de partajare a documentelor, cum ar fi cele oferite de Google și Dropbox, funcționează și

trimițând unui utilizator un simbol de securitate care este inclus în adresa URL. Un atacator care obține astfel de adrese URL poate obține acces deplin la contul sau datele unei victime.

Exploit-ul funcționează împotriva aproape tuturor browserelor și sistemelor de operare.

Atacul Sweet32

Atacul Sweet32 sparge toate cifrurile bloc pe 64 de biți utilizate în modul CBC așa cum sunt utilizate în TLS, exploatând un atac de naștere și fie un atac de tip man-in-the-middle, fie injectarea unui JavaScript rău intenționat într-o pagină web. Scopul atacului „man-in-the-middle” sau al injecției JavaScript este de a permite atacatorului să capteze suficient trafic pentru a realiza un atac de naștere.

Erori de implementare: Heartbleed bug, attacul BERserk, Cloudflare bug

Bug Heartbleed este o vulnerabilitate gravă specifică implementării SSL/TLS în bibliotecă de software criptografic OpenSSL, care afectează versiunile 1.0.1 până la 1.0.1f. Această slăbiciune, raportată în aprilie 2014, permite atacatorilor să fure chei private de la servere care ar trebui în mod normal protejate. Bug-ul Heartbleed permite oricui de pe Internet să citească memoria sistemelor protejate de versiunile vulnerabile ale software-ului OpenSSL. Acest lucru compromite cheile private secrete asociate cu certificatele publice utilizate pentru a identifica furnizorii de servicii și pentru a cripta traficul, numele și parolele utilizatorilor și conținutul real. Acest lucru permite atacatorilor să asculte cu urechea comunicațiilor, să fure date direct de la servicii și utilizatori și să uzurpare identitatea serviciilor și utilizatorilor. Vulnerabilitatea este cauzată de o eroare de supra-citire a memoriei tampon în software-ul OpenSSL, mai degrabă decât de un defect în specificația protocolului SSL sau TLS.

În septembrie 2014, o variantă a vulnerabilității PKCS#1 v1.5 RSA Signature Forgery a lui Daniel Bleichenbacher a fost anunțată de Intel Security Advanced Threat Research. Acest atac, numit BERserk, este rezultatul decodării incomplete a lungimii ASN.1 a semnăturilor cheii publice în unele implementări SSL și permite un atac de tip man-in-the-middle prin falsificarea unei semnături de cheie publică.

În februarie 2015, după ce mass-media a raportat preinstalarea ascunsă a adware-ului Superfish pe unele notebook-uri Lenovo, un cercetător a descoperit că un certificat rădăcină de încredere pe mașinile Lenovo afectate nu este sigur, deoarece cheile puteau fi accesate cu ușurință folosind numele companiei, Komodia, ca expresie de acces. Biblioteca Komodia a fost concepută pentru a intercepta traficul TLS/SSL la nivelul clientului pentru control parental și supraveghere, dar a fost folosită și în numeroase programe adware, inclusiv Superfish, care erau adesea instalate pe ascuns fără ca utilizatorul de computer să știe. La rândul lor, aceste programe potențial nedorite au instalat certificatul rădăcină cospart, permițând atacatorilor să controleze complet traficul web și să confirme site-urile web false ca fiind autentice.

În mai 2016, a fost raportat că zeci de site-uri web daneze protejate prin HTTPS aparținând Visa Inc. erau vulnerabile la atacuri care le permit hackerilor să injecteze cod rău intenționat și conținut falsificat în browserele vizitatorilor. Atacurile au funcționat deoarece implementarea TLS utilizată pe serverele afectate a reutilizat incorect numere aleatorii (nonces) care sunt destinate să fie utilizate o singură dată, asigurându-se că fiecare Handshake TLS este unică.

În februarie 2017, o eroare de implementare cauzată de un singur caracter greșit în codul folosit pentru a analiza HTML a creat o eroare de depășire a memoriei tampon pe serverele Cloudflare. Asemănătoare ca efecte cu eroarea Heartbleed descoperită în 2014, această eroare de overflow, cunoscută pe scară largă ca Cloudbleed, a permis terților neautorizați să citească date din memoria programelor care rulează pe servere - date care altfel ar fi trebuit protejate de TLS.

Sondaj: site-uri web vulnerabile la atacuri

Începând cu iulie 2021, Trustworthy Internet Movement a estimat proporția de site-uri web care sunt vulnerabile la atacurile TLS.

Sondajul vulnerabilităților TLS ale celor mai populare site-uri web

Atacurile	Securitate			
	Nesigur	Depinde	Sigur	Alte
Atacul de renegociere	< 0,1% susțin renegocierea nesigură	< 0,1% le susțin pe ambele	99,4% susțin	0,6% fără suport

			renegocierea securizată	
Atacurile RC4	0,2% acceptă suitele RC4 utilizate cu browserele moderne	3,6% acceptă unele suite RC4	96,2% fără suport	—
Compresie TLS (atac CRIME)	0% vulnerabil	—	—	—
Heartbleed	0% vulnerabil	—	—	—
Atacul de injecție ChangeCipherSpec	< 0,1% vulnerabil și exploatabil	< 0,1% vulnerabil, neexploatabil	99,3% nu sunt vulnerabile	0,7% necunoscut
Atacul POODLE împotriva TLS **(POODLE original împotriva SSL 3.0 nu este inclus)**	< 0,1% vulnerabil și exploatabil	—	99,8% nu sunt vulnerabile	0,1% necunoscut
Degradarea protocolului	4,1% Apărarea downgrade nu este acceptată	—	77,8% Apărare downgrade susținută	18,1% necunoscut

Forward secrecy

Forward secrecy este o proprietate a sistemelor criptografice care asigură că o cheie de sesiune derivată dintr-un set de chei publice și private nu va fi compromisă dacă una dintre cheile private este compromisă în viitor. Fără Forward secrecy, dacă cheia privată a serverului este compromisă, nu numai că toate sesiunile viitoare criptate TLS care utilizează acel certificat de server vor fi compromise, ci și orice sesiuni anterioare care au folosit-o (cu condiția ca aceste sesiuni anterioare să fi fost interceptate și stocate la momentul transmiterii). O implementare a TLS poate oferi Forward secrecy prin necesitatea folosirii schimbului de chei Diffie-Hellman efemer pentru a stabili cheile de sesiune, iar unele implementări notabile TLS fac acest lucru exclusiv: de exemplu, Gmail și alte servicii Google HTTPS care utilizează OpenSSL. Cu toate acestea, mulți clienți și servere care acceptă TLS (inclusiv browsere și servere web) nu sunt configurate pentru a implementa astfel de restricții. În practică, cu excepția cazului în care un serviciu web folosește schimbul de chei Diffie-Hellman pentru a implementa Forward secrecy, tot traficul web criptat către și de la acel serviciu poate fi decriptat de o terță parte dacă obține masterul serverului (privat) cheie; de exemplu, printr-o hotărâre judecătorească.

Chiar și acolo unde este implementat schimbul de chei Diffie-Hellman, mecanismele de gestionare a sesiunilor de la server pot afecta Forward secrecy. Utilizarea tichetelor de sesiune TLS (o extensie TLS) face ca sesiunea să fie protejată de AES128-CBC-SHA256, indiferent de orice alți parametri TLS negociați, inclusiv suitele de criptare de secretizare directă, iar cheile de tichete de sesiune TLS cu viață lungă înfrâng încercarea de implementare. Forward secrecy. Cercetările de la Universitatea Stanford din 2014 au constatat, de asemenea, că din 473.802 de servere TLS chestionate, 82,9% dintre serverele care implementau schimbul de chei Diffie–Hellman (DHE) efemer pentru a susține Forward secrecy foloseau parametri Diffie–Hellman slabi. Aceste alegeri slabe ale parametrilor ar putea compromite eficiența secretului direct pe care serverele au încercat să o ofere.

De la sfârșitul anului 2011, Google a furnizat în mod prestabilit confidențialitate cu TLS utilizatorilor serviciului său Gmail, împreună cu Google Docs și căutare criptată, printre alte servicii. Din noiembrie 2013, Twitter a furnizat Forward secrecy cu TLS utilizatorilor serviciului său. [164] Începând cu august 2019, aproximativ 80% dintre

site-urile web compatibile cu TLS sunt configurate să utilizeze suite de criptare care oferă secret de transmitere către majoritatea browserelor web.

Interceptia TLS

Interceptarea TLS (sau interceptarea HTTPS, dacă se aplică în special protocolului respectiv) este practica de a intercepta un flux de date criptate pentru a-l decripta, a-l citi și, eventual, a-l manipula, apoi a-l recripta și a trimite din nou datele pe drum. Acest lucru se face prin intermediul unui „ proxy transparent”: software-ul de interceptare încheie conexiunea TLS de intrare, inspectează textul simplu HTTP și apoi creează o nouă conexiune TLS la destinație.

Interceptarea TLS/HTTPS este utilizată ca măsură de securitate a informațiilor de către operatorii de rețea pentru a putea scana și a proteja împotriva intruziunii de conținut rău intenționat în rețea, cum ar fi virușii informatici și alte programe malware. Altfel, un astfel de conținut nu ar putea fi detectat atâta timp cât este protejat prin criptare, ceea ce se întâmplă din ce în ce mai mult ca urmare a utilizării de rutină a HTTPS și a altor protocoale securizate.

Un dezavantaj semnificativ al interceptării TLS/HTTPS este că introduce noi riscuri de securitate proprii. O limitare notabilă este că oferă un punct în care traficul de rețea este disponibil necriptat, oferind astfel atacatorilor un stimulent să atace acest punct în special pentru a obține acces la conținut altfel sigur. Interceptarea permite, de asemenea, operatorului de rețea sau persoanelor care au acces la sistemul său de interceptare, să efectueze atacuri de tip om-in-the-middle împotriva utilizatorilor rețelei. Un studiu din 2017 a constatat că „interceptarea HTTPS a devenit uimitor de răspândită și că produsele de interceptare ca clasă au un impact dramatic negativ asupra securității conexiunii”.

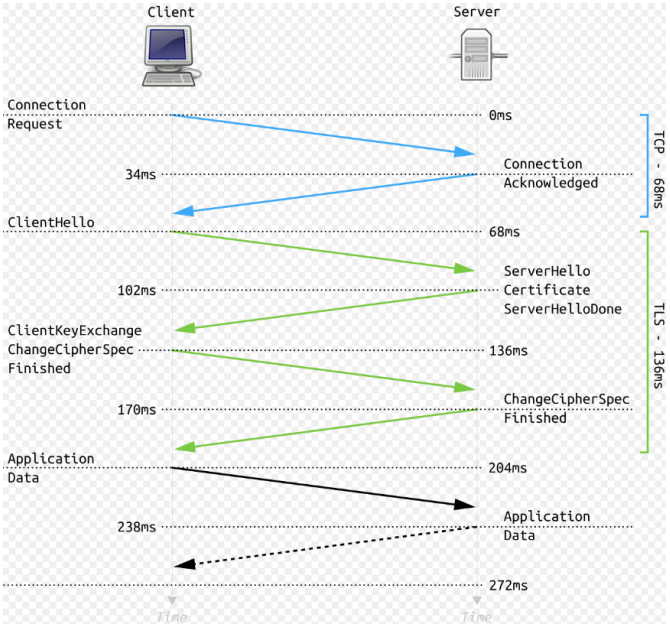
Detalii protocol

Protocolul TLS face schimb *de înregistrări*, care încapsulează datele care urmează să fie schimbate într-un format specific (vezi mai jos). Fiecare înregistrare poate fi comprimată, completată, atașată cu un cod de autentificare a mesajelor (MAC) sau criptată, totul în funcție de starea conexiunii. Fiecare înregistrare are un câmp *de tip de conținut* care desemnează tipul de date încapsulate, un câmp de lungime și un câmp de versiune TLS. Datele încapsulate pot fi mesaje de control sau procedurale ale TLS în sine, sau pur și simplu datele aplicației necesare pentru a fi transferate de TLS. Specificațiile (suită de cifrare, chei etc.) necesare pentru schimbul de date aplicației prin TLS, sunt convenite în „ Handshake-ul TLS” între clientul care solicită datele și serverul care răspunde la solicitări. Prin urmare, protocolul definește atât structura sarcinilor utile transferate în TLS, cât și procedura de stabilire și monitorizare a transferului.

Handshake TLS

Când începe conexiunea, înregistrarea încapsulează un protocol de „control” – protocolul de mesagerie de Handshake (*tip de conținut* 22). Acest protocol este utilizat pentru a face schimb de toate informațiile necesare de ambele părți pentru schimbul datelor aplicației reale de către TLS. Acesta definește formatul mesajelor și ordinea schimbului lor. Acestea pot varia în funcție de cerințele clientului și serverului – adică, există mai multe proceduri posibile pentru a configura conexiunea. Acest schimb inițial are ca rezultat o conexiune TLS reușită (ambele părți gata să transfere datele aplicației cu TLS) sau un mesaj de alertă (după cum este specificat mai jos).

Handshake TLS de bază



Urmează un exemplu tipic de conexiune, ilustrând o Handshake în care serverul (dar nu clientul) este autentificat prin certificatul său:

1. Faza de negociere:
    - Un client trimite un mesaj **ClientHello** specificând cea mai înaltă versiune de protocol TLS pe care o acceptă, un număr aleatoriu, o listă de suite de criptare sugerate și metode de compresie sugerate. Dacă clientul încearcă să efectueze o Handshake reluată, poate trimite un *ID de sesiune*. Dacă clientul poate utiliza Application-Layer Protocol Negotiation, acesta poate include o listă de protocoale de aplicație acceptate, cum ar fi HTTP/2.
    - Serverul răspunde cu un mesaj **ServerHello**, care conține versiunea de protocol aleasă, un număr aleatoriu, suită de criptare și metoda de compresie din opțiunile oferite de client. Pentru a confirma sau permite reluarea Handshake, serverul poate trimite un *ID de sesiune*. Versiunea de protocol aleasă ar trebui să fie cea mai mare pe care o acceptă atât clientul, cât și serverul. De exemplu, dacă clientul acceptă TLS versiunea 1.1 și serverul acceptă versiunea 1.2, trebuie selectată versiunea 1.1; versiunea 1.2 nu trebuie selectată.
    - Serverul trimite mesajul **certificat** (în funcție de suita de criptare selectată, acesta poate fi omis de server).
    - Serverul trimite mesajul **ServerKeyExchange** (în funcție de suita de criptare selectată, acesta poate fi omis de server). Acest mesaj este trimis pentru toate suitele de criptare DHE, ECDHE și DH_anon.
    - Serverul trimite un mesaj **ServerHelloDone**, indicând că s-a încheiat cu negocierea Handshake.
    - Clientul răspunde cu un mesaj **ClientKeyExchange**, care poate conține un *PreMasterSecret*, cheie publică sau nimic. (Din nou, aceasta depinde de cifrul selectat.) Acest *PreMasterSecret* este criptat folosind cheia publică a certificatului de server.
    - Clientul și serverul folosesc apoi numerele aleatoare și *PreMasterSecret* pentru a calcula un secret comun, numit „secretul principal”. Toate celelalte date cheie (chei de sesiune, cum ar fi IV, cheie de criptare simetrică, cheie MAC) pentru această conexiune sunt derivate din acest secret principal (și valorile aleatoare generate de client și server), care este trecut printr-un sistem atent proiectat. funcție pseudoaleatorie.
2. Clientul trimite acum o înregistrare **ChangeCipherSpec**, spunându-i în esență serverului: „Tot ceea ce vă spun de acum înainte va fi autentificat (și criptat dacă parametrii de criptare au fost prezenți în certificatul serverului).” ChangeCipherSpec este în sine un protocol la nivel de înregistrare cu tip de conținut de 20.
    - Clientul trimite un mesaj **Terminat** autentificat și criptat, care conține un hash și MAC peste mesajele anterioare de Handshake.
    - *Serverul va încerca să decripteze mesajul Terminat* al clientului și să verifice hash-ul și MAC. Dacă decriptarea sau verificarea eșuează, se consideră că Handshake-ul a eșuat și conexiunea ar trebui încheiată.
3. În cele din urmă, serverul trimite un **ChangeCipherSpec**, spunându-i clientului: „Tot ceea ce vă spun de acum înainte va fi autentificat (și criptat, dacă criptarea a fost negociată).”
    - Serverul trimite mesajul **Terminat** autentificat și criptat.
    - Clientul efectuează aceeași procedură de decriptare și verificare ca și serverul la pasul anterior.
4. Faza de aplicare: în acest moment, „ Handshake-ul” este completă și protocolul de aplicație este activat, cu tipul de conținut 23. Mesajele de aplicație schimbate între client și server vor fi, de asemenea, autentificate și opțional criptate exact ca în mesajul lor *Terminat*. În caz contrar, tipul de conținut va returna 25 și clientul nu se va autentifica.

Handshake TLS autentificată de client

Următorul exemplu *complet* arată un client care este autentificat (în plus față de server, ca în exemplul de mai sus; vezi autentificarea reciprocă) prin TLS folosind certificate schimbate între ambii colegi.

1. Faza de negociere:
    - Un client trimite un mesaj **ClientHello** specificând cea mai înaltă versiune de protocol TLS pe care o acceptă, un număr aleatoriu, o listă de suite de criptare sugerate și metode de compresie.

- Serverul răspunde cu un mesaj **ServerHello**, care conține versiunea de protocol aleasă, un număr aleatoriu, suită de criptare și metoda de compresie din opțiunile oferite de client. Serverul poate trimite, de asemenea, un *ID de sesiune* ca parte a mesajului pentru a efectua o Handshake reluată.
- Serverul trimite mesajul **certificat** (în funcție de suita de criptare selectată, acesta poate fi omis de server)
- Serverul trimite mesajul **ServerKeyExchange** (în funcție de suita de criptare selectată, acesta poate fi omis de server). Acest mesaj este trimis pentru toate suitele de criptare DHE, ECDHE și DH_anon. [1]
- Serverul trimite un mesaj **CertificateRequest**, pentru a solicita un certificat de la client.
- Serverul trimite un mesaj **ServerHelloDone**, indicând că s-a încheiat cu negocierea Handshake.
- Clientul răspunde cu un mesaj **Certificat**, care conține certificatul clientului, dar nu cheia privată a acestuia.
- Clientul trimite un mesaj **ClientKeyExchange**, care poate conține un *PreMasterSecret*, cheie publică sau nimic. (Din nou, aceasta depinde de cifrul selectat.) Acest *PreMasterSecret* este criptat folosind cheia publică a certificatului de server.
- Clientul trimite un mesaj **CertificateVerify**, care este o semnătură peste mesajele anterioare de Handshake folosind cheia privată a certificatului clientului. Această semnătură poate fi verificată utilizând cheia publică a certificatului clientului. Acest lucru permite serverului să știe că clientul are acces la cheia privată a certificatului și, astfel, deține certificatul.
- Clientul și serverul folosesc apoi numerele aleatoare și *PreMasterSecret* pentru a calcula un secret comun, numit „secretul principal”. Toate celelalte date cheie („cheile de sesiune”) pentru această conexiune sunt derivate din acest secret principal (și din valorile aleatoare generate de client și server), care sunt transmise printr-o funcție pseudoaleatoare atent proiectată.

2. Clientul trimite acum o înregistrare **ChangeCipherSpec**, spunând în esență serverului: „Tot ceea ce vă spun de acum înainte va fi autentificat (și criptat dacă criptarea a fost negociată). „ChangeCipherSpec este în sine un protocol la nivel de înregistrare și are tipul 20 și nu 22.
    - În cele din urmă, clientul trimite un mesaj criptat **Finished**, care conține un hash și MAC peste mesajele anterioare de handshake.
    - *Serverul va încerca să decripteze mesajul Terminat* al clientului și să verifice hash-ul și MAC. Dacă decriptarea sau verificarea eșuează, se consideră că Handshake-ul a eșuat și conexiunea trebuie întrespartă.
3. În cele din urmă, serverul trimite un **ChangeCipherSpec**, spunându-i clientului: „Tot ceea ce vă spun de acum înainte va fi autentificat (și criptat dacă criptarea a fost negociată).”
    - Serverul trimite propriul mesaj criptat **Terminat.**
    - Clientul efectuează aceeași procedură de decriptare și verificare ca și serverul la pasul anterior.
4. Faza de aplicare: în acest moment, „ Handshake-ul” este completă și protocolul de aplicație este activat, cu tipul de conținut 23. Mesajele de aplicație schimbate între client și server vor fi, de asemenea, criptate exact ca în mesajul lor *Terminat*.

Handshake TLS reluată

Operațiunile cu cheie publică (de exemplu, RSA) sunt relativ costisitoare în ceea ce privește puterea de calcul. TLS oferă o comandă rapidă securizată în mecanismul de Handshake pentru a evita aceste operațiuni: sesiuni reluate. Sesiunile reluate sunt implementate folosind ID-uri de sesiune sau bilete de sesiune.

Pe lângă beneficiul de performanță, sesiunile reluate pot fi folosite și pentru conectare unică, deoarece garantează că atât sesiunea inițială, cât și orice sesiune reluată provin de la același client. Acest lucru este de o importanță deosebită pentru protocolul FTP peste TLS/SSL, care altfel ar suferi de un atac de tip man-in-the-middle în care un atacator ar putea intercepta conținutul conexiunilor de date secundare.

Handshake TLS 1.3

Handshake-ul TLS 1.3 a fost condensată la o singură călătorie dus-întors în comparație cu cele două călătorii dus-întors necesare în versiunile anterioare de TLS/SSL.

Pentru a începe Handshake-ul, clientul ghicește ce algoritm de schimb de chei va fi selectat de server și trimite un mesaj **ClientHello** către server, care conține o listă de cifruri acceptate (în ordinea preferințelor clientului) și chei publice pentru o parte sau toată cheia sa. face schimb de presupuneri. Dacă clientul ghicește cu succes algoritmul de schimb de chei, 1 călătorie dus-întors este eliminată din Handshake. După ce primește **ClientHello**,

serverul selectează un cifr și trimite înapoi un **ServerHello** cu propria sa cheie publică, urmat de **certificatul** serverului și mesajele **Terminate.**

După ce clientul primește mesajul final al serverului, acesta este acum coordonat cu serverul pe care suită de criptare să folosească.

ID-urile sesiunii [modificați]

Într-o Handshake obișnuită, serverul trimite un *ID de sesiune* ca parte a mesajului **ServerHello**. Clientul asociază acest *ID de sesiune* cu adresa IP a serverului și cu portul TCP, astfel încât atunci când clientul se conectează din nou la acel server, să poată folosi *ID-ul sesiunii* pentru a scurta Handshake-ul. În server, *id-ul de sesiune* se mapează la parametrii criptografici negociați anterior, în special „secretul principal". Ambele părți trebuie să aibă același „secret principal" sau Handshake-ul reluată va eșua (acest lucru împiedică un interceptator să folosească un *ID de sesiune*). Datele aleatorii din mesajele **ClientHello** și **ServerHello** garantează practic că cheile de conexiune generate vor fi diferite de cele din conexiunea anterioară. În RFC-uri, acest tip de Handshake se numește Handshake *prescurtată*. De asemenea, este descris în literatură ca o Handshake *de repornire*.

1. Faza de negociere:
    - Un client trimite un mesaj **ClientHello** specificând cea mai înaltă versiune de protocol TLS pe care o acceptă, un număr aleatoriu, o listă de suite de criptare sugerate și metode de compresie. În mesaj este inclus *ID-ul sesiunii* de la conexiunea TLS anterioară.
    - Serverul răspunde cu un mesaj **ServerHello**, care conține versiunea de protocol aleasă, un număr aleatoriu, suită de criptare și metoda de compresie din opțiunile oferite de client. Dacă serverul recunoaște *ID-ul de sesiune* trimis de client, acesta răspunde cu același *ID de sesiune*. Clientul folosește acest lucru pentru a recunoaște că se efectuează o Handshake reluată. Dacă serverul nu recunoaște *ID-ul de sesiune* trimis de client, trimite o valoare diferită pentru *ID-ul de sesiune*. Acest lucru îi spune clientului că o Handshake reluată nu va fi efectuată. În acest moment, atât clientul, cât și serverul au „secretul principal" și date aleatorii pentru a genera datele cheie care vor fi utilizate pentru această conexiune.
2. Serverul trimite acum o înregistrare **ChangeCipherSpec**, spunându-i în esență clientului: „Tot ceea ce vă spun de acum înainte va fi criptat". ChangeCipherSpec este în sine un protocol la nivel de înregistrare și are tipul 20 și nu 22.
    - În cele din urmă, serverul trimite un mesaj criptat **Finished**, care conține un hash și MAC peste mesajele anterioare de handshake.
    - Clientul va încerca să decripteze mesajul *Terminat* al serverului și să verifice hash-ul și MAC. Dacă decriptarea sau verificarea eșuează, se consideră că Handshake-ul a eșuat și conexiunea trebuie întrespartă.
3. În cele din urmă, clientul trimite un **ChangeCipherSpec**, spunându-i serverului: „Tot ceea ce vă spun de acum înainte va fi criptat".
    - Clientul trimite propriul mesaj criptat **Terminat.**
    - Serverul efectuează aceeași procedură de decriptare și verificare ca și clientul la pasul anterior.
4. Faza de aplicare: în acest moment, „ Handshake-ul" este completă și protocolul de aplicație este activat, cu tipul de conținut 23. Mesajele de aplicație schimbate între client și server vor fi, de asemenea, criptate exact ca în mesajul lor *Terminat*.

Tichete de sesiune

RFC 5077 extinde TLS prin utilizarea biletelor de sesiune, în loc de ID-uri de sesiune. Acesta definește o modalitate de a relua o sesiune TLS fără a necesita ca starea specifică sesiunii să fie stocată pe serverul TLS.

Când se utilizează bilete de sesiune, serverul TLS stochează starea sa specifică sesiunii într-un bilet de sesiune și trimite biletul de sesiune clientului TLS pentru stocare. Clientul reia o sesiune TLS prin trimiterea biletului de sesiune la server, iar serverul reia sesiunea TLS conform stării specifice sesiunii din bilet. Biletul de sesiune este criptat și autentificat de server, iar serverul își verifică valabilitatea înainte de a-și folosi conținutul.

Un punct slab al acestei metode cu OpenSSL este că limitează întotdeauna securitatea de criptare și autentificare a biletului de sesiune TLS transmis la `AES128-CBC-SHA256`, indiferent de ce alți parametri TLS au fost negociați pentru sesiunea TLS reală. Aceasta înseamnă că informațiile de stare (biletul de sesiune TLS) nu sunt la fel de bine protejate ca sesiunea TLS în sine. O preocupare deosebită este stocarea de către OpenSSL a cheilor într-un context la nivel de aplicație (`SSL_CTX`), adică pe durata de viață a aplicației și nepermiterea reintroducerii `AES128-CBC-SHA256` tichetelor de sesiune TLS fără a reseta contextul OpenSSL la nivel de aplicație (ceea ce este mai puțin obișnuit)., predispus la erori și necesită adesea intervenție administrativă manuală).

Înregistrare TLS

Acesta este formatul general al tuturor înregistrărilor TLS.

Format de înregistrare TLS, general				
Decalaj	**Octet+0**	**Octet+1**	**Octet+2**	**Octet+3**
Octet 0	Tipul de conținut	—		
Octeții 1–4	Versiune moștenită		Lungime	
	(Major)	*(Minor)*	*(biții 15–8)*	*(biții 7–0)*
Octeți 5–(*m* −1)	Mesaj(e) de protocol			
Octeți *m* –(*p* −1)	MAC (opțional)			
Octeți *p* –(*q* −1)	Padding (doar cifrurile bloc)			

Tipul de conținut

Acest câmp identifică tipul de protocol al stratului de înregistrare conținut în această înregistrare.

Tipuri de conținut		
Hex	**Dec**	**Tip**
0×14	20	ChangeCipherSpec
0×15	21	Alerta
0×16	22	Handshake
0×17	23	Aplicație
0×18	24	Bătăile inimii

Versiune moștenită

Acest câmp identifică versiunea majoră și minoră a TLS anterioară TLS 1.3 pentru mesajul conținut. Pentru un mesaj ClientHello, aceasta nu trebuie să fie cea *mai înaltă* versiune acceptată de

client. Pentru TLS 1.3 și versiunile ulterioare, acesta trebuie să fie setat 0x0303, iar aplicația trebuie să trimită versiuni acceptate într-un bloc suplimentar de extensii de mesaje.

Versiuni		
Versiune majoră	**Versiune minoră**	**Tip versiune**
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

Lungime

Lungimea câmpurilor „mesaj(e) de protocol”, „MAC” și „padding” combinate (adică q −5), să nu depășească 2^{14} octeți (16 KiB).

Mesaj(e) de protocol

Unul sau mai multe mesaje identificate prin câmpul Protocol. Rețineți că acest câmp poate fi criptat în funcție de starea conexiunii.

MAC și padding

Un cod de autentificare a mesajului calculat în câmpul „mesaj(e) de protocol”, cu material cheie suplimentar inclus. Rețineți că acest câmp poate fi criptat sau să nu fie inclus în întregime, în funcție de starea conexiunii.

Nu pot fi prezente câmpuri „MAC” sau „padding” la sfârșitul înregistrărilor TLS înainte ca toți algoritmii și parametrii de cifrare să fi fost negociați și comunicați și apoi confirmați prin trimiterea unei înregistrări CipherStateChange (vezi mai jos) pentru a semnala că acești parametri vor intra în vigoare în toate înregistrări suplimentare trimise de același peer.

Protocol de Handshake

Majoritatea mesajelor schimbate în timpul configurării sesiunii TLS se bazează pe această înregistrare, cu excepția cazului în care apare o eroare sau avertizare și trebuie semnalată printr-o înregistrare a protocolului de alertă (vezi mai jos), sau modul de criptare al sesiunii este modificat de o altă înregistrare (vezi protocolul ChangeCipherSpec de mai jos).

Format de înregistrare TLS pentru protocolul de Handshake				
Decalaj	**Octet+0**	**Octet+1**	**Octet+2**	**Octet+3**
Octet 0	22	—		
	Versiune moștenită		Lungime	

Octeții 1–4	*(Major)*	*(Minor)*	*(biții 15–8)*	*(biții 7–0)*
Octeții 5–8	Tipul mesajului	Lungimea datelor mesajului de Handshake		
		(biții 23–16)	*(biții 15–8)*	*(biții 7–0)*
Octeți 9–(*n* −1)	Datele mesajului de Handshake			
Octeți *n* –(*n* +3)	Tipul mesajului	Lungimea datelor mesajului de Handshake		
		(biții 23–16)	*(biții 15–8)*	*(biții 7–0)*
Octeți (*n* +4)–	Datele mesajului de Handshake			

Tipul mesajului

Acest câmp identifică tipul de mesaj de Handshake.

Tipuri de mesaje	
Cod	**Descriere**
0	HelloRequest
1	ClientSalut
2	ServerSalut
4	NewSessionTicket
8	EncryptedExtensions (numai TLS 1.3)
11	Certificat
12	ServerKeyExchange

13	Cerere de certificat
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Terminat

Lungimea datelor mesajului de Handshake

Acesta este un câmp de 3 octeți care indică lungimea datelor de Handshake, fără a include antetul.

Rețineți că mai multe mesaje de Handshake pot fi combinate într-o singură înregistrare.

Protocol de alertă

În mod normal, această înregistrare nu ar trebui să fie trimisă în timpul schimburilor normale de handshake sau de aplicații. Cu toate acestea, acest mesaj poate fi trimis în orice moment în timpul Handshake și până la închiderea sesiunii. Dacă aceasta este folosită pentru a semnala o eroare fatală, sesiunea va fi închisă imediat după trimiterea acestei înregistrări, astfel încât această înregistrare este folosită pentru a oferi un motiv pentru această închidere. Dacă nivelul de alertă este marcat ca un avertisment, telecomanda poate decide să închidă sesiunea dacă decide că sesiunea nu este suficient de fiabilă pentru nevoile sale (înainte de a face acest lucru, telecomanda poate trimite și propriul semnal).

Format de înregistrare TLS pentru protocolul de alertă

Decalaj	**Octet+0**	**Octet+1**	**Octet+2**	**Octet+3**
Octet 0	21	—		
Octeții 1–4	Versiune moștenită		Lungime	
	(Major)	*(Minor)*	0	2
Octeții 5–6	Nivel	Descriere	—	
Octeți *7* –(*p* −1)	MAC (opțional)			
Octeți *p* –(*q* −1)	Padding (doar cifrurile bloc)			

Nivel

Acest câmp identifică nivelul de alertă. Dacă nivelul este fatal, expeditorul ar trebui să închidă imediat sesiunea. În caz contrar, destinatarul poate decide să încheie sesiunea în sine, prin trimiterea propriei alerte fatale și închiderea sesiunii în sine imediat după trimiterea acesteia. Utilizarea înregistrărilor de alertă este opțională, însă dacă lipsește înainte de închiderea sesiunii, sesiunea poate fi reluată automat (cu Handshake).
Închiderea normală a unei sesiuni după terminarea aplicației transportate ar trebui, de preferință, să fie alertată cu cel puțin tipul de alertă *de notificare Închidere* (cu un nivel de avertizare simplu) pentru a preveni o astfel de reluare automată a unei noi sesiuni. Semnalarea explicită a închiderii normale a unei sesiuni securizate înainte de închiderea efectivă a stratului său de transport este utilă pentru a preveni sau detecta atacuri (cum ar fi încercările de a trunchia datele transportate în siguranță, dacă în mod intrinsec nu are o lungime sau o durată predeterminată pe care destinatarul datelor securizate). se poate aștepta).

Tipuri de nivel de alertă

Cod	Tipul de nivel	Starea conexiunii
1	**avertizare**	conexiunea sau securitatea pot fi instabile.
2	**fatal**	conexiunea sau securitatea pot fi compromise sau a apărut o eroare irecuperabilă.

Descriere

Acest câmp identifică ce tip de alertă este trimis.

Tipuri de descriere a alertelor

Cod	Descriere	Tipuri de nivel	Notă
0	Închideți notificarea	**avertisment / fatal**	
10	Mesaj neașteptat	**fatal**	
20	Înregistrare proastă MAC	**fatal**	Este posibil ca o implementare SSL proastă sau sarcina utilă a fost modificată, de exemplu, regula firewall FTP pe serverul FTPS.
21	Decriptarea a eșuat	**fatal**	Numai TLS, rezervat
22	Înregistrare depășire	**fatal**	Numai TLS
30	Eșecul decompresiei	**fatal**	
40	Eșecul Handshake	**fatal**	

41	Fără certificat	**avertisment / fatal**	Doar SSL 3.0, rezervat
42	Certificat prost	**avertisment / fatal**	
43	Certificat neacceptat	**avertisment / fatal**	de exemplu, certificatul are activată numai utilizarea autentificarea serverului și este prezentat ca un certificat de client
44	Certificat revocat	**avertisment / fatal**	
45	Certificatul a expirat	**avertisment / fatal**	Verificați expirarea certificatului de server, de asemenea, verificați că niciun certificat din lanțul prezentat nu a expirat
46	Certificat necunoscut	**avertisment / fatal**	
47	Parametru ilegal	**fatal**	
48	CA necunoscută (autoritate de certificare)	**fatal**	Numai TLS
49	Acces interzis	**fatal**	Numai TLS – de exemplu nu a fost prezentat niciun certificat de client (TLS: mesaj de certificat necompletat sau SSLv3: alertă fără certificat), dar serverul este configurat să necesite unul.
50	Eroare de decodare	**fatal**	Numai TLS
51	Eroare de decriptare	**avertisment / fatal**	Numai TLS
60	Restricție la export	**fatal**	Numai TLS, rezervat
70	Versiunea protocolului	**fatal**	Numai TLS
71	Securitate insuficientă	**fatal**	Numai TLS
80	Eroare internă	**fatal**	Numai TLS

86	Soluție inadecvată	**fatal**	Numai TLS
90	Utilizatorul a fost anulat	**fatal**	Numai TLS
100	Fără renegociere	**avertizare**	Numai TLS
110	Extensie neacceptată	**avertizare**	Numai TLS
111	Certificat de neobținut	**avertizare**	Numai TLS
112	Nume nerecunoscut	**avertisment / fatal**	Numai TLS; Indicatorul numelui serverului clientului a specificat un nume de gazdă neacceptat de server
113	Răspuns greșit privind starea certificatului	**fatal**	Numai TLS
114	Valoare hash a certificatului incorectă	**fatal**	Numai TLS
115	Identitate PSK necunoscută (utilizată în TLS-PSK și TLS-SRP)	**fatal**	Numai TLS
116	Se cere certificat	**fatal**	Doar versiunea TLS 1.3
120 sau 255	Fără protocol de aplicare	**fatal**	Doar versiunea TLS 1.3

Protocolul ChangeCipherSpec

Format de înregistrare TLS pentru protocolul ChangeCipherSpec

Decalaj	**Octet+0**	**Octet+1**	**Octet+2**	**Octet+3**
Octet 0	20	—		
Octeții 1–4	Versiune moștenită		Lungime	
	(Major)	*(Minor)*	0	1

Octetul 5	Tipul de protocol CCS	—	

Tipul de protocol CCS

Momentan doar 1.

Protocolul de aplicare

Format de înregistrare TLS pentru protocolul de aplicare

Decalaj	**Octet+0**	**Octet+1**	**Octet+2**	**Octet+3**
Octet 0	23	—		
Octeții 1–4	Versiune moștenită		Lungime	
	(Major)	*(Minor)*	*(biții 15–8)*	*(biții 7–0)*
Octeți 5–(*m* −1)	Application Data			
Octeți *m* –(*p* −1)	MAC (opțional)			
Octeți *p* –(*q* −1)	Padding (doar cifrurile bloc)			

Lungime

Lungimea datelor aplicației (excluzând antetul protocolului și inclusiv MAC și trailere de padding)

MAC

32 de octeți pentru HMAC bazat pe SHA-256, 20 de octeți pentru HMAC bazat pe SHA-1, 16 octeți pentru HMAC bazat pe MD5.

Padding

Lungime variabila; ultimul octet conține lungimea de padding.

Suport pentru servere virtuale bazate pe nume

Din punct de vedere al protocolului de aplicație, TLS aparține unui strat inferior, deși modelul TCP/IP este prea grosier pentru a-l arăta. Aceasta înseamnă că Handshake-ul TLS este de obicei efectuată (cu excepția cazului STARTTLS) înainte ca protocolul de aplicare să poată începe. În caracteristica serverului virtual bazată pe nume furnizată de stratul de aplicație, toate serverele virtuale găzduite în comun partajează același certificat, deoarece serverul trebuie să selecteze și să trimită un certificat imediat după mesajul ClientHello. Aceasta este o mare problemă în mediile de găzduire, deoarece înseamnă fie partajarea aceluiași certificat între toți clienții, fie utilizarea unei adrese IP diferite pentru fiecare dintre ei.

Există două soluții cunoscute oferite de X.509 :

- Dacă toate serverele virtuale aparțin aceluiași domeniu, se poate folosi un certificat wildcard. [171] Pe lângă selecția liberă a numelui gazdei care ar putea fi o problemă sau nu, nu există un acord comun cu privire

la modul de potrivire a certificatelor wildcard. Se aplică reguli diferite în funcție de protocolul aplicației sau de software-ul utilizat. [172]

- Adăugați fiecare nume de gazdă virtuală în extensia subjectAltName. Problema majoră este că certificatul trebuie reemis ori de câte ori este adăugat un nou server virtual.

Pentru a furniza numele serverului, extensiile RFC 4366 Transport Layer Security (TLS) permit clienților să includă o extensie Server Name Indication (SNI) în mesajul extins ClientHello. Această extensie sugerează imediat serverului la ce nume dorește să se conecteze clientul, astfel încât serverul să poată selecta certificatul potrivit pentru a-l trimite clienților.

RFC 2817 documentează, de asemenea, o metodă de implementare a găzduirii virtuale bazate pe nume prin actualizarea HTTP la TLS printr-un antet de actualizare HTTP/1.1. În mod normal, aceasta este pentru a implementa în siguranță HTTP peste TLS în cadrul schemei URI „http” principale (care evită bifurcarea spațiului URI și reduce numărul de porturi utilizate), cu toate acestea, puține implementări suportă în prezent acest lucru.

