Sistemi di Elaborazione: Sicurezza su Reti **Anno Acc. 1998-99**
PROF. ALFREDO DE SANTIS

Password

a cura di Gengo, Pagnotta, Peluso, Pietrafesa, Picariello, Troncone *16 Marzo 1999*

Indice

1 Introduzione

1.1 Un pò di storia

L'esigenza di poter contraddistinguere le entità ha dato vita al problema dell'identificazione. Per risolverlo varie tecniche sono state ideate e sviluppate nel corso degli anni. Il primo metodo fu usato dai messaggeri dei Re che si identificavano tramite anelli con sigillo. Più tardi in Cina fu introdotto il metodo delle impronte digitali. Nel 1870, il sistema Bertillon venne impiegato negli Stati Uniti per identificare i criminali. Questo sistema riconosce una persona considerando un insieme di misure come altezza, apertura delle braccia, lunghezza dell'avambraccio, delle dita, dei piedi e circonferenza della testa. Comunque, nel 1903 accadde un evento che è rimasto famoso: due persone di nome "Willie West" avevano le stesse misure trovate con il metodo Bertillon. Attualmente il metodo più usato è quello delle password che ebbe ampio utilizzo anche durante le guerre. Di seguito parleremo delle password, delle loro caratteristiche, dei possibili schemi di identificazione tramite esse e dei possibili attacchi che gli si possono portare. Tratteremo in particolare dello schema adottato dal sistema Unix.

2 Identificazione debole

I tradizionali schemi di password sono caratterizzati da password stabili nel tempo, che determinano la così detta *autenticazione debole*. L'idea di base è la seguente. *Una password*, associata con ogni utente (entità), è tipicamente una stringa lunga da 6 a 10 caratteri. Essa funge da segreto condiviso tra il sistema e l'utente. (I tradizionali schemi di password rientrano così nella categoria di tecniche a chiave simmetrica che forniscono l'autenticazione unilaterale). Per ottenere l'accesso ad una risorsa del sistema (per esempio, computer account, stampante, o applicazioni software), l'utente inserisce una coppia del tipo <userid, password>; *l'userid* (user identificator) è una dichiarazione di identità, e *la password* è una prova a supporto dell'identificazione. Il sistema controlla che la password abbia una corrispondenza tra i dati memorizzati, confrontando la stringa digitata con la password dell'userid specificato, a quel punto l'identità stabilita è autorizzata ad accedere alla risorsa. L'insieme delle idee presentate nelle seguenti sezioni motiva le decisioni progettuali realizzate in schemi di password tipici. Una sezione successiva riassume gli attacchi standard che questi progetti neutralizzano. Le minacce da cui dobbiamo proteggerci includono: rivelazione di password (al di fuori del sistema) ed intercettazione (nel sistema), entrambe consentono attacchi replay.

2.1 Schemi a password fissa: tecniche

2.1.1 File di password memorizzati

L'approccio più semplice per il sistema è quello di memorizzare in un file di sistema non cifrato, protetto sia in lettura che in scrittura (per esempio attraverso il sistema operativo si accede ad istruzioni privilegiate di controllo), la coppia <userid, password> relativa ad ogni utente. Quando l'utente vuole identificarsi al sistema immette la sua coppia <userid, password>. Il sistema ricerca nel file l'userid inserito, confronta la password con quella memorizzata relativa a tale userid. Se le password sono identiche, l'utente accede al sistema altrimenti viene respinto.

Un inconveniente di tale metodo è che non fornisce alcuna protezione contro gli utenti privilegiati o *superuser* (userid speciali che godono di accessi privilegiati ai file di sistema e alle risorse).

2.1.2 File di password "cifrate"

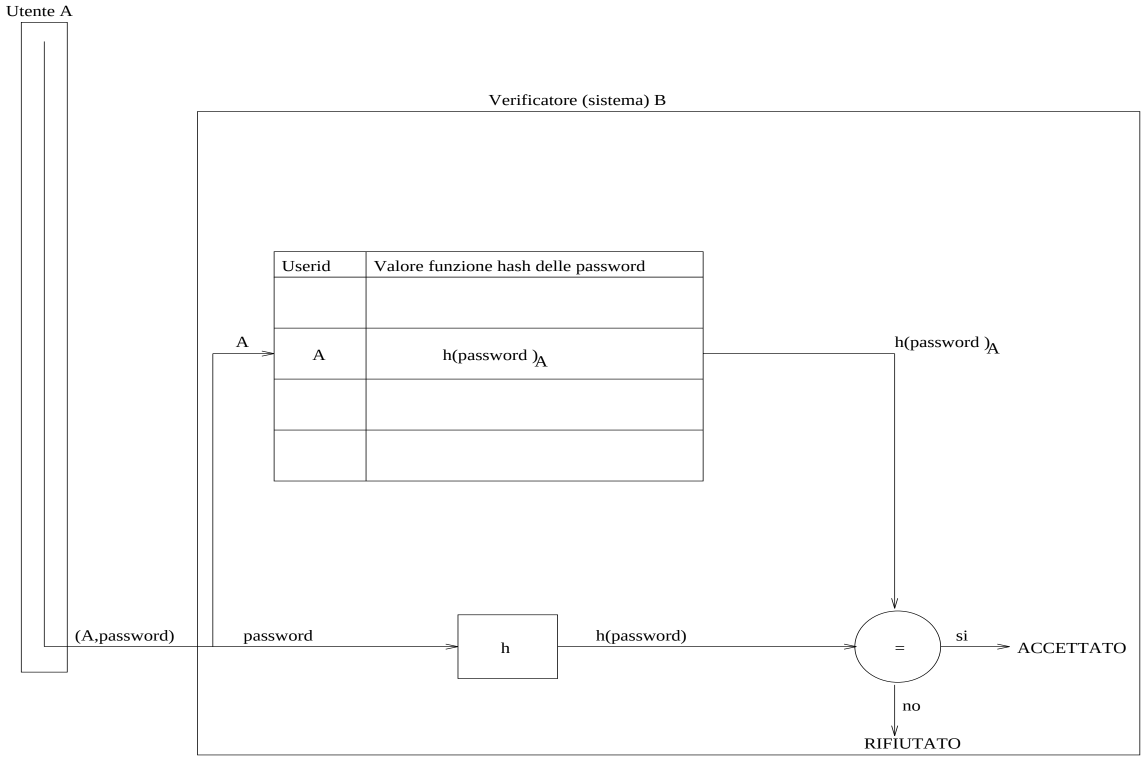

Figura 1: Uso di una funzione one-way per password-checking

Piuttosto che memorizzare la password dell'utente in chiaro in un file, si memorizza una funzione one-way della password (Figura 1). Con questo metodo il sistema memorizza nel file di password una coppia <userid, h(password)>, dove h(password) è una funzione hash della password. L'utente che vuole accedere al sistema inserisce la sua coppia <userid, password>. Il sistema ricerca l'userid nel file, calcola una funzione hash della password inserita e la confronta con quella memorizzata; se sono uguali l'identificazione ha successo, altrimenti viene rifiutata.

Osservazione. (*funzione one-way contro cifratura*) Per proteggere i file delle password, l'uso di una funzione one-way è in genere preferito ad una cifratura reversibile; tra le ragioni vi sono le restrizioni all'esportazione, e la necessità di utilizzare chiavi per la codifica. Comunque, in entrambi i casi, per ragioni storiche, i valori risultanti sono tipicamente chiamati password "cifrate".

2.1.3 Regole della password

Le caratteristiche fondamentali che le password dovrebbero rispettare sono:

- difficili da indovinare;
- facili da ricordare.

In realtà è arduo combinare queste due proprietà, in quanto qualcosa che è facile da ricordare risulta anche prevedibile! Dato che gli attacchi di tipo dizionario hanno successo con le password prevedibili, alcuni semplici accorgimenti adottati nella loro scelta consentono di evitare gli attacchi o comunque renderli più complessi. Alcuni sistemi impongono delle "regole della password" per scoraggiare o impedire l'uso di password "deboli" da parte degli utenti. Regole della password tipiche richiedono un limite inferiore alla lunghezza della password (per esempio, 8 o 12 caratteri); inoltre si richiede che ogni password contenga almeno un carattere di ciascun insieme delle categorie (per esempio, maiuscole, numerici, non-alfanumerici); si controlla che le password candidate non siano presenti in dizionari disponibili on-line e che non siano composte da informazioni relative all'account (userid o sue sottostringhe). In effetti, conoscendo tali regole, un avversario potrebbe usare una strategia di attacco di tipo dizionario modificata tenendo conto delle regole, proponendosi di trovare la più debole forma di password che le soddisfi. L'obiettivo delle regole della password è di incrementare l'entropia (piuttosto che solo la lunghezza) delle password degli utenti per opporsi agli attacchi di tipo dizionario e di ricerca esaustiva. Alcuni aspetti che rendono le password vulnerabili derivano dai criteri di scelta. Infatti le password sono per lo più costituite da:

- parole comuni (in qualsiasi lingua);
- nomi comuni (nomi TV, nomi di musicisti, vezzeggiativi, nomi di amici, membri della famiglia, soprannomi);
- informazioni facili da ottenere (date di nascita, numeri di telefono, codice fiscale, targa della macchina);
- tasti consecutivi di tastiere ("qwerty");
- password su altri sistemi;
- permutazioni di queste lettere (lettere al contrario).

Alcune soluzioni per risolvere il problema della vulnerabilità di una password potrebbero essere:

- mescolare lettere minuscole e maiuscole;
- usare qualcosa di impronunciabile;
- includere caratteri non alfanumerici;
- mescolare lettere e numeri;
- eseguire sostituzioni sistematiche, come o$\to$0 oppure i$\to$1;
- scegliere lettere da un periodo o da una frase;
- generazione di una sequenza casuale da parte di un computer. Questo sistema genera combinazioni difficili da ricordare, ma anche tali programmi possono avere dei bug. Per esempio, si sono avuti casi in cui la stessa combinazione apriva più lucchetti.

Un'altra tecnica procedurale intesa a migliorare la sicurezza della password è l'invecchiamento della password. Un periodo di tempo è definito come limite di ciascuna password (per esempio 30 o 90 giorni). Ciò richiede che le password siano cambiate periodicamente.

2.1.4 Rallentare il mapping della password

La funzione di verifica della password (per esempio, la funzione one-way) può essere resa intensa dal punto di vista computazionale; per esempio, iterando una semplice funzione per t volte, con $t>1$, con l'output dell'iterazione i usato come input all'iterazione $i+1$. Il numero totale di iterazioni deve essere limitato in modo da non imporre ai legittimi utenti un ritardo notevole o inammissibile.

2.1.5 Salting password

Per rendere gli attacchi di tipo dizionario meno efficaci, ogni password, nella lista iniziale, può essere aumentata con una stringa random di s bit chiamata *salt* prima di applicare la funzione one-way. Sia le password modificate che i bit di salt sono registrati nel file di password. Quando l'utente introdurrà una password, il sistema cercherà il salt, aggiungerà alla password tale valore e applicherà una funzione one-way alla stringa ottenuta. La difficoltà di una ricerca esaustiva su qualsiasi password di un particolare utente non è modificata dal salting (siccome il salt è dato in chiaro nel file delle password); comunque, il salting incrementa la complessità di un attacco di tipo dizionario effettuato simultaneamente su un insieme di password. Il dizionario richiede di contenere 2^s variazioni di ciascun tentativo, implicando una grande richiesta di memoria per registrare un dizionario cifrato, e in concomitanza molto tempo per la sua preparazione. Da notare che con il salting due utenti che scelgono la stessa password hanno differenti entry nel file di password. In alcuni sistemi può essere appropriato usare lo stesso userid dell'entità come salt.

2.1.6 Passphrase

Per consentire una maggiore entropia tenendo conto della capacità di memoria degli utenti, le password possono essere estese in *passphrase*; in questo caso, gli utenti digitano una frase o un detto, piuttosto che una semplice " parola ". La passphrases viene troncata se supera la lunghezza fissata ed ha lo stesso ruolo della password; è importante che la passphrase non sia facilmente troncata dal sistema, come viene fatto per le password in alcuni sistemi. L'idea è che l'utente possa ricordare le frasi più facilmente rispetto ad una sequenza di caratteri casuali. Se le password derivano dal testo inglese (in cui ogni carattere contiene solo circa 1.5 bit di entropia) una passphrase fornisce una sicurezza maggiore attraverso l'incremento dell'entropia rispetto ad una breve password. Un inconveniente è il dover digitare una quantità maggiore di testo.

2.2 Schemi a password fissa: attacchi

2.2.1 Reply alle password fisse

Una debolezza dello schema che usa password fisse riproponibili, è la possibilità che un avversario capisca la password di un utente osservando il modo in cui viene digitata. Un secondo problema di sicurezza è che le password inserite dall'utente (o funzioni one-way) siano trasmesse in chiaro sulla linea di comunicazione tra l'utente ed il sistema, e siano disponibili sempre in chiaro temporaneamente durante la verifica del sistema. Un avversario curioso potrebbe registrare questi dati ed effettuare successive impersonificazioni. Schemi di password fissa sono usati perciò quando la password è trasmessa su linee di comunicazione affidabili, sicure dal monitoraggio, ma non sono adatti nel caso in cui le password siano trasmesse su reti di comunicazioni aperte. Per esempio nella Figura 1, il richiedente A potrebbe essere un utente che si connette da casa con un modem sulla linea telefonica, ad un ufficio B, lontano due (o migliaia) di chilometri; la password in chiaro potrebbe viaggiare su una linea non sicura (includendo la possibilità di un collegamento via etere). Se si effettua una verifica remota dell'identità per accedere ad una risorsa locale, per esempio uno sportello bancario automatico con verifica dell' identità on-line, la risposta del sistema (accettazione/rifiuto) dovrebbe essere protetta insieme alla password.

2.2.2 Ricerca esaustiva di password

Un attacco molto semplice richiede che un avversario provi le password una alla volta, con un sitema verificatore nella speranza di trovare la password corretta. Questo tipo di attacco può essere reso più difficile ampliando lo spazio in cui esse vengono scelte, limitando il numero di tentativi non validi (on-line) consentiti in determinati periodi di tempo e rallentando il mapping della password. Gli *attacchi off-line* che non richiedono alcuna diretta interazione con il sistema verificatore sono di maggiore interesse. Dato un file di password contenente funzioni one-way delle password utente, un avversario può tentare di sconfiggere il sistema testando le password una alla volta e confrontando la funzione one-way di ciascuna con le password presenti nel file di password. Questo è teoricamente possibile siccome sia il mapping one-way che il testo (ipotetico) sono noti (ciò potrebbe essere evitato gestendo i dettagli di un mapping one-way o tenendo segreto il file delle password, ma non è considerato prudente basare la sicurezza del sistema sull'assunzione che tali dettagli rimangano segreti per sempre). La flessibilità dell'attacco dipende dal numero di password che devono essere controllate prima di trovare quella giusta e dal tempo richiesto per testarne una. Infine conta il mapping di password usato, la sua implementazione, il tempo di esecuzione dell'istruzione del processo host e il numero di processori disponibili (è da notare che la ricerca esaustiva è parallelizzabile). Il tempo richiesto attualmente per confrontare ogni tentativo con tutte le password in un file di password è di solito trascurabile.

Esempio (entropia della password) Supponiamo che le password consistano di caratteri ASCII di 7 bit. Ognuno ha un valore numerico nell'intervallo 0-127 (quando sono usati caratteri a 8 bit il valori 128-255 compongono *l'insieme di caratteri esteso*, generalmente non accessibile dalle tastiere standard). I codici ASCII 0-31 sono riservati per il controllo dei caratteri; 32 è il carattere di spazio; 33-126 sono caratteri stampabili accessibili da tastiera; 127 è un carattere speciale. La Tabella 1 fornisce i numeri di possibili password composte da n caratteri distinti scelti tra combinazioni tipiche di caratteri, indicando un limite superiore sulla sicurezza di tali password.

2.2.3 Attacchi dizionario

Per migliorare la probabilità di successo di una ricerca esaustiva, piuttosto che cercare nello spazio di tutte le possibili password, un avversario può cercare in uno spazio ridotto. Idealmente le stringhe arbitrarie di N caratteri potrebbero essere tanto probabili quanto le password selezionate dall'utente, di solito però la maggior parte degli utenti seleziona le password da un piccolo sottoinsieme dell'intero spazio delle password (per esempio, password brevi, parole del dizionario, nomi propri, stringhe in minuscolo). Tali password deboli con bassa entropia, sono facilmente prevedibili; infatti, studi indicano che un gran numero di password selezionate

→ C ↓ N	**26 (minuscole)**	**36 (minuscole alfanumeriche)**	**62 (miste alfanumeriche)**	**95 (caratteri da tastiera)**
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6
9	42.3	46.5	53.6	59.1
10	47.0	51.7	59.5	65.7

Tabella 1: *Il numero di password composte da N caratteri, date C scelte per carattere, è C^N. La tabella fornisce il logaritmo in base 2 del numero di possibili password.*

→ C ↓ N	**26 (minuscole)**	**36 (minuscole alfanumeriche)**	**62 (miste alfanumeriche)**	**95 (caratteri da tastiera)**
5	0.67 ore	3.4 ore	51 ore	430 ore
6	17 ore	120 ore	130 giorni	4.7 anni
7	19 giorni	180 giorni	22 anni	440 anni
8	1.3 anni	18 anni	1400 anni	42000 anni
9	34 anni	640 anni	86000 anni	$4.0*10^6$ anni
10	890 anni	23000 anni	$5.3*10^6$ anni	$3.8*10^8$ anni

Tabella 2: *Tempo richiesto per effettuare una ricerca nell'intero spazio della password. La Tabella dà il tempo T (in ore, giorni, o anni) richiesto per una ricerca nell'intero spazio usando un singolo processore. $T = C^N * t * y$, dove t è il numero di volte che il mapping della password è iterato, e y è il tempo per ogni iterazione, C rappresenta le possibili scelte per i caratteri della password, N è il numero di caratteri che compongono la password. Per $t = 25$, $y = 1/125000$ sec. (ciò approssima il comando crypt di UNIX su un PC che esegue il DES a 1.0 MBytes/s).*

dagli utenti sono trovate in tipici dizionari di sole 150000 parole, mentre un ampio dizionario di 250000 parole rappresenta solo una piccola frazione di tutte le possibili password di N caratteri (Tabella 1). Nel 1979, Morris e Thompson [4] esaminarono un campione di 3289 password e ne trovarono 2831 facili da attaccare per una delle seguenti ragioni:

- 15 avevano un singolo carattere ASCII;
- 72 avevano una stringa di due caratteri ASCII;
- 464 avevano una stringa di tre caratteri ASCII;
- 477 avevano una stringa di quattro caratteri ASCII;
- 706 avevano una stringa di cinque lettere, dello stesso tipo (minuscole o maiuscole);
- 605 avevano una stringa di sei lettere, tutte minuscole.

Le password trovate in qualsiasi lista di parole on-line potrebbero essere scoperte da un avversario che tenta tutte le parole in questa lista usando un cosidetto attacco dizionario. Oltre i tradizionali dizionari, ne esistono altri di parole in lingue straniere, o su argomenti specializzati come musica, film etc. Gli attacchi di tipo dizionario non hanno successo in genere nel trovare la password di un particolare utente, ma trovano molte password nella maggior parte dei sistemi.

2.3 Password di UNIX

Il sistema operativo UNIX utilizza un file di password che viene consultato nel momento in cui l'utente vuole accedere alle risorse. Per il riconoscimento dell'utente, UNIX usa una funzione one-way. Ogni password dell'utente viene usata come chiave per cifrare un testo in chiaro (64 bit posti a zero). Si sceglie una funzione f in modo che, data una password x, posso calcolare f facilmente mentre l'inversa è difficile da calcolare.

$$x \xrightarrow{\text{"facile"}} f(x) \qquad\qquad f(y)^{-1} \xleftarrow{\text{"difficile"}} y$$

I valori contenuti nel file di password sono i seguenti:

- *userid* è utilizzato per identificare l'utente mentre la password ne conferma l'identità.
- s è il valore dei bit di salt generati dal sistema (sono 12 bit).
- $f(x, s)$ dove x è la password digitata dall'utente e s è il salt. Tale funzione permette il riconoscimento della password.

Il motivo per cui vengono usati i bit di salt è di rendere difficile l'attacco di tipo dizionario (in tale attacco si mantengono un insieme di password comuni che sono confrontabili con la $f(x, s)$). Questa difficoltà è dovuta al fatto che avendo 12 bit di salt per ogni possibile valore del dizionario, si dovrebbero specificare $2^{12} = 4096$ possibili password. I 12 bit sono random e vegnono generati tramite il clock di sistema.

UNIX utilizza come funzione f una versione del DES modificata, applicando la tecnica di rallentamento del mapping delle password (cfr. 2.1.4), ed imponendo il valore t=25. In dettaglio, la password di un utente è troncata ai suoi primi 8 caratteri ASCII, ognuno dei quali è codificato con 7 bit, otteniamo quindi i 56 bit della chiave che useremo nell'algoritmo DES modificato. In particolare cifriamo 64 bit posti tutti a zero, l'output entrerà nel ciclo delle t iterazioni.

Ai 64 bit forniti come output del DES vengono aggiunti 12 bit di salt per un totale di 76 bit, che divisi in gruppi di 7 ci daranno 11 caratteri ASCII che vengono scritti nel file di password. Gli 11 caratteri ottenuti rappresenteranno la password. I 12 bit di salt vengono usati per modificare l'output della funzione espansione standard del DES. Ai 48 bit ottenuti si applica una permutazione basata sui valori dei bit di salt. (Scambia i bit di posizione i e $24 + i$ se l'i-esimo bit di salt è pari a 1, li lascia invariati se vale 0, dove $1 \leq i \leq 12$. Gli altri bit restano invariati in ogni caso). Questo procedimento è detto *password salting* e serve per prevenire l'uso di *off-the-shelf DES chip*, in quanto la funzione di espansione E del DES dipende dai bit di salt. Non è quindi possibile usare i chip DES standard per implementare l'algoritmo di UNIX. Un avversario non potrebbe usare i chip in commercio per rompere il cifrario. I valori memorizzati nel file di password /etc/passwd (protetto in scrittura) in corrispondenza di ciascun userid scaturiscono dall'applicazione di questo algoritmo che viene chiamato *crypt* di UNIX.

2.3.1 Sicurezza in UNIX

Su un sistema Unix, *l'autenticazione dell'utente* è basata su uno schema di password.

Assumiamo che l'utente abbia un *account* sul sistema, che richiede una login, dello spazio allocato su disco, e qualunque cosa necessaria, affinchè l'utente possa lavorare sul sistema. Il protocollo è molto semplice, l'utente introduce la sua *login* e la sua *password*.

Il sistema cifra queste password e le confronta con una versione memorizzata, contenuta nel file **/etc/passwd**. Ecco riportato di seguito un esempio del file:

```
root:Igl6derBr45Tc:0:0:The Superuser:/:/bin/sh
bin:*:1:1:Pseudo-system:/:/bin/sh
bob:J4wg6qi6IkZEj:100:10:& Clark:/u/bob:/bin/sh
alice:ntkYb1io0kk3j:101:20:& Tyler:/u/alice:/bin/sh
ftp:*:200:20:Anonymous Ftp User:/u/ftp:/bin/sh
...
```

Ad ogni utente è associata una riga all'interno del file. Ogni riga è suddivisa in campi, i quali sono separati dai due punti: la login (bob), la password cifrata (J4wg6qi6IkZEj), l'identificatore dell'utente (100), un identificatore di gruppo (10), alcune informazioni sull'utente (& Clark), la sua home directory (/u/bob), e la sua shell (/bin/sh).

3 PIN e passkey

3.1 PIN (numero di identificazione personale)

I PIN sono password fisse. Per aumentare la sicurezza, ad ogni utente viene associato un *token* fisico, o come una carta plastificata dotata di una striscia magnetica, o una chipcard. Chiunque voglia accedere all'uso di particolari risorse, deve farsi riconoscere come utente autorizzato del token, digitando correttamente il proprio PIN. In questo modo i PIN rafforzano la sicurezza nel caso in cui il token venga smarrito o rubato. Per ragioni storiche e di convenienza, il PIN è un numero breve, per esempio da 4 a 8 cifre. Per evitare una ricerca esaustiva in uno spazio di chiavi piccolo, occorre aggiungere alcuni vincoli procedurali. Per esempio, alcuni sportelli automatici bancari ritirano la carta magnetica se l'utente non digita correttamente il proprio PIN per tre volte consecutive; in altri casi la carta viene disattivata se si esegue un numero prestabilito di immissioni scorrette, per poi essere riattivata richiedendo un PIN più lungo. In un sistema on-line che fa uso di PIN o di password riutilizzabili, una richiesta di identificazione seguita dall'introduzione del PIN dell'utente può essere verificata confrontando i dati in ingresso con il PIN memorizzato nel database di sistema. In un sistema off-line senza accesso ad un database centrale, l'informazione che rende possibile la verifica del PIN deve essere memorizzata sul token stesso. In questo caso l'utente può eseguire o meno una digitazione, in caso affermativo, il PIN può essere cifrato sotto una master key e memorizzato sul token, con la master key conosciuta a tutti i terminali off-line che necessitano di essere capaci di identificare il token. Un progetto preferibile è di memorizzare una funzione one-way del PIN, l'identità dell'utente e la master key sul token.

3.2 Identificazione a due livelli e password-key

Gli utenti umani hanno difficoltà a ricordare le chiavi segrete, soprattutto se hanno un'entropia sufficiente a garantire un buon livello di sicurezza. Descriveremo due tecniche che evitano tale inconveniente.

Quando i token sono usati per la verifica off-line del PIN, una comune tecnica prevede di usare il PIN per consentire al token di riconoscere l'utente, mentre il token stesso contiene ulteriori informazioni indipendenti che gli consentono di farsi autentificare dal sistema. L'utente è indirettamente identificato dal sistema tramite un processo a due livelli. Ciò richiede che l'utente sia in possesso del token, ma che ricordi solo un breve PIN, mentre una chiave lunga (con un'adeguata entropia) provvede alla sicurezza crittografica per l'identificazione su un canale non sicuro. Una seconda tecnica mappa una password utente, tramite una funzione hash, in una chiave crittografica. Tali *password-deliver key* sono chiamate *passkey*. La passkey è usata per assicurare una comunicazione tra l'utente e il sistema, che conosce la password utente. L'entropia della password utente deve essere talmente alta che una ricerca esaustiva nello spazio della password non deve essere più efficiente di una ricerca esaustiva nello spazio della passkey (Tabella 1). Passkey potrebbero essere viste come chiavi a lungo termine con l'uso ristretto all'autenticazione e alla gestione della chiave (per esempio piuttosto che per blocchi di dati di utente cifrati). Uno svantaggio dell'uso di password-delivered key è che memorizzare ogni password dell'utente con il sistema richiede alcuni meccanismi per proteggere la riservatezza delle password registrate.

4 One-time password

Un progresso naturale dagli schemi a password fisse ai protocolli di identificazione challenge-response può essere osservato considerando gli schemi one-time password. La maggiore sicurezza rispetto agli schemi di password fisse è nello scovare e successivamente ripetere le password. Una parziale soluzione è lo schema *password one-time*: ogni password è usata solo una volta. Tali schemi sono protetti da avversari passivi che scoprono le password e poi tentano di usarle.

Le variazioni a tale schema includono:

1. *Liste condivise di password one-time.* L'utente ed il sistema usano una sequenza o un insieme di t password segrete (ognuna valida per una singola autenticazione), distribuite come una lista pre-condivisa. Uno svantaggio è il dover mantenere una lista condivisa. Se la lista non fosse usata sequenzialmente, il sistema potrebbe controllare le password inserite rispetto a quelle inutilizzate.

   Una variazione coinvolge l'uso di una tabella challenge-response, per cui l'utente ed il sistema condividono una tabella di confronto *challenge-response*, idealmente con ogni coppia valida al più una volta. Questa tecnica non crittografica differisce dalla challenge-response crittografica.

2. *One-time password aggiornate sequenzialmente.* Inizialmente solo un'unica password segreta è condivisa. Durante l'autentificazione usando la password i, l'utente crea e trasmette al sistema una nuova password (password $i+1$) cifrata con una chiave derivata dalla password i. Questo metodo diventa difficile se si verifica un crollo della comunicazione.

3. *Sequenze di one-time password basate su una funzione one-way.* Lo schema della one-time password di Lamport è descritto successivamente. Questo metodo è più efficiente rispetto all'aggiornamento sequenziale delle one-time password, e può essere visto come un protocollo challenge-response dove la sfida (challenge) è implicitamente definita dalla posizione corrente nella sequenza delle password.

4.1 Schema di Lamport (One-time password basate su funzioni one-way)

Nello schema one-time password di Lamport l'utente inizia con un segreto w. Una funzione one-way H è usata per definire la sequenza di password: w,$H(w)$,$H(H(w))$,....,$H^t(w)$.

La password per l'i-esima sessione di identificazione, $1 \leq i \leq t$, è definita come $w_i = H^{(t-i)}(w)$.

Sia A un utente che si vuole identificare ad un sistema B. L'identificazione avviene usando una one-time password estrapolata da una sequenza.

Descriviamo di seguito il protocollo di Lamport basato su one-time password.

1. *One-time setup*
   - L'utente inizia con un segreto w. Sia H una funzione one-way.
   - Un vincolo t è fissato (per esempio $t = 100$ oppure 1000), definendo il numero di identificazioni che devono essere eseguite (Il sistema è poi riavviato con un nuovo w, per evitare attacchi di replay).
   - A trasferisce $w_0 = H^t(w)$ (il segreto condiviso), in modo da garantire la sua autenticità al sistema B. B inizializza il suo contatore per A ad $i_A = 1$.
2. *Messaggi inviati.* Durante l'i-esima identificazione $1 \leq i \leq t$, A invia a B un messaggio composto da: identificatore utente (A), numero dell'identificazione (i) e la password per l'i-esima identificazione ($w_i = H^{(t-i)}(w)$).
3. *Computazioni effettuate.* Per identificarsi per la sessione i, A si comporta come segue:
   - A calcola $w_i = H^{(t-i)}(w)$, facilmente derivato sia da w stesso o da un appropriato valore intermedio salvato durante la computazione di $H^t(w)$, per poi effettuare il punto 2.
   - B controlla che i=i_A, e che la password ricevuta w_i soddisfi: $H(w_i) = w_{i-1}$. Se entrambi i controlli hanno successo, B accetta la password, aggiorna $i_A \leftarrow i_A + 1$, e salva w_i per la successiva sessione di verifica.

Riferimenti

[1] Bart Preneel - Renè Govaerts - Joos Vandewalle, *"Computer Security and Industrial Cryptography"*, Springer - Verlag, 1991.

[2] Daniel V. Klein *"Foiling the Cracker: A Survey of, and improvements to, password security."* Draft, 1990.

[3] Menezes - Vanstone *"Handbook of Applies Cryptography."* CRC Press, 1998.

[4] Morris - Thompson, *CACM*, Nov. 1979, pp. 594-597.