Certificati Digitali e sicurezza di SSL/TLS

Nei precedenti articoli abbiamo descritto il protocollo SSL/TLS, in particolare la versione TLSv1.0, ed accennato ad alcune sue caratteristiche dal punto di vista della sicurezza. In questo articolo consideriamo l'aspetto più problematico del protocollo, meglio della infrastruttura nel quale è previsto il suo impiego, ovvero l'autenticazione fra le parti. Vogliamo sottolineare come gli aspetti che discuteremo non fanno in realtà parte esplicitamente del protocollo come descritto nel RFC, ma che costituiscono l'ambiente e l'uso comune del protocollo. Quello che faremo è di considerare quindi se ed in che modo il protocollo crittografico permette di ottenere la sicurezza che si prefigge per l'utente finale in alcune particolari situazioni.

L'autenticazione di un server Web

Per semplicità , essendo poi anche il caso più comune, consideriamo solo il caso di un client che vuole sincerarsi dell'autenticità del server, ma non il viceversa. Come abbiamo visto nei precedenti articoli, il protocollo richiede solo che il server fornisca un certificato *valido*, con il che si intende che dimostri di possedere la chiave segreta corrispondente alla chiave pubblica presente nel certificato. Se ci limitassimo solo a questo, sarebbe molto facile fare attacchi di impersonificazione o *Man-in-the-middle* (MITM). Se il client accettasse qualunque certificato valido, non sarebbe in grado di distinguere il server vero da un server truffaldino, nel caso in cui anche quest'ultimo presentasse un certificato valido. SSL/TLS non risolve questo problema, lascia che sia l'applicazione (browser web, client di posta elettronica eccetera) a decidere se accettare o meno il certificato presentato dal server. L'organizzazione e la gestione dei certificati digitali, delle CA eccetera, è descritta formalmente dalle *Public Key Infrastructure* (PKI). In questa sede non entreremo in dettaglio nella descrizione dell'organizzazione delle PKI, anche perché non ne avremo bisogno, ma ci limiteremo a vedere come i certificati digitali possono essere utilizzati da client e server.

Ricordiamo che possiamo dividere i certificati digitali in due grandi categorie:

1. quelli *self-signed*, ovvero firmati con la stessa chiave privata associata alla chiave pubblica presente nel certificato
2. quelli firmati da un ente certificatore, detto comunemente CA.

Per contro, cercando di riassumere a grandi linee, l'applicazione del client (nel nostro caso un web browser) ha le seguenti opzioni su come procedere per accettare o meno un certificato:

1. *accettare tutti i certificati validi*: ovvero per cui il server dimostri il possesso[1] della corrispondente chiave privata, per questo basta che il server firmi digitalmente qualche dato e che il client verifichi la firma con la chiave pubblica del server
2. *accettare solo dei certificati digitali già conosciuti*: in questo caso il client ha copia dei certificati che accetta e verifica che il server oltre a presentare un certificato valido, presenti un certificato della lista nota
3. *accettare solo i certificati validi firmati da una CA nota*: (opzionalmente con ulteriori vincoli ad esempio sul nome dell'entità intestataria del certificato e simili) ovviamente in questo caso i certificati *self-signed* non sono accettati per definizione.

Nel primo caso non vi è in pratica alcuna autenticazione, i client sono solo in grado di stabilire dei canali di comunicazione sicuri con i server, chiunque essi siano.

Il secondo caso è il più sicuro, ma demanda all'applicazione, ovvero all'utente, l'ottenimento iniziale dei certificati ritenuti autentici. Una procedura spesso utilizzata è la seguente: la prima volta che ci si collega ad un server, il client chiede all'utente se vuole accettare e memorizzare il certificato del server, in caso positivo le volte successive il certificato del server viene confrontato con quello memorizzato[2] e se sono differenti l'applicazione blocca la procedura di autenticazione TLS impedendo la connessione ed avvisando l'utente.

Il terzo caso è in pratica il più comune ma anche il più problematico. Infatti potremmo considerarlo come un caso particolare del precedente se lo vedessimo dal punto di vista di un utente che trasferisce ad un ente terzo, la CA, l'onere ed il compito di validare i server ed i relativi certificati digitali. In altre parole, l'utente *si fida* della CA ed invece di memorizzare i certificati dei singoli server, memorizza i certificati delle CA, validando e quindi accettando i certificati firmati da loro.

Anche in questo caso, quindi, come primo passo l'utente deve caricare i certificati delle CA, ma in pratica questo non è di solito necessario perché i certificati delle CA sono già inclusi nelle applicazioni. Vediamo quindi che c'è un ulteriore anello nella *catena della fiducia*: l'utente *si fida* del fornitore dell'applicazione che *sceglie per lui* le CA di cui l'utente si fiderà. Anche se questo è un parere personale, riteniamo che in questa situazione tutti noi utenti abbiamo perso quasi del tutto il controllo del processo di autenticazione del server, finendo a porre la nostra fiducia in entità spesso a noi del tutto sconosciute.

Attacchi

Cosa può succedere in pratica? Purtroppo gli attacchi, quali il Phishing, sono all'ordine del giorno,

1 Non vi è garanzia a priori che il possesso sia *unico.*
2 Il server è identificato tramite il suo indirizzo web (URL).

ed in questa sede ci limiteremo ad illustrare sommariamente il caso più semplice e comune. L'attaccante prepara un sito graficamente identico a quello della banca di cui si vogliono truffare i clienti, dà al sito un nome simile, ma non identico ovviamente, tale da non destare immediatamente attenzione ed acquista, oppure ruba, per il sito un certificato digitale da una CA riconosciuta dalla maggior parte dei browser, ma magari in un paese lontano.[3] A questo punto l'attaccante invia ai clienti della banca un email nel quale si dice, ad esempio, che la banca per motivi di sicurezza richiede all'utente di cambiare le proprie credenziali di accesso al servizio di Internet Banking e di fare ciò immediatamente seguendo il link riportato nel messaggio. L'ignaro utente segue il link e si ritrova sul sito dell'attaccante, con una connessione sicura con TLS, e nessun avviso poiché il certificato offerto dal sito è firmato da una CA riconosciuta ed è valido. L'utente introduce le proprie credenziali, che ovviamente vengono registrate dall'attaccante, e poi ridiretto sul vero sito della banca. Ovviamente l'attaccante accede con le credenziali rubate, al servizio Internet Banking ed effettua delle operazioni a proprio favore. In questo caso tutto ciò è completamente trasparente per l'utente che, se non è in grado di notare la strana URL contenuta nel messaggio email, che a sua volta può essere camuffata ed offuscata con varie tecniche, non ha modo di rendersi conto di quanto veramente stia succedendo.[4] Nessun avviso o messaggio è segnalato dal browser dell'utente.

Anche se l'attaccante utilizzasse un certificato self-signed od una CA non riconosciuta dall'utente, avrebbe parecchie possibilità di successo. Come abbiamo visto, poiché i web browser accettano automaticamente solo i certificati firmati dalle CA note, un certificato self-signed non è accettato automaticamente ed il web browser chiede all'utente come proseguire. L'utente ha tre possibilità

1. non effettuare la connessione
2. effettuare la connessione ma non memorizzare il certificato, per cui in caso di una successiva connessione il certificato risulterebbe ancora ignoto al browser
3. effettuare la connessione e memorizzare il certificato come valido in futuro.

Quale è la scelta più comune dal punto di vista psicologico di un utente non esperto? Purtroppo è la seconda! I più infatti seguiranno questo ragionamento: provo ad accettare il certificato solo per questa volta, e se tutto va bene la prossima volta lo accetto definitivamente. Ma all'attaccante basta ottenere una volta le credenziali di accesso al servizio di Intrenet Banking per poter eseguire la propria truffa.

Infine l'attaccante potrebbe anche decidere di non proteggere con SSL/TLS il proprio sito, confidan-

3 Ovviamente le CA verificano la titolarità dei nomi per cui generano i certificati, ma sono comunque sempre degli enti commerciali e non autorità statali.

4 L'attaccante può anche utilizzare un vero indirizzo email della banca quale mittente dell'email in quanto non si aspetta e non vuole ricevere alcuna risposta via email.

do nel fatto che l'utente inesperto non si renda conto dell'assenza del canale cifrato (praticamente del famoso lucchetto chiuso in un angolo del browser) sino all'accesso al vero sito della banca.
E' vero che attacchi di questo tipo comprendono una buona dose di Social Engineering; è anche vero però che il sistema TLS+certificati digitali+CA+PKI è percepito dagli utenti finali come una architettura in grado di proteggerli da truffe di questo tipo, cosa che ovviamente non fa e non può fare.
Si noti che in tutto ciò, il protocollo SSL/TLS svolge perfettamente il proprio compito secondo le specifiche. Dal punto di vista di SSL/TLS i certificati sono validi quando il server dimostra di possedere la chiave privata corrispondente a quella pubblica presente nel certificato. Come abbiamo già detto, è responsabilità dell'applicazione, e quindi dell'utente, decidere se il certificato appartiene o meno al sito a cui l'utente vuole connettersi. Poiché l'utente comune non è in grado di fare questa scelta, delega il produttore del software a farla per lui, che a sua volta delega le CA. Non solo, nel caso in cui questa catena non funzioni, certificati self-signed o CA non riconosciute, l'utente è impreparato a scegliere in prima persona. Inoltre va tenuto conto che la presenza del lucchetto chiuso nel browser dà un senso di sicurezza all'utente che rende ancora più pericoloso questo tipo di attacco.

Revoca dei certificati

Un'altra problematica associata alla gestione dei certificati digitali è quella della loro revoca. I certificati digitali hanno di solito una data di scadenza, durano per lo più un anno, ed anche se questa è una buona politica di sicurezza in vista di quanto discuteremo in questa sezione, la motivazione principale per obbligare i gestori dei server a cambiare annualmente i certificati è più commerciale che tecnica. Il problema si pone però se è necessario revocare un certificato prima della sua data di scadenza ad esempio perché è stata trafugata la sua chiave privata. Come fa una applicazione ad accertarsi che un certificato valido dal punto di vista tecnico è stato o meno revocato dalla CA che lo ha emesso? Al giorno d'oggi non vi è una soluzione veramente soddisfacente a questo problema. Sono state proposte principalmente due soluzioni.
La prima consiste nelle cosiddette CRL (Certificate Revocation List). Ogni CA prepara una lista di tutti i certificati da lei emessi e revocati, in un formato simile a quello di un certificato digitale anch'esso firmato, e pubblica questa lista sul proprio sito web. Ogni applicazione, quindi ogni web browser (e non il protocollo TLS), dovrebbe scaricarsi quotidianamente tutte le liste CRL di tutte le CA che conosce. Ovviamente ben pochi lo fanno visto anche il traffico che genererebbe.
L'altra soluzione è il protocollo OCSP (*Online Certificate Status Protocol*) secondo il quale, in tem-

po reale, una applicazione (di nuovo non TLS) al momento della verifica di un certificato emesso da una CA nota, chiede alla CA se quel particolare certificato con quel numero di serie è stato revocato o no. I dati da scambiare sono pochi, in pratica un numero di serie ed un si/no, ma i potenziali ritardi per le attese della risposta in momenti di alto traffico, alta utilizzazione dei server della CA e poca banda dell'utente, possono facilmente portare a tempi di attesa poco apprezzati dall'utente.
In entrambi i casi, l'indirizzo ove il browser web trova la CRL o il responder OSCP, devono essere contenuti nel certificato stesso, ma questo è ovviamente opzionale e ben pochi certificati lo specificano.

Andrea Pasquinucci
pasquinucci@ucci.it

Riferimenti Bibliografici

[1] SSLv3.0: http://wp.netscape.com/eng/ssl3/
[2] TLSv1.0 descritto in RFC-2246
[3] E. Gerck, *Overview of Certification Systems*, http://nma.com/papers/certover.pdf
[4] RSA: http://www.rsasecurity.com/rsalabs/ in particolare gli standard PKCS