Luku V: Verkkojen tietoturva I

Luvussa käsitellään seuraavien protokollien haavoittuvuuksia:

- ARP,
- IP,
- TCP,
- Palvelunestohyökkäykset protokollia vastaan.

ARP-huijaus I

- ARP (Address Resolution Protocol) on linkkikerroksen protokolla, jooka antaa palveluja verkkokerrokselle.
- Verkkokerroksella lähettävä kone tuntee IP-osoitteen. Koska lähetys delegoidaan linkkikerrokselle, lähettävän koneen täytyy selvittää kohdekoneen MAC-osoite. ARP tekee tämän lähettämällä yleislähetyksen paikallisverkkoon kysyen MAC-osoitteita.
- ARP-tiedustelu on muotoa

  "Kenellä on IP-osoite 192.168.1.105?"

  Tämä kysymys lähetetään paikallis-
  verkon kaikille koneille. IP-osoitteen 192.168.1.105 omaava kone vastaa

  "192.168.1.105 is at 00:16:B7:29:E4:7D."

  Vastaus lähetetään tiedustelun tehneelle koneelle.

ARP-huijaus II

- ARP:issa ei ole mukana minkäänlaista todennusta. Mikä tahansa kone voi väittää, että sillä on kysytty IP-osoite. Tämä mahdollistaa huijauksen.
- Oletetaan, että Alice lähettää kyselyn ja IP-osoite kuuluu Bobille. Bobin sijasta Eve lähettää vastauksen Alicelle omalla MAC-osoitteellaan. Eve lähettää lisäksi ARP-vastauksen Bobille yhdistäen Alicen IP-osoitteen omaan MAC-osoitteeseen. Tätä kutsutaan nimellä *ARP-käteismuistin myrkytys*. Itse asiassa on ARPin ominaisuus, että jokainen kone, joka saa ARP-vastauksen, päivittää ARP-käteismuistia, vaikka kone ei olisikaan tehnyt ARP-tiedustelua.
- Näiden valmistelujen jälkeen kaikki liikenne Alicen ja Bobin välillä kulkee Even kautta. Eve voi tarkkailla liikennettä passiivisesti tai hän voi yrittää muuttaa paketteja.

ARP-huijaus III

- Tällaisen hyökkäyksen johdosta paikallisverkkojen suojaukseen on kiinnitettävä huomiota. Tähän on useita menetelmiä. Eräs menetlmä on tarkastella, esiintyykä sama MAC-osoite useaan kertaan.
- Toinen ratkaisu perustuu staattisiin ARP-tauluihin. Tällöin ylläpitäjät päivittävät ARP-käteismuistia manuaalisesti. Automaattiset päivitykset on estetty.
- Kolmantena mahdollisuutena on käyttää ohjelmallisia ratkaisuja. Nämä ohjelmat tutkivat jokaisen ARP-paketin ja vertaavat niiden sisältöä talletettuihin ARP-tietoihin. Esimerkkejä ohjelmista ovat *anti-arpspoof*, *Xarp* ja *Arpwatch*.

IP-huijaus I

- Jokaisessa IP-paketissa on lähteen ja kohteen IP-osoite. Reititysprotokollat eivät tarkista koskaan lähdeosoitetta ja sen voi helposti muuttaa.
- Jos hyökkääjä muuttaa lähdeosoitetta, hän ei saa vastauspaketteja. Esimerkiksi palvelunestohyökkäyksissä tämä ei ole tarpeenkaan. Toisaalta on olemassa keinoja saada vastauspaketteja. Näitä käsitellään TCP-istunnon kaappausten ja palomuurien yhteydessä.
- IP-huijausta ei voida estää, mutta usein se voidaan huomata. Reitittimet voivat hylätä paketit, jotka tulevat ulkopuolelta, mutta joiden lähdeosoite on sisäverkossa. Samoin sisäverkosta lähtevät paketit, joiden lähdeosoite on ulkopuolella, voidaan hylätä.
- Vielä on mahdollista käyttää IP-jäljitystä. Siinä paketin kulkua voidaan seurata taaksepäin sen todelliseen lähettäjään asti. Sen jälkeen voidaan esittää pyyntö tämän polun autonomisille systeemeille estää pakettien kulku lähettäjältä.

Pakettien sieppaus I

- Useimmat Internetissä kulkevat datapaketit eivät ole salakirjoitettuja, joten on mahdollista seurata liikennettä ja tutkia pakettien sisältö.
- On useita keinoja estää pakettien sieppaus. Ensimmäinen toimenpide on tietenkin varmistaa, etteivät ulkopuoliset pääse yksityiseen verkkoon.
- Tooiseksi, jos käytetään Ethernet-kytkimiä (switch) keskittimien (hub) sijasta, niin se vähentää hyökkääjän kanssa samassa segmentissä olevien koneiden määrää. Langattomissa verkoissa ei kuitenkaan ole kytkimiä vastaavia ratkaisuja.
- On myös mahdollista havaita, ovatko verkkolaitteet sekamoodissa (promiscuous mode). Verkkolaite on sekamoodissa, jos se vastaanottaa kaikki paketit, nekin, joita ei ole sille tarkoitettu.

Pakettien sieppaus II

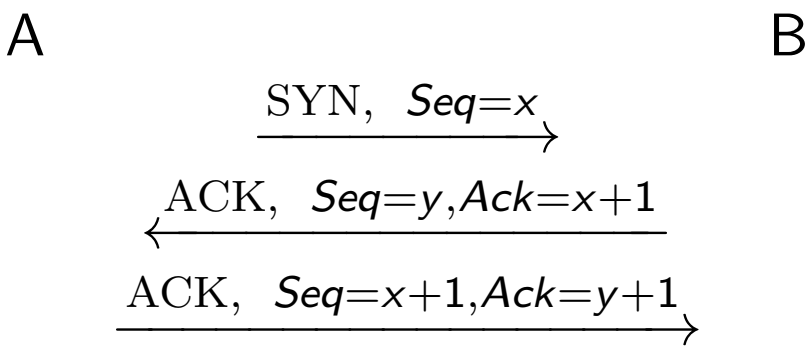
- Käytännössä sekamoodin havaitseminen on kuitenkin hankalaa. Eräs tekniikka on ottaa huomioon vastauksiin kuluva aika. Jos laite ottaa vastaan kaikki paketit, laitteen KJ prosessoi paljon enemmän kuin jos paketteja jätettäisiin huomiotta. Siten vastaukset laitteelta viivästyvät hieman.
- Toinen tekniikka on lähettää epäkelpoja paketteja verkkolaitteille ja seurata, minkälaisia vastauksia tulee takaisin. Esim. jos lähetetään paketti epäillyn koneen IP-osoitteella, mutta väärällä MAC-osoitteella, niin verkkolaite normaalisti hylkää tällaisen paketin, mutta sekamoodissa toimiva laite saattaa lähettää vastauksen.
- Huolimatta varo- ja vastatoimenpiteistä pakettien sieppaus jää ongelmaksi, jota ei pidä aliarvioida. Salausta tulisi käyttää sovellustasolla pakettien sieppausten estämiseksi, varsinkin jos välitetään vähänkään arkaluonteista tietoa.

Pakettien sieppaus III

- Esim. www-liikenne sisältää HTTP-paketin sovellustasolla, joka kapseloidaan TCP-pakettiin kuljetuskerroksessa ja edelleen IP-pakettiin verkkokerrokosessa sekä vielä lopuksi sopivaan linkkitason pakettiin kuten Ethetnet-pakettiin tai langattoman 802.11-verkon pakettiin. Mikäli salausta ei käytetä, pakettien sieppaaja voi seurata kaikkea HTTP-liikennettä. Sen sijaan käytettäesssä HTTPS-protokollaa välitettävä tieto salakirjoitetaan.

TCP-istunnon kaappaus I

- Ensimmäinen kaappausversio on itse asiassa huijaus, sillä siinä luodaan valeistunto. Palautetaan ensin mieleen TCP:n kolminkertainen kättely, jonka avulla perustetaan yhteys kahden osapuolen välille:



- TCP:n järjestysnumeron ennustamishyökkäyksessä tunkeutuja yrittää arvata palvelimen lähettämän ensimmäisen järjestysnumeron. Mikäli arvaus on oikea, on mahdollista luoda tekaistu TCP-istunto.

TCP-istunnon kaappaus II

- Ensimmäiset TCP-toteutukset implementoivat järjestysnumerot käyttäen yksinkertaista laskuria, jota kasvatettiin yhdellä jokaisen lähetyksen yhteydessä. Koska satunnaisuutta ei käytetty, oli triviaalia arvata seuraava järjestysnumero. Nykyiset toteutukset käyttävät pseudosatunnaislukugeneraattoreita, mikä tekee arvaamisesta vaikeaa, muttei mahdotonta. Hyökkäys voi edetä seuraavasti:
  1. Hyökkääjä tekee palvelunestohyökkäyksen asiakasta vastaan estääkseen asiakasta sekaantumasta hyökkäykseen.
  2. Hyökkääjä lähettää SYN-paketin palvelimelle väärentäen lähdeosoitteen asiakkaan osoitteeksi.
  3. Odotettuaan sen verran, että palvelin ehtii lähettää vastauksen asiakkaalle, hyökkääjä päättää kättelyn lähettämällä ACK-paketin, jossa järjestysnumero on arvattu. Huomattakoon, etti hyökkääjä saa palvelimen asiakkaalle lähettämää vastauspakettia, joten seuraava järjestysnumero täytyy päätellä muusta, aikaisemmin kerätystä tiedosta.
  4. Hyökkääjä voi nyt lähettää pyyntöjä palvelimelle asiakkaan nimissä.

TCP-istunnon kaappaus III

- Tämä hyökkäys sallii vain yhdensuuntaisen kommunikoinnin johtuen väärennetystä IP-osoitteesta. Tällainen hyökkäys voi kuitenkin olla hyödyllinen, kun tavoitteena on kiertää suojauksia, jotka perustuvat IP-osoitteisiin. Hyökkäystä kutsutaan sokeaksi solutukseksi (blind injection). On tietenkin mahdollista, että palvelupyynnöillä saadaan aikaan yhteys palvelimen ja hyökkääjän välille.

ACK-myrsky I

- Sokean solutuksen sivuvaikutuksena asiakas ja palvelin voivat ajautua epäsynkroniseen tilaan. TCP:ssä on mekanismi, joka pyrkii uudelleen synkronoimaan asiakkaan ja palvelimen. Mekanismi ei kuitenkaan siedä sokean solutuksen aiheuttamia häiriöitä synkronoinnissa. Siten voi sattua, että asiakas ja palvelin lähettävät toisilleen ACK-sanomia yrittäen saada toisen lähettämään oikeita järjestysnumeroita.
- Tämä edestakainen kommunikointi tunnetaan nimellä ACK-myrsky. Se voi jatkua niin kauan, kunnes toinen paketeista katoaa sattumalta tai palomuuri havaitsee myrskyn ja hylkää virheellisen ACK-sanoman.

Istunnon täydellinen kaappaus I

- Kun hyökkääjä on samassa verkkosegmentissä kuin kohteena oleva palvelin tai asiakas, hyökkääjä voi kaapata täydellisesti TCP-istunnon. Kaappaus on mahdollinen, koska hyökkääjä voi selvittää järjestysnumerot sieppaamalla paketteja. Lisäksi hän voi tehdä ARP-hyökkäyksen, jolloin kaikki liikenne asiakkaan ja palvelimen välillä kulkee hänen kauttaan. Hyökkäys etenee seuraavasti:

1. Client $\xrightarrow{\text{Source A, Dest B, } Seq{=}x}$ Server
2. Client $\xleftarrow{\text{Source B, Dest A, } Seq{=}y,\ Ack{=}x{+}1}$ Server
3. ARP-hyökkäys, liikenne kaappaajan kautta
4. Attacker $\xrightarrow{\text{Source A, Dest B, } Seq{=}x{+}1,\ Ack{=}y{+}1}$ Server

- Kaappauksen estämiseksi tulisi käyttää todennusta ja salausta joko IP-tasolla (IPsec) tai sovelluskerroksessa. Lisäksi ww-palvelujen tulisi välttää istuntoja, jotka aloitetaan vahvalla todennuksella, mutta jotka siirtyvät tämän jälkeen salaamattomaan tiedon välitykseen.

Esimerkkejä erilaisista palvelunestohyökkäyksistä

- Nämä hyökkäykset pyrkivät joko ajamaan alas verkon, tietokoneen tai prosessin tai muuten vaikeuttamaan palvelun tarjontaa.
- Seuraava lista ei ole täydellinen, sillä uusia palvelunestohyökkäyksiä keksitään lähes päivittäin. Tässä listassa on lähinnä sellaisia palvelunestohyökkäyksiä, jotka ovat esiintyneet aikaisemmin, jopa kauan sitten, ovat hyvin tunnettuja ja perustuvat TCP/IP-pinon heikkouksiin.

Maahyökkäys

- Maahyökkäyksessä (Land Attack) konstruoidaan TCP SYN -paketti, jossa kohde- ja lähdeosoite ovat samoja.
- Joissakin vanhoissa järjestelmissä tällainen paketti aiheutti vastaanottajan puolella lukkiuman, jonka johdosta kone täytyi käynnistää uudelleen. Hyökkäykseen tarvitaan siis vain yksi ainoa paketti.
- Hyökkäys ei liene kovin relevantti nykyisissä ympäristöissä, mutta ohjelmistojen uusiokäytön ja mahdollisten koodausvirheiden vuoksi se saattaa taas tulevaisuudessa putkahtaa esiin.
- Menetelmä on tyypillinen palvelunestohyökkäyksissä. Outo tai mahdoton paketti konstruoidaan ja lähetetään. Vastaanottopäässä tällainen paketti aiheuttaa hämmennystä, joka johtuu joko toteutusvirheestä tai protokollan epätäydellisyydestä.

Neptunus I

- Neptunus tai SYN-tulvitus käyttää hyväkseen sitä tietoa, että jokaista puoliksi avoinna olevaa TCP-yhteyttä kohti tcpd (tcp-demoni) luo tietueen, jossa pidetään yhteystietoja.
- Jos yhteyttä ei luoda loppuun asti tietyn ajan kuluessa, yhteys katkaistaan ja tietueen varaama tila vapautetaan.
- Jos kuitenkin riittävä määrä yhteyksiä alustetaan ennen kuin ajastin laukeaa, tietorakenne vuotaa yli. Se aiheuttaa taulukon ylivuodon (segmentation fault) ja tietokoneen lukkiintumisen.
- Tässä hyökkäyksessä konstruoidaan paketti, jonka IP-lähdeosoite on saavuttamaton. Toisin sanoen mikään kone ei vastaa SYN/ACK-sanomaan, jonka kohdekone lähettää paketin saatuaan. Siten yhteys jää auki.

Neptunus II

- Eräs mekanismi estää SYN-tulvitus perustuu Daniel Bernsteinin v. 1996 ehdottamiin SYN-evästeisiin. Tällöin yhteyden katkaisemisen sijasta lähetetään eväste ilman yhteyden kirjaamista muistiin. Palvelimen pitäisi lähettää normaalisti ACK-viesti, jossa on muun muassa järjestysnumero. Järjestysnumero voi olla mikä tahansa palvelimen valitsema numero ja sen avulla kootaan myöhemmin tulevat paketit. Eväste on itse asiassa huolellisesti suunniteltu järjestysnumero, joka koostuu seuraavista kentistä: Evästeessä on seuraavia tietoja:
  - aikaleima: 5 bittiä, ilmaisee ajan mod 32 minnuuttia.
  - maksimaalinen segmenttikoko: 3 bittiä
  - MAC-arvo: 24 bittiä, lasketaan lähteen ja kohteen IP-osoitteesta, porttinumeroista ja aikaisemmasta aikaleimasta.

Neptunus III

TCP:n määrityksen mukaan asiakkaan täytyy vastata järjestysnumerolla, joka on yhtä suurempi kuin edellinen järjestysnumero. Kun asiaks vastaa ACK-paketilla, palvelin vähentää 1:n saadaksen edellinen järjestysnumeron. Sen jälkeen palvelin vertaa paketin aikaleimaa nähdäkseen, onko yhteys jo rauennut. Seuraavaksi palvelin vertaa laskemaansa MAC-arvoa ACK-paketissa tulleeseen. Lopuksi lukee keskimmäiset 3 bittiä, joita käytetään vietäessä yhteys SYN-jonoon.

- SYN-evästeet implementoitiin ensin Linuxissa eikä niitä ollut Windowsissa ainakaan muutama vuosi sitten. Nykyään SYN-evästeitä ei välttämättä enää suositellakaan:
  - Maksimaalinen segmenttikoko voi olla vain 8, koska tämän enempää ei voi esittää 3:lla bitillä.
  - SYN-evästeet eivät salli käyttää TCP:n valinnaista kenttää, koska tämä tieto on yleensä viety SYN-jonon tietueisiin.

Neptunus IV

Linux yrittää korjata jälkimmäisen puutteen sijoittamalla TCP:n valinnaisen tiedon TCP-paketin aikaleimakenttään. Kuitenkaan kaikkia tärkeitäkään optioita ei voi esittää 8:lla bitillä, joten SYN-evästeet eivät sovi kaikkiin tilanteisiin.

- Muita keinoja on varata oma jononsa puolittain avatuille yhteyksille. Tällöin ei varata muita resursseja TCP-yhteydelle enenkuin ACK on vastaanotettu. Tätä tekniikkaa sovelletaan Windowsissa.

Optimistinen TCP-hyökkäys I

- TCP:n ruuhkan torjunta käyttää liukuvaa ikkunaa. Ikkunassa on TCP-paketteja, joille ei ole vielä tullut kuittauksia. Kun kuittaus saapuu, ikkunaa kasvatetaan, ja kavennetetaan, kun segmentit tulevat väärässä järjestyksessä tai eivät saavu ollenkaan.
- Optimistinen TCP ACK -hyökkäys on hyökkäys ruuhkan torjuntaa vastaan. Hyökkäävä asiakas yrittää saada palvelinta kasvattamaan lähetystiheyttä kunnes palvelin ei enää pysty lähettämään kaistanleveyden rajoitusten tähden. Tästä kärsivät muut yhteydet. Jos hyökkäys tehdään samanaikaisesti useita palvelimia vastaan, tuloksena voi olla koko Internetin laajuinen ruuhka.
- Hyökkäys perustuu siihen, että kuittauksia lähetetään paketteihin, jotka on lähetetty mutta jotka eivät ole vielä tulleet perille. Tämän johdosta palvelin nopeuttaa lähetyksiä.

Optimistinen TCP-hyökkäys II

- Hyökkäys on esiintynyt harvoin käytännössä. Se perustuu TCP-protokollan rakenteeseen ja protokollatason tason torjunta vaatisi TCP:n uutta suunnittelua. Käytännössä voidaan asettaa lähetystiheydelle yläraja asiakasta kohden ja tukkia liikenne asiakkailta, jotka näyttävät tekevän palvelunestohyökkäystä.

Ping O'Death

- Tämä hyökkäys koostuu ICMP echo-pyynnöstä (ping), jossa on liian pitkä (yli 64 KB) "hyötykuorma". Vanhemmat käyttöjärjestelmät lukkiintuvat tai käynnistyvät uudelleen, kun puskuri, johon paketti varastoidaan, vuotaa yli.
- Ensimmäiset Windows 95 -versiot sisälsivät ping-ohjelman, joka salli käyttäjän määritellä paketin koon, vaikka se olisikin ollut liian pitkä. Tämä teki järjestelmän suosituksi hyökkäyskohteeksi.
- Samoin kuin maahyökkäyksessä tämäkin hyökkäys vaatii vain yhden paketin. Nykyiset käyttöjärjestelmät eivät ole enää haavoittuvia tälle hyökkäykselle.

Prosessitaulu I

- Prosessitauluhyökkäys kehitettiin MIT:n Lincoln DARPA-laboratorioissa testaamaan tunkeutumisen estojärjestelmiä. Tavoitteena oli kehittää uusia hyökkäyksiä, jotta nähtäisiin, voitaisiinko ne paljastaa.
- Hyökkäyksen idea perustuu siihen, että joka kerran kun TCP-yhteyspyyntö saapuu, prosessi haarautuu (fork). Jos pyyntöjä saapuu hyvin paljon, prosessitaulu täyttyy. Kun taulu on täynnä, uusia prosesseja ei voida luoda. Seurauksena on tilanne, jossa tietokone ei voi tehdä mitään.
- Tämä hyökkäys täytyy kohdistaa prosesseille, jotka sallivat monia yhteyksiä yhtäaikaa. Esimerkiksi sendmail ei hyväksi uusia yhteyksiä, jos niitä on jo ennestään paljon.

Prosessitaulu II

- Sen sijaan finger sallii aina uusia yhteyksiä. Jotkut aikaisemmat fingerin versiot eivät käyttäneet ajastinta yhteyksien sulkemiseen, vaan ne jäivät auki niin pitkäksi aikaa, kunnes toinen sulki ne. Jokainen yhteys sai oman tunnuksensa ja jos riittävästi yhteyksiä avattiin, prosessitaulu täyttyi.

Targa3

Targa3-hyökkäyksessä lähetetään laittomia paketteja kohteelle. Nämä virheelliset paketit saavat jotkut systeemit kaatumaan. Vaikka systeemi ei kaatuisikaan, virheelliset paketit kuluttavat tavallista enemmän resursseja. Tyypillisesti paketeissa on vialla jotain seuraavista:

- Virheellinen paloittelu, protokolla, koko tai IP-otsakkeen arvo.
- Virheelliset optiot.
- Virheelliset TCP-segmentit.
- Virheelliset reitityslipput.

Smurf-hyökkäys

- Smurf-hyökkäyksessä on kolme osapuolta: hyökkääjä, kohde ja välittäjä, joka höynäytetään tekemään varsinainen hyökkäys.
- Hyökkääjä konstruoi echo request -paketteja, joissa on lähteenä aiottu kohde ja kohteena välittäjä. Nämä paketit yleislähetetään, jotta maksimoidaan vastausten lukumäärä.
- Kaikki koneet välittäjän verkossa vastaavat kohteelle, joka ei voi käsitellä niin suurta määrää paketteja. Kohde kaatuu tai ainakin hidastuu siinä määrin, ettei kykene toimimaan normaalisti.
- Hyökkääjä ei näy kohteen lokitiedostoissa. Hyökkääjä paljastuu vain välittäjän verkon lokeista.

Teardrop-hyökkäys

Teardrop perustuu siihen, että jotkut vanhat TCP/IP-toteutukset eivät käsittele asianmukaisesti tilannetta, jossa paketti on pilkottu, mutta palaset eivät ole erillisiä. Jos koneessa on tämän vian sisältävä TCP/IP-toteutus ja hyökkääjä lähettää paketin, joka näyttää oikealta, mutta jonka palaset eivät ole erillisiä, kone kaatuu.

UDP-tulva

- UDP-tulva saa aikaan, että kaksi konetta hyökkäävät toisiaan vastaan. On tietty määrä portteja, jotka vastaavat paketilla saatuaan paketin. Echo (portti 7) ja chargen (portti 19) ovat tällaisia. Echo kaiuttaa paketin takaisin, kun taas chargen generoi merkkivirran.
- Tarkastellaan UDP-pakettia, jolla on lähdeporttina 7 ja kohdeporttina 19. Paketti generoi joitakin merkkejä kohdekoneesta, jotka merkit lähetetään lähdekoneen echo-porttiin.
- Lähde kaiuttaa nämä paketit takaisin, mikä puolestaan aiheuttaa lisää paketteliikennetta koneiden välille jne. Jossain vaiheessa molemmat koneet kuluttavat kaiken aikansa lähettämällä paketteja edestakaisin kunnes toinen tai molemmat kaatuvat.
- Jos ulkopuolelta tulee paketteja, joiden lähdeosoite on sisäpuolella, palomuurin pitäisi pysäyttää nämä paketit. Tällaiset paketit ovat melkein varmasti väärennettyjä. Jos kysymyksessä ei olisikaan hyökkäys, tällaiset paketit ovat merkki siitä, että jotain on mennyt vikaan.

Palvelunestohyökkäysten estäminen I

- Palvelunestohyökkäyksessä vastapuoli estää laillisia käyttäjiä käyttämästä protokollaa täydellisesti.
- Käytännössä palvelunestohyökkäyksiä tehdään sellaisia palvelimia vastaan, joiden on tarkoitus palvella asiakkaita verkon yli.
- Nämä hyökkäykset voidaan jakaa hyökkäyksiin, jotka pyrkivät kuluttamaan palvelimen laskentaresurssit (resource depletion attacks), ja hyökkäyksiin, joiden tavoitteena on ylittää sallittu yhteyksien määrä (connection depletion attacks).
- Periaatteellisella tasolla näyttää siltä, ettei palvelunestohyökkäyksiä voida täysin estää. Jokainen yhteysyritys aiheuttaa sen, että joko yhteys hyväksytään tai se hylätään tietyn laskentamäärän jälkeen.
- On kuitenkin joitakin menetelmiä, joista on hyötyä pienennettäessä estohyökkäysten vaikutuksia. Toiset protokollat ovat haavoittuvaisempia estohyökkäyksille kuin toiset, joten hyökkäys ja sen torjuntamenetelmiä on aiheellista tuntea.

Auran ja Nikanderin ehdotus I

- Aura ja Nikander ovat ehdottaneet tilattomia yhteyksiä suojaamaan yhteyksien ylittymiseltä ([**?**]).
- Ideana on vaatia asiakasta tallentamaan kaikki tilatieto palvelimen sijasta ja palauttamaan tieto palvelimelle sanomien yhteydessä. Tällä tavoin palvelin säästyy tietojen varastoinnilta.
- On selvää, että palvelimelle palaavan tilatiedon tulee olla tarkistettavissa ja sen täytyy olla todennettavissa.
- Sen tulisi olla myös luottamuksellista. Tämä lisää prosessesointia kummassakin päässä.

Evästeet I

- Käytännöllisempi tapa on käyttää evästeitä. Tätä tapaa ovat ensin ehdottaneet Karn ja Simpson Photuris-protokollassaan (1. versio 1995, viimeisin 1999, RFC 2522).
- Kun asiakas yrittää solmia yhteyttä, palvelin lähettää takaisin evästeen. Menetelmä on samanlainen kuin www-palvelimissa käytetty, mutta nyt evästeillä on erityinen muoto: ne on muodostettu vain palvelimen tuntemasta salaisuudesta ja yhteystiedoista.
- Tässä vaiheessa palvelin ei talleta mitään yhteystietoja itselleen. Asiakkaan täytyy lähettää eväste seuraavassa sanomassa ja palvelin tarkistaa evästeen oikeellisuuden lähetetystä datasta ja evästeen sisältämästä salaisuudesta.
- Menetelmän tavoitteena on varmistaa, että asiakas todella aikoo solmia aidon yhteyden.

Evästeet II

- Menetelmä estää sellaiset palvelunestohyökkäykset, joissa yhteyspyyntöjä lähetetään satunnaisesti suuri määrä. Jos palvelimen salaisuutta vaihdetaan aika ajoin (joka 60. sekunti), ei edes sama asiakas voi tehdä rajoittamatonta määrää yhteyspyyntöjä.

Meadowsin ehdotus I

- Meadows ehdottaa yhteyksiin kohdistuvan hyökkäyksen torjumiseksi, että jokainen sanoma tulisi todentaa ([**?**]).
- Jotta turhaa laskentaa ei tulisi liikaa, todennuksen tulisi olla kevyttä protokollan alussa ja vahvistua myöhempien sanomien myötä.
- Evästeet, jotka palvelin lähettää ja jotka täytyy palauttaa, voivat alussa toimia todennuksena.
- Meadows on kehittänyt jopa formaalin kehyksen, joka perustuu Gongin ja Syversonin fail-stop -protokollille. Tällaiset protokollat lopettavat suorituksensa heti, kun epäaito sanoma on havaittu.

Juelin ja Brainardin ehdotus I

- Juels ja Brainard ehdottavat mekanismia, jota he kutsuvat nimellä client puzzles ([**?**]). Mekanismi muodostaa vahvemman todennuksen kuin evästeet.
- Ideana on, että kun palvelimen kuormitus kasvaa suureksi, kenties estohyökkäyksen johdosta, palvelin lähettää asiakkaille jonkin verran laskentaa vaativan probleeman, joka asiakkaan täytyy ratkaista ennen kuin uusi yhteys tehdään.
- Todellisille asiakkaille tästä aiheutuu vain vähän vaivaa, mutta estohyökkäyksen tekijä joutuu ratkomaan monia probleemoja.

HIP

- Esimerkkinä modernista protokollasuunnittelusta voidaan mainita Host Identity Protocol eli HIP. Siinä osapuolet identifioidaan julkisten avainten avulla. Samoin lähetykset todennetaan digitaalisten allekirjoitusten avulla. Lisäksi palvelunestohyökkäysten varalta palvelimen ensimmäinen vastauspaketti kuluttaa mahdollismman vähän resursseja ja mahdollisen palvelunestohyökkäyksen jatkaminen vaatii hyökkääjältä enemmän resursseja.
- HIP on uusi väliohjelmisto. Se muodostaa yhteyden ilman TCP:tä ja yhteyden avauksen perusteella voidaan samalla perustaa IPsec-yhteys IP-tasolla. Sen jälkeen sovelluskerroksen pakettien välitys sujuu normaalisti esimerkiksi TCP:n kautta. HIP ei ole vielä saanut paljon käyttöä, vaikka RFC-standardit ovatkin valmiit.