Password Based Encryption

Riccardo Longo

Password

- La **password** è uno dei metodi di autenticazione più semplici e antichi
- Molto diffuso per **comodità** di implementazione e uso
- Molti problemi di sicurezza derivati da **uso improprio**
  - Responsabilità della sicurezza ricade in gran parte sull'**utente**
  - Gli utenti in generale sono **poco formati** sulle pratiche di sicurezza
  - Ci sono tanti **fraintendimenti** riguardo l'uso corretto

Autenticazione

- Molte applicazioni richiedono il **riconoscimento** dell'utente:
  - Permettere l'**accesso** solo a utenti autorizzati
  - Assegnare le risorse di **competenza** (es propri documenti)
  - Far **riconoscere** gli utenti tra loro
- L'autenticazione si basa su uno o più dei seguenti **fattori**:
  - Qualcosa che l'utente **sa** (password)
  - Qualcosa che l'utente **ha** (token, smartcard)
  - Qualcosa che l'utente **è** (biometria)

Pro e contro

- Le password hanno il vantaggio di non richiedere **dispositivi addizionali**:
  - L'utente non deve portare con se *chiavi* fisiche
  - Il sistema non deve avere appositi lettori
- La **facilità** d'uso per l'autenticazione è elevata
- Ma hanno anche seri svantaggi:
  - L'utente può **dimenticare** la password
  - L'utente medio non è molto abile a creare e memorizzare password con sufficiente **entropia**

Casi d'uso

- **Log-in**
- Riconoscimento / conferma dell'**identità**
- **Protezione** di risorse
  - La password è usata per **ricavare una chiave** di cifratura
  - Una password **non è adatta** ad essere usata direttamente come chiave
  - È necessaria molta **entropia**

Entropia

- Misura la **casualità** e la difficoltà di predizione
- La forza di una password è la sua imprevedibilità
- Se ogni possibilità fosse equiprobabile l'entropia sarebbe:

$$E = \log_2(N) = \log_2(C^L) \tag{1}$$

  dove N è il numero di password possibili, C il numero di caratteri ammessi ed L la lunghezza della password
- Dà una misura del numero di tentativi necessari per indovinare
- Misurata in bit

Sicurezza di password random

entropia	dec	hex	alf. CI	a.n. CI	alf. CS	a.n. CS	ASCII	ASCII E	Diceware
8	3	2	2	2	2	2	2	2	1
32	10	8	7	7	6	6	5	5	3
40	13	10	9	8	8	7	7	6	4
64	20	16	14	13	12	11	10	9	5
80	25	20	18	16	15	14	13	11	7
96	29	24	21	19	17	17	15	13	8
128	39	32	28	25	23	22	20	17	10
160	49	40	35	31	29	27	25	21	13
192	58	48	41	38	34	33	30	25	15
224	68	56	48	44	40	38	35	29	18
256	78	64	55	50	45	43	39	33	20

dec: cifra decimale, **hex**: cifra esadecimale **alf.**: carattere alfabetico, **a.n.**: carattere alfanumerico
CI: case insensitive, **CS**: case sensitive
ASCII carattere ASCII stampabile, **ASCII E** carattere stampabile ASCII esteso
Diceware: parole selezionate tramite lancio di 5 dadi

Password generate da umani

- In realtà utenti umani tendono ad usare password **non random**
- I caratteri di un testo scritto hanno poco più di **un bit** di entropia
- Molte scelte sono più **probabili** di altre
- Ci sono **pattern e schemi ricorrenti**
- Gli attacchi tengono in considerazione queste **debolezze**

Password deboli

- Password di **default** (password, admin,...)
- Parole del **dizionario**
- Sequenze della **tastiera** (qwerty, asdfgh)
- Derivati dell'username o altre **informazioni personali**
- Sequenze numeriche **famose** (3142592)
- **Citazioni** di testi o canzoni
- Semplici **variazioni**:
  - Aggiunte finali di numeri
  - Sostituzioni comuni (a-@, i-1, o-0,...)
  - Raddoppio di parole

Equivoci sulla sicurezza delle password

- L'obbligo ad usare maiuscole, minuscole, numeri e caratteri speciali non aumenta in pratica la sicurezza: sono adottate **semplici variazioni**
- Il **cambio frequente** tende a diminuire la sicurezza: sono usate password più deboli e derivate l'una dall'altra
- L'uso di password diverse per ogni cosa può portare all'indebolimento di tutte le password, i **servizi critici** vanno protetti maggiormente
- Il divieto di **scrivere e/o salvare le password** porta al riuso e/o all'indebolimento: è meglio avere una copia e proteggerla a dovere

Attacchi alle password

- **Intercettazione**, **shoulder-surfing**
- **Social Engineering**
- Attacchi **on-line**
- Attacchi **off-line**

- Forza bruta
- Dizionario
- Combinazioni
- Permutazioni
- Toggle-case
- Ibridi
- Rule-based
- Mask
- PRINCE
- ...

Protezione On-line

- **Limitare** il numero di tentativi
- **Timeout** incrementali
- Blocco dopo tot fallimenti **consecutivi**
- Forzare il cambio dopo tot fallimenti **cumulativi**
- Mostrare l'ultimo accesso

Attacchi Off-line

- Viene violato il server di autenticazione
- C'è accesso diretto alla macchina
- Vengono aggirati blocchi e limitazioni del sistema
- Tentativi illimitati, senza interazione

Le password non vanno mai salvate in chiaro!

One-Way Function

- Funzioni facili da calcolare ma estremamente **difficili da invertire**

$$x \to y = f(x) \qquad y \nrightarrow x : y = f(x)$$

- **Hash crittografica**:
  - Deterministica
  - Efficiente
  - Piccole variazioni nell'input cambiano molto il risultato
  - Dal risultato non si deve dedurre nulla sull'input se non andando per tentativi

Sicurezza delle Hash crittografiche

- **Pre-image**

$$y \nrightarrow x : y = f(x)$$

- **Second pre-image**

$$x_1 \nrightarrow x_2 : f(x_1) = f(x_2),\ x_1 \neq x_2$$

- **Collision**

$$\nrightarrow x_1, x_2 : f(x_1) = f(x_2),\ x_1 \neq x_2$$

Principali Hash

- **MD5**: molto veloce, **non sicura**
- **SHA-1**: molto diffusa, **non più sicura**
- **RIPEMD**: design potenzialmente vulnerabile
- **SHA-2**: standard, ritenuta ancora sicura ma con **caveat**
- **SHA-3**: ultimo standard, **più sicura**, velocissima in hardware
- **BLAKE2**: finalista per lo standard SHA-3, estremamente **veloce** su CPU moderne

Usi delle Hash

- Verifica dell'**integrità** di un messaggio
- **Proof-of-Work**
- **Identificatore** di file o dati
- Base per la **costruzione** di altre primitive crittografiche (PRNG)
- Verifica delle **password**

Hash di password

- Evita l'**esposizione diretta** delle password
- Permette l'autenticazione veloce **confrontando l'hash**
- L'hash da solo non permette autenticazioni malevole
- Suscettibile ad **attacchi off-line**: facile verificare i tentativi
- Espone password **comuni e ri-usate**
- Possibilità di **pre-computazione** e look-up (dizionario, rainbow table)

Salt

- Input addizionale generato **random**, concatenato alla password
- Salvato in chiaro (dev'essere **non-prevedibile**, non **segreto**)
- Costringe il **ricalcolo** dell'hash per ogni password
- Protegge da **collisioni** tra utenti
- Impedisce la **pre-computazione**
  - Non vanno **mai ri-usati**
  - La **lunghezza** dev'essere sufficiente

Attacchi di Forza Bruta

- Le funzioni di hash sono efficienti, quindi si possono provare molte password al secondo
- L'attacco può essere parallelizzato
- Si può sfruttare hardware specializzato
  - GPU
  - ASIC
  - FPGA
  - Cluster
  - Botnet
- Meglio usare funzioni appositamente dispendiose

Funzioni volutamente dispendiose

- Un utente legittimo deve fare **un solo** tentativo
- Un attaccante vuole provare **molte ipotesi**
- Si **rallenta** il più possibile il singolo tentativo per **frustrare** l'attaccante
- Per limitare **parallelizzazione** e **compromessi tra tempo e memoria** dev'essere dispendioso su tutti i fronti
- Design basati su **iterazione** e **accesso pseudorandom** alla memoria
- Dispendiosità regolata da **parametri**

Key Derivation

- Queste funzioni sono usate per **generare chiavi** a partire da password
- L'output **pseudorandom** maschera l'origine
- Permettono la creazione di chiavi della lunghezza adatta **evitando pattern deboli**
- La dispendiosità della ricerca riduce l'efficacia della forza bruta, **compensando il più piccolo spazio di origine**

Key Stretching, Key Strengthening

- **Key Stretching**:
  - Deriva chiavi più sicure a partire da chiavi corte, o con poca entropia (password, passphrase)
  - Usano un salt e un parametro di complesssità per aumentare le risorse necessarie per testare la chiave
- **Key Strengthening**:
  - La rafforzatