

Kriptográfia és Információbiztonság

6. előadás

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
`mggyongyi@ms.sapientia.ro`

2022

Webbiztonság (web security)

Biztonsági szempontok:

- a WWW lényegében egy kliens/szerver alkalmazás, amely az Interneten és a TCP/IP intraneten keresztül fut
- egyszerű:
 - a webböngészők használata
 - a webszerverek konfigurálása és kezelése
 - a webtartalom fejlesztése
- a web alapjául szolgáló szoftver **komplex**, számos **biztonsági hibát** tartalmazhat, **sebezhető** lehet a különféle támadásokkal szemben
- webszerver: egy vállalat, egy intézmény teljes számítógéprendszerének az indítópadjaként funkcionál
- sikeres támadás esetén a támadó hozzáférhet érzékeny adatokhoz, amelyek a lokális szerveren vannak tárolva

Webbiztonság, biztonsági szempontok

Biztonsági szempontok:

- a webalapú szolgáltatásokat gyakran igénybe veszik olyan felhasználók akik
 - nincsenek tisztában az alapvető biztonsági kérdésekkel
 - nem látják át az alapvető biztonsági kockázatokat
 - nem rendelkeznek megfelelő ismeretekkel
 - nincsenek alkalmas eszközeik, amelyek segítségével védekezni tudnának

Webbiztonság, veszélyek

- integritás (**integrity**):
 - fenyegetés: a felhasználó adatainak, a memóriának, a tranzit üzenetforgalomnak a módosítása, trójai faló a böngészőben
 - következmények: információvesztés, a számítógép teljes sérülése
 - megoldás: MAC használata
- bizalmasság (**confidentiality**):
 - fenyegetés: adatszerzés szerverről, ügyfelektől, a hálózati konfigurációról, a kliens szerver kommunikációról
 - következmények: információvesztés, a magánélet elvesztése, a szoftver megsemmisítése
 - megoldás: titkosítás, proxy szerver

Webbiztonság, veszélyek

- szolgáltatás megtagadás (**denial of service**):
 - fenyegetés: a felhasználó thread-ek megsemmisítése, a gép elárasztása kérésekkel, a memória megtöltése
 - következmények: bosszantó, a feladatvégzések megszakítása
 - megoldás: nehéz kivédeni
- hitelesítés (**authentication**):
 - fenyegetés: legális felhasználók megszemélyesítése, adathamisítás
 - következmények: a felhasználók félrevezetése, hamis adatok valódinak való feltüntetése
 - megoldás: kriptográfiai technikák

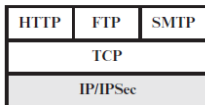
Webbiztonság, fenyegetések

A fenyegetéseket kétféle szempont szerint lehet csoportosítani:

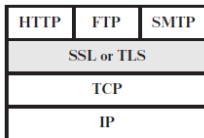
- passzív és aktív támadások
 - **passzív támadások**
 - a böngésző és a szerver közötti hálózati forgalom lehallgatása
 - a biztonságosnak hitt webhelyen található információkhoz való hozzáférés
 - **aktív támadások**
 - egy másik felhasználó megszemélyesítése, a kliens és a kiszolgáló közötti adatátvitel során
 - az üzenetek megváltoztatása az adatátvitel során
 - a webhelyen található információk megváltoztatása
- a fenyegetés helye szerint: ez lehet a webszerver, a webböngésző, valamint a böngésző és a szerver közötti hálózati forgalom

Webbiztonság, megvalósítások

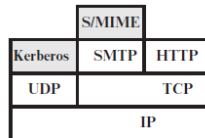
- a webbiztonságot többféleképpen lehet megvalósítani
- a megvalósítások a szolgáltatásokat figyelembe véve nem sokban különböznek
- a megvalósítások az alkalmazási területet, illetve a TCP/IP protokoll vermen belüli helyüket figyelembe véve azonban különböznek
- megkülönböztetünk hálózati (**network level**), szállítási (**transport level**), illetve alkalmazási rétegben (**application level**) megvalósuló biztonságot



(a) Network level



(b) Transport level

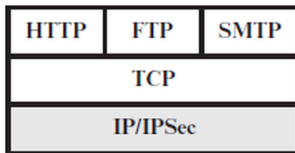


(c) Application level

Webbiztonság, megvalósítások

a **hálózati rétegben** megvalósuló webbiztonságot az IP security (IPsec) biztosítja

- előnye: átlátható a végfelhasználók és az alkalmazások számára
- általános célú megoldást kínál, szűrési lehetőséggel: csak a kiválasztott forgalom kell átmenjen az IPsec-feldolgozáson

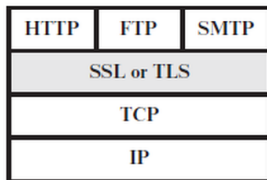


(a) Network level

Webbiztonság, megvalósítások

a **szállítási rétegben** megvalósuló biztonságot a TCP felett implementálják

- a Secure Sockets Layer (SSL) és az azt követő Transport Layer Security (TLS) biztosítja a biztonságot
- két implementációs lehetőség ismert:
 - az SSL (vagy TLS) az alapul szolgáló protokollcsomag része, ezért átlátható az alkalmazások számára
 - az SSL (vagy TLS) meghatározott csomagokba van beágyazva, például minden böngészőben benne van a TLS

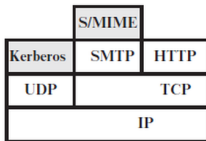


(b) Transport level

Webbiztonság, megvalósítások

az **alkalmazási rétegben** megvalósuló biztonság

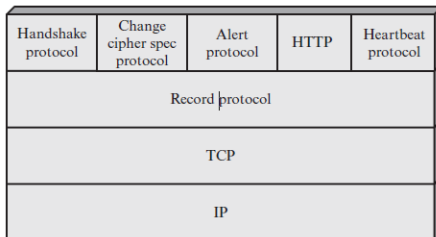
- a biztonsági szolgáltatások egy adott alkalmazásba vannak beágyazva
- előnye, hogy a szolgáltatás az adott alkalmazás speciális igényeihez szabható



(c) Application level

A TLS, Transport Layer Security

- az Internet Engineering Task Force (IETF) által ajánlott internetes szabvány
- először 1999-ben írták le, a jelenlegi standard a TLS 1.3, ezt 2018-ban definiálták
- az első SSL specifikációknak is része volt (1994, 1995, 1996), amelyeket a Netscape Communications fejlesztett a saját webböngészői számára
- általános célú szolgáltatás, amelyet TCP-n alapuló protokollok részeként valósítanak meg
- kliens/szerver alkalmazásokban a kommunikáció lehallgatása és manipulálása ellen nyújt védelmet



A TLS, Transport Layer Security

TLS-kapcsolat (**connection**):

- a kliens/szerver alkalmazásokban az alkalmazásoknak kérnie kell a szervertől a TLS-kapcsolat létrehozását (anélkül is tudnak kommunikálni)
- a TLS kapcsolat létrehozásához:
 - más portszámot használnak a TLS-kapcsolatokhoz, pl a 80-as portot a titkosítatlan HTTP-forgalomhoz, a 443-as portot a titkosított HTTPS-forgalomhoz
 - a kliens protokoll-specifikus kérést küld a szervernek, hogy kapcsolja át a TLS-re; például STARTTLS kérést kezdeményez a levelezési, a hír-protokollok esetében
- a felek között több biztonságos kapcsolat is létezhet

A TLS, Transport Layer Security

TLS-munkamenet (**session**):

- a munkameneteket a Handshake protokoll hozza létre
- egy munkameneten belül több kriptográfiai paraméterben is megegyeznek, amelyeket több kapcsolaton belül is lehet használni
- elméletileg egyidejűleg több munkamenet is előfordulhat a felek között, de ezt a funkciót a gyakorlatban nem használják
- az egyes munkamenetekhez több állapot tartozik: aktuális működési állapot íráshoz/olvasáshoz, függőben lévő működési állapot íráshoz/olvasáshoz

A TLS, Transport Layer Security

egy TLS-kapcsolat paraméterei (**connection's parameters**) :

- a szerver és kliens random értéke: minden kapcsolathoz más random értéket választanak ki
- a szerver MAC értéke: a szerver által küldött adatok esetében használt titkos MAC érték/kulcs
- a kliens MAC értéke: a kliens által küldött adatok esetében használt titkos MAC érték/kulcs
- a szerver titkos kulcsa: a szimmetrikus titkosítási eljárásokban használt kulcs, a szerver titkosít ezzel a kulccsal a kliens meg a visszafejtéshez használja
- a kliens titkos kulcsa: a szimmetrikus titkosítási eljárásokban használt kulcs, a kliens titkosít ezzel a kulccsal a szerver meg a visszafejtéshez használja
- inicializálási vektor (IV): CBC módban esetében minden kulcshoz más inicializálási vektort (IV) társítunk, kezdetben a TLS Handshake Protokoll inicializálja
- sorszámok: kapcsolatonként mindegyik fél külön sorszámozza az elküldött és fogadott üzeneteket. Amikor az egyik fél "change cipher spec message" üzenetet küld vagy fogad, a megfelelő sorszám nullára lesz állítva. A sorszámok nem haladhatják meg a $2^{64} - 1$ értéket.

A TLS, Transport Layer Security

TLS-munkamenet (session) paraméterei (**session's parameters**):

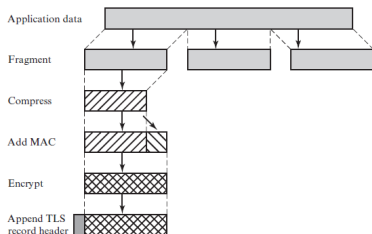
- a munkamenet azonosító: a szerver által választott tetszőleges bájt szekvencia
- peer tanúsítvány: egy X509.v3 tanúsítvány, lehet null is
- tömörítési módszer: az adatok tömörítésére használt algoritmus (titkosítás előtt)
- a titkosítás paraméterei: meghatározza a MAC-hez használt titkosítási algoritmust (pl. AES), a hash függvényt (pl. SHA-1), a hash méretet
- mester kulcs/titok (master secret): a kliens és a szerver között megosztott 48 bájtos kulcs/titok.
- jelző érték: amely jelzi, hogy a munkamenet használható-e új kapcsolatok indítására

A TLS Record Protokoll

két fontos szolgáltatása:

- bizalmas kommunikáció (**confidentiality**):
 - a Handshake Protokoll létrehoz egy titkos kulcsot a titkosításhoz
 - a használt titkosítási algoritmusok: AES(128, 256), 3DES(168), RC4-128(128)
- üzenetrk sértetlensége (**message integrity**):
 - a Handshake Protokoll létrehoz egy titkos kulcsot a MAC-hez
 - a HMAC algoritmust használja, amelynek leírása az RFC 21024-ben van
 - $HMAC_{key}(m) = H[(key^+ \oplus opad) || H[(key^+ \oplus ipad) || m]]$
 - az *ipad*, *opad* konstansok, *H* hash függvény, *key* a titkos kulcs (szükség esetén paddingolni kell), *m* az üzenet

A TLS Record Protokoll



- fragmentálás (**fragment**): minden üzenetet (application data) legtöbb 2^{14} bájt méretű blokkra oszt
- tömörítés (**compress**): opcionális, veszteségmentesnek kell lennie, és a tartalmat nem növelheti több mint 1024 bájjal
- MAC alkalmazása (**add MAC**): a HMAC kerül alkalmazásra
- titkosítás (**encrypt**): 128 bites kulcsméretű szimmetrikus titkosítás, a tartalmat nem növelheti több mint 1024 bájjal, AES(128, 256), 3DES-168, RC4-128. Blokk titkosítás esetében a padding-et a MAC után végzi
- a header hozzáadása (**append TLS record header**): négy típusú mező hozzáadását jelenti, az első három a fragment feldolgozásához használt módszer írja le, a negyedik az üzenet hosszát tartalmazza

A TLS Change Cipher Spec és Alert Protokollok

Change Cipher Spec protokoll:

- egyetlen egy bájtos üzenetből áll; jelzi, ha a munkamenet titkosítási módját egy másik rekord módosítja

Alert Protokoll:

- két bájtból áll, az első warning, vagy fatal értéket jelent, a második a riasztás kódját tartalmazza
- normal handshake vagy alkalmazáscsere esetén nem kerül elküldésre, de bármikor amikor riasztás szükséges elküldhető
- a munkamenet lezárását is maga után vonhatja

A TLS Heartbeat Protokoll

- 2012-ben határozták meg az RFC 6250-ben
- két célt szolgál:
 - biztosítja a feladót, hogy a címzett még mindig *életben* van, még akkor is, ha egy ideig nem volt semmilyen tevékenység
 - aktivitást generál a kapcsolaton keresztül tétlenségi időszakokban, így elkerülhető a tétlen kapcsolatokat nem toleráló tűzfal azonnali bezárása
- egy hardver vagy szoftver által generált periodikus jel, amely jelzi a normál működést vagy a rendszer más részeinek szinkronizálását végzi
- a TLS Record Protokollon felül fut

A TLS Handshake Protokoll

- a TLS legösszetettebb része
- lehetővé teszi a szerver és a kliens kölcsönös hitelesítését
- lehetővé teszi, hogy a felek megállapodjanak, hogy milyen titkosító, illetve MAC-algoritmust használnak a kommunikáció során
- az alkalmazásadatok átvitele előtt használják

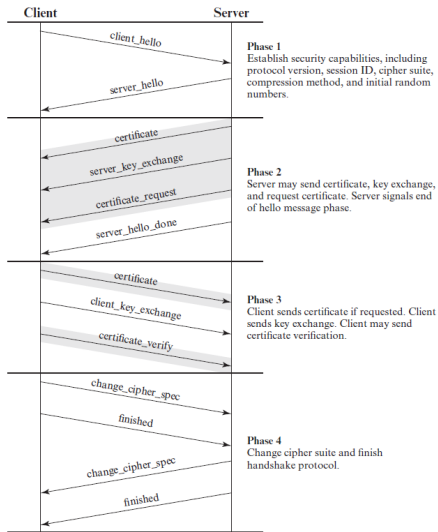
A TLS Handshake Protokoll

- a kliens és a szerver által váltott üzenetek sorozatából áll, ahol minden üzenet három mezővel rendelkezik:
 - a 10 lehetséges üzenet egyikét jelzi (1 bájt)
 - az üzenet hossza bájtban (3 bájt)
 - az üzenethez tartozó paraméterek

Az üzenetek a következő típusúak lehetnek:

típus	paraméter
hello_request	null
client_hello	version, random, session ID, cipher suite, compression method
server_hello	version, random, session ID, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

A TLS Handshake Protokoll



A TLS Handshake Protokoll

Az **1. fázisban** a kliens kezdeményezésére egy logikai kapcsolat jön létre, a megfelelő **biztonsági beállításokkal**. A kliens egy `client_hello` üzenetet küld, amelynek a következők a paraméterei:

- **version**: a kliens által ismert legnagyobb TLS verziószám
- **random**: a kliens generálja, egy 32 bites timestamp-ből és egy véletlenszám-generátor által generált számból (28 bájt) áll; nonces-ként a replay-attack támadások megelőzésére a kulcscsere során kerülnek használatra
- **session ID**: változó hosszúságú, a nullától eltérő érték jelzi, hogy a kliens frissíteni szeretné egy meglévő kapcsolata paramétereit. A nullás érték azt jelzi, hogy a kliens egy új kapcsolatot akar létrehozni.
- **cipher suite** (titkosító készlet): a kliens által támogatott kriptográfiai algoritmusokat tartalmazza, preferencia szerint csökkenő sorrendben. A lista minden eleme meghatároz egy kulcscsere-algoritmust és egy CipherSpec-et
- **compression method**: a kliens által támogatott tömörítési módszerek listája

A TLS Handshake Protokoll

A `server_hello`-nak ugyanazok a paraméterei, mint a `client_hello`-nak:

- **version**: a szerver által ismert legkisebb TLS verziószám
- **random**: a szerver által generált random érték, amely független a kliens random értékétől
- **session ID**: ha a kliens session ID-ja nullától különbözött, akkor a szerver ugyanezt az értéket fogja használni, másképp egy új munkemenethez szükséges érték lesz
- **cipher suite**: a kliens által megadott listából kiválasztott cipher suite-et fogja tartalmazni
- **compression method**: a kliens által megadott listából kiválasztott tömörítési módszert fogja tartalmazni

A TLS Handshake Protokoll

A cipher suite paraméter első eleme meghatározza a kulcscsere módszert, amely a következő lehet:

- RSA: a titkos kulcs a vevő RSA publikus kulcsával van titkosítva; elérhető kell legyen a vevő publikus kulcsának a tanúsítványa
- Fixed Diffie–Hellman: egy fix titkos kulcsot eredményez, ahol a szerver tanúsítványa tartalmazza a tanúsító hatóság (CA) által aláírt Diffie–Hellman publikus paramétereket. Ha a szerver igényli a kliens hitelesítését, akkor a kliensnek biztosítania kell a Diffie–Hellman publikus paramétereire kiállított tanúsítványt.
- Ephemeral Diffie–Hellman: átmeneti, egyszeri hitelesített titkos kulcsokat eredményez, ahol a Diffie–Hellman publikus paramétereket a küldő aláírja a saját RSA vagy DSS privát kulcsával. Az RSA, DSS publikus kulcsok hitelesítésére tanúsítványokat használnak. Ez tűnik a **legbiztonságosabbnak** a három Diffie–Hellman opció közül.
- Anonymous Diffie–Hellman: az alap Diffie–Hellman algoritmus kerül alkalmazásra, hitelesítés nélkül. Mindkét oldal elküldi a publikus Diffie–Hellman paramétereit a másiknak hitelesítés nélkül. **Támadható** a man-in-the middle típusú támadással.

A TLS Handshake Protokoll

A kulcscsere módszer definícióját követi a CipherSpec, amely a következő mezőket tartalmazza:

- CipherAlgorithm: a korábban felsorolt algoritmusok bármelyike: RC4, RC2, DES, 3DES, DES40 vagy IDEA
- MACAlgorithm: MD5 vagy SHA-1
- CipherType: Stream vagy Block
- isExportable: igaz vagy hamis
- HashSize: 0, 16 MD5 esetén, vagy 20 bájt SHA-1 esetén
- IV méret: A CBC mód használata esetén az inicializáló vektor értékének mérete

A TLS Handshake Protokoll

A 2. fázisban a **szerver hitelesítésére és a kulcscserére** kerül sor:

- ha szükség van a szerver hitelesítésére, a szerver elküldi a tanúsítványát; az üzenet egy vagy több X.509 tanúsítványt tartalmaz.
- a tanúsítványüzenetre minden kulcscsere módszer esetében szükség van, kivéve az anonymous Diffie–Hellmant
- ha a fixed Diffie–Hellman kulcserére kerül sor akkor ez a tanúsítványüzenet a szerver kulcscsere-üzeneteként működik, mivel tartalmazza a szerver publikus Diffie–Hellman paramétereit
- a `server_key_exchange` üzenetre **nincs szükség** ha
 - a szerver küldött egy tanúsítványt fixed Diffie–Hellman paraméterekkel
 - RSA kulcscserére fog sor kerülni

A TLS Handshake Protokoll

a `server_key_exchange` üzenetre **szükség van**:

- anonymous Diffie–Hellman esetében: az üzenet tartalmazza a két publikus Diffie–Hellman értéket (egy prímszámot és a generátor elemet) és a szerver publikus Diffie–Hellman kulcsát
- efemer Diffie–Hellman esetében: az üzenet tartalmazza az anonymous Diffie–Hellmanhoz szükséges három paraméter értékét, valamint ezen paraméterek aláírását
- RSA-kulcscsere esetében, amikor csak az aláírásra szolgáló RSA-kulccsal rendelkezik:
 - a kliens nem küldheti el egyszerűen a titkos kulcsot a szerver publikus kulcsával titkosítva
 - a szervernek létre kell hozzon egy ideiglenes RSA publikus/privát kulcspárt,
 - a `server_key_change` üzenettel el kell küldje az ideiglenes RSA publikus kulcsot (a modulus és kitevő értékeket) és ennek a publikus kulcsnak az aláírt értékét

A TLS Handshake Protokoll

Megjegyzések:

- az aláírás általában úgy jön létre, hogy meghatározzuk az üzenet hash értékét, majd titkosítjuk azt a küldő privát kulcsával
- az aláírás most a kezdeti hello üzenetek két nonce értékét is tartalmazza, hogy biztosítja a replay-attack támadásokkal szembeni védelmet:

`hash = hash (Client.random || Server.random || ServerParams)`

- DSS aláírás esetén a hash az SHA-1 algoritmussal történik
- RSA aláírás esetén egy MD5 és egy SHA-1 hash is kiszámításra kerül, és a két hash összefűzött értéke (36 bájt) lesz titkosítva a szerver privát kulcsával

A TLS Handshake Protokoll

Megjegyzések:

- anonymous Diffie–Hellman esetében a szerver kérheti a kliens tanúsítványát, ez két értéket megadását jelenti: a tanúsítvány típusát és a tanúsítványt kiállító hatóságok listáját, ahol a tanúsítvány típusa a következő lehet:
 - RSA, csak aláírás
 - DSS, csak aláírás
 - RSA a fixed Diffie–Hellmanhoz; csak hitelesítésre használható az RSA-val aláírt tanúsítvány
 - DSS fix Diffie–Hellmanhoz; ismét csak hitelesítésre használható
- a 2. fázis utolsó kötelező üzenete a `server_done`, amelyet a szerver küld el, hogy jelezze a kapcsolódó üzenetek végét, majd várja a kliens válaszát

A TLS Handshake Protokoll

A 3. fázisban a **kliens hitelesítésére és a kulcscserére** kerül sor:

- a `server_done` kézhezvételekor a kliensnek ellenőriznie kell, hogy a szerver tanúsítványa érvényes-e, és hogy a `server_hello` paraméterek elfogadhatóak-e
- ha minden rendben, a kliens egy vagy több üzenetet küld vissza a szervernek
- ha a szerver kéri a tanúsítványát, akkor a `certificate` üzenetet küldi vissza, utána pedig a `client_key_exchange` üzenet következik, amelynek tartalma a kulcscsere típusától függ:
 - RSA: a kliens egy 48 bájtos *pre-master secret* értéket állít elő, amelyet titkosít a szerver publikus kulcsával vagy az ideiglenes RSA kulccsal
 - ephemeral vagy anonymous Diffie–Hellman: elküldésre kerül a kliens publikus Diffie–Hellman paramétere
 - fixed Diffie–Hellman: A kliens publikus Diffie–Hellman paraméterei a tanúsítványa üzenetben már elküldésre kerültek, ezért ennek az üzenetnek a tartalma nulla.

A TLS Handshake Protokoll

A 3. fázisban a kliens hitelesítésére és a kulcscserére kerül sor (folytatás):

- a kliens küldhet egy `certificate_verify`-t, amely az előző üzeneteinek egy aláírt értéke, amelyet a hiteles privát kulcsával kell aláírjon
- a kliens és a szerver az átküldött random számok és adott esetben a *pre-master secret* értéket felhasználva kiszámolják a közös titkot, a **master-secret** értékét

A 4. fázis a kommunikáció lezárásához szükséges üzenetváltásból áll.

SSL/TLS támadások

A TLS megjelenése óta számos sikeres támadás jelent meg \Rightarrow módosították a titkosítási eszközöket, az implementációt, stb. A támadásokat négy kategóriába sorolhatjuk:

- a **handshake protokoll elleni támadások**: 1998-ban sikeres támadás a handshake protokoll ellen, amely az RSA titkosítási séma támadásán alapult
- a **record- és application-protokollok** elleni támadások: 2011-ben Thai Duong és Juliano Rizzo bemutatták a BEAST-et (Browser Exploit Against SSL/TLS). A támadás a választott nyíltszöveg támadás (chosen-plaintext attack) egy gyakorlati megvalósítása. Ugyanez a csapat 2012-ben a CRIME-ban (Compression Ratio Info-leak Made Easy) bemutatták, hogy miként tudja egy támadó visszaállítani a webes cookie-k tartalmát, ha az adattömörítést TLS-sel együtt használták.

SSL/TLS támadások

- a **PKI elleni támadások**: az X.509-tanúsítványok érvényességének ellenőrzése számos támadásnak van kitéve (nem csak a SSL/TLS-ben). Közismert, hogy az OpenSSL, a GnuTLS, a JSSE, az ApacheHttpClient, a Weberknecht, a cURL, a PHP, a Python és az ezekre a termékekre épülő alkalmazások forráskódjának számos hiányossága létezik.
- **egyéb támadások**: a The Hackers Choice német hackercsoport által 2011-ben bejelentett egy DoS támadást, amely megterhelte a szervert handshake kérések sokaságával. A handshake kérelmek feldolgozása a szerveren történik: szerver kényszerítve van, hogy új random értékeket határozzon meg, új kulcsokat generáljon.
- számos SSL/TLS támadás és ellenintézkedés látott napvilágot
- nincs "tökéletes" protokoll és nincs "tökéletes megvalósítás"
- a fenyegetések és az ellenintézkedések határozzák meg a biztonságos internet alapú protokollok fejlődését