Standarde

Standarde primare

Versiunea actuală aprobată a (D)TLS este versiunea 1.3, care este specificată în:

- RFC 8446 : „Protocolul Transport Layer Security (TLS) Versiunea 1.3”.
- RFC 9147 : „Protocolul Datagram Transport Layer Security (DTLS) Versiunea 1.3”

Standardele actuale înlocuiesc aceste versiuni anterioare, care sunt acum considerate învechite:

- RFC 5246 : „Protocolul Transport Layer Security (TLS) Versiunea 1.2”.
- RFC 6347 : „Datagram Transport Layer Security Versiunea 1.2”
- RFC 4346 : „Protocolul Transport Layer Security (TLS) Versiunea 1.1”.
- RFC 4347 " "Securitatea Transport Layer de date"
- RFC 2246 : „Versiunea 1.0 a protocolului TLS”.
- RFC 6101 : „Protocolul Secure Sockets Layer (SSL) Versiunea 3.0”.
- Internet Draft (1995) : „Protocolul SSL”

Extensii

Alte RFC-uri au extins ulterior (D)TLS.

Extensiile la (D)TLS 1.3 includ:

- RFC 9367 : „GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3”.

Extensiile la (D)TLS 1.2 includ:

- RFC 5288 : „Suite de cifrare AES Galois Counter Mode (GCM) pentru TLS”.
- RFC 5289 : „Suite de cifrare pentru curbe eliptice TLS cu SHA-256/384 și modul de contorizare AES Galois (GCM)”.
- RFC 5746 : „Extensie de indicație de renegociere pentru securitatea Transport Layer (TLS)”.
- RFC 5878 : „Extensii de autorizare Transport Layer Security (TLS)”.
- RFC 5932 : „Camellia Cipher Suites for TLS”
- RFC 6066 : „Extensii Transport Layer Security (TLS): Definiții de extensie”, include indicația numelui serverului și capsarea OCSP.
- RFC 6091 : „Utilizarea cheilor OpenPGP pentru autentificarea Transport Layer Security (TLS)”.
- RFC 6176 : „Interzicerea Secure Sockets Layer (SSL) versiunea 2.0”.
- RFC 6209 : „Adăugarea suitelor ARIA Cipher la Transport Layer Security (TLS)”.
- RFC 6347 : „Datagram Transport Layer Security Versiunea 1.2”.
- RFC 6367 : „Adăugarea suitelor Camellia Cipher la Transport Layer Security (TLS)”.
- RFC 6460 : „Profil suită B pentru securitatea Transport Layer (TLS)”.
- RFC 6655 : „AES-CCM Cipher Suites for Transport Layer Security (TLS)”.
- RFC 7027 : „Criptografie cu curbe eliptice (ECC) Brainpool Curves for Transport Layer Security (TLS)”.
- RFC 7251 : „Suite de criptare a curbei eliptice AES-CCM (ECC) pentru TLS”.
- RFC 7301 : „Extensie de negociere a protocolului de nivel de aplicație (Transport Layer Security (TLS)) ”.
- RFC 7366 : „Criptați-apoi-MAC pentru securitatea Transport Layer (TLS) și securitatea Transport Layer al datagramelor (DTLS)”.

- RFC 7465 : „Interzicerea suitelor de cifrare RC4”.
- RFC 7507 : „TLS Fallback Signaling Cipher Suite Value (SCSV) pentru prevenirea atacurilor de downgrade a protocolului”.
- RFC 7568 : „Deprecierea Versiunii 3.0 a Secure Sockets Layer”.
- RFC 7627 : „Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension”.
- RFC 7685 : „O extensie de padding ClientHello pentru securitatea Transport Layer (TLS)”.
- RFC 9189 : „GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.2”.

Extensiile la (D)TLS 1.1 includ:

- RFC 4366 : „Extensii Transport Layer Security (TLS)” descrie atât un set de extensii specifice, cât și un mecanism de extensie generic.
- RFC 4492 : „ Suite de criptare cu curbe eliptice (ECC) pentru securitatea Transport Layer (TLS)”.
- RFC 4680 : „Mesaj de Handshake TLS pentru date suplimentare”.
- RFC 4681 : „Extensie de mapare a utilizatorului TLS”.
- RFC 4785 : „Suite de criptare Pre-Shared Key (PSK) cu criptare NULL pentru securitatea Transport Layer (TLS)”.
- RFC 5054 : „Utilizarea protocolului SRP (Secure Remote Password) pentru autentificarea TLS”. Definește suitele de criptare TLS-SRP.
- RFC 5077 : „Reluarea sesiunii de securitate a Transport Layer (TLS) fără starea serverului”.
- RFC 5081 : „Utilizarea cheilor OpenPGP pentru autentificarea Transport Layer Security (TLS)”, învechit de RFC 6091.
- RFC 5216 : „ Protocolul de autentificare EAP -TLS”

Extensiile la TLS 1.0 includ:

- RFC 2595 : „Utilizarea TLS cu IMAP, POP3 și ACAP”. Specifică o extensie a serviciilor IMAP, POP3 și ACAP care permit serverului și clientului să utilizeze securitatea nivelului de transport pentru a furniza comunicații private și autentificate prin Internet.
- RFC 2712 : „Adăugarea suitelor de criptare Kerberos la Transport Layer Security (TLS)”. Suitele de criptare pe 40 de biți definite în acest memoriu apar doar în scopul documentării faptului că acele coduri ale suitelor de criptare au fost deja atribuite.
- RFC 2817 : „Actualizarea la TLS în cadrul HTTP/1.1”, explică cum să utilizați mecanismul de actualizare în HTTP/1.1 pentru a iniția Transport Layer Security (TLS) printr-o conexiune TCP existentă. Acest lucru permite traficului HTTP nesecurizat și securizat să partajeze același port *binecunoscut* (în acest caz, http: la 80 mai degrabă decât https: la 443).
- RFC 2818 : „HTTP Over TLS”, distinge traficul securizat de traficul nesecurizat prin utilizarea unui „port de server” diferit.
- RFC 3207 : „Extensie de serviciu SMTP pentru SMTP securizat peste securitatea Transport Layer”. Specifică o extensie a serviciului SMTP care permite unui server SMTP și unui client să utilizeze securitatea nivelului de transport pentru a furniza comunicații private și autentificate prin Internet.
- RFC 3268 : „AES Ciphersuites pentru TLS”. Adaugă suite de criptare Advanced Encryption Standard (AES) la cifrurile simetrice existente anterior.
- RFC 3546 : „Extensii Transport Layer Security (TLS)”, adaugă un mecanism pentru negocierea extensiilor de protocol în timpul inițializării sesiunii și definește unele extensii. Devenit învechit de RFC 4366.
- RFC 3749 : „Metode de comprimare a protocolului de securitate al Transport Layer”, specifică cadrul pentru metodele de compresie și metoda de compresie DEFLATE.
- RFC 3943 : „Comprimarea protocolului Transport Layer Security (TLS) folosind Lempel-Ziv-Stac (LZS)”.
- RFC 4132 : „Adăugarea suitelor Camellia Cipher la Transport Layer Security (TLS)”.
- RFC 4162 : „Adăugarea suitelor SEED Cipher la Transport Layer Security (TLS)”.
- RFC 4217 : „Securizarea FTP cu TLS ”.
- RFC 4279 : „Suite de cifră cu chei pre-partajate pentru securitatea Transport Layer (TLS)”, adaugă trei seturi de suite de cifră noi pentru protocolul TLS pentru a sprijini autentificarea bazată pe chei pre-partajate.

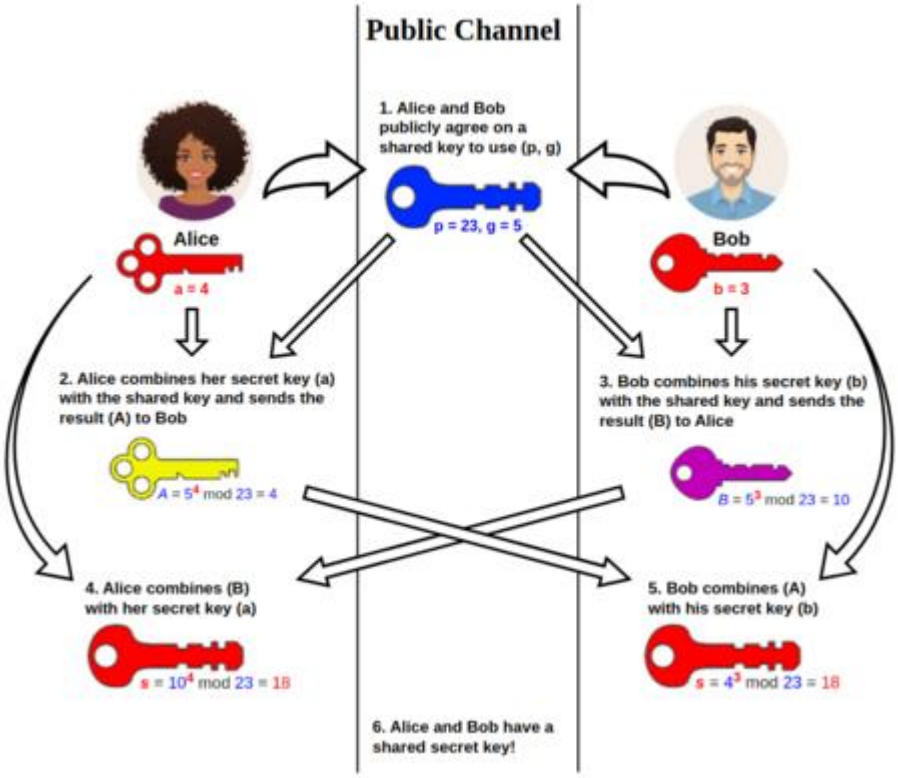
RFC-uri informaționale

- RFC 7457 : „Rezumarea atacurilor cunoscute asupra securității Transport Layer (TLS) și a datagramelor TLS (DTLS)”
- RFC 7525 : „Recomandări pentru utilizarea în siguranță a securității Transport Layer (TLS) și a securității Transport Layer al datagramelor (DTLS)”

Lucrări citate

- *Sullivan, Nick (26.12.2017). „De ce TLS 1.3 nu este încă în browsere”. Blogul Cloudflare. Accesat 2020-03-14.*
- *Thomson, Martin; Pauly, Tommy (decembrie 2021). Viabilitatea pe termen lung a mecanismelor de extindere a protocolului. doi : 10.17487/RFC9170. RFC 9170.*

*Diffie-Hellman



Cea mai simplă și cea mai originală implementare, [2] oficializată ulterior ca **Finite Field Diffie-Hellman** în *RFC 7919*, [9] a protocolului folosește grupul multiplicativ de numere întregi modulo *p*, unde *p* este prim și *g* este o rădăcină primitivă modulo *p*. Aceste două valori sunt alese în acest fel pentru a se asigura că secretul partajat rezultat poate lua orice valoare de la 1 la *p* –1. Iată un exemplu de protocol, cu valori non-secrete în albastru și valori secrete în **roșu**.

1. Alice și Bob sunt de acord public să utilizeze un modul $p = 23$ și baza $g = 5$ (care este o rădăcină primitivă modulo 23).
2. Alice alege un întreg secret $\boldsymbol{a} = 4$, apoi îi trimite lui Bob $A = g^a \bmod p$
   - $A = 5^4 \bmod 23 = 4$ (în acest exemplu atât A cât și a au aceeași valoare 4, dar de obicei nu este cazul)
3. Bob alege un întreg secret $\boldsymbol{b} = 3$, apoi trimite Alice $B = g^b \bmod p$
   - $B = 5^3 \bmod 23 = 10$
4. Alice calculează $\boldsymbol{s} = B^a \bmod p$
   - $\boldsymbol{s} = 10^4 \bmod 23 = 18$
5. Bob calculează $\boldsymbol{s} = A^b \bmod p$
   - $\boldsymbol{s} = 4^3 \bmod 23 = 18$
6. Alice și Bob împărtășesc acum un secret (numărul 18).

Atât Alice, cât și Bob au ajuns la aceleași valori deoarece sub *mod p*,

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

Mai exact,

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

Doar *a* și *b* sunt ținute secrete. Toate celelalte valori – *p*, *g*, $g^a \bmod p$ și $g^b \bmod p$ – sunt trimise în clar. Puterea schemei provine din faptul că $g^{ab} \bmod p = g^{ba} \bmod p$ durează foarte mult timp pentru a calcula cu ajutorul oricărui algoritm cunoscut doar din cunoștințele lui *p*, *g*, $g^a \bmod p$ și $g^b \bmod p$. O astfel de funcție care este ușor de calculat, dar greu de inversat se numește funcție unidirecțională. Odată ce Alice și Bob calculează secretul partajat, îl pot folosi ca o cheie de criptare, cunoscută doar de ei, pentru a trimite mesaje prin același canal de comunicații deschis.

Desigur, ar fi necesare valori mult mai mari ale *a*, *b* și *p* pentru a asigura acest exemplu, deoarece există doar 23 de rezultate posibile ale *n* mod 23. Cu toate acestea, dacă *p* este un prim de cel puțin 600 de cifre, atunci chiar și cele mai rapide calculatoare moderne care folosesc cel mai rapid algoritm cunoscut nu pot *găsi* doar *g*, *p* și $g^a \bmod p$.

O astfel de problemă se numește problema logaritmului discret. Calculul $g^a \bmod p$ este cunoscut sub numele de exponențiere modulară și poate fi făcut eficient chiar și pentru numere mari. Rețineți că *g* nu trebuie să fie deloc mare și, în practică, este de obicei un număr întreg mic (cum ar fi 2, 3,...).

** OSI Model

OSI model

Layer			Protocol data unit (PDU)	Function
Host layers	7	Application	Data	High-level protocols such as for resource sharing or remote file access, e.g. HTTP.
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Transmission of data frames between two nodes connected by a physical layer
	1	Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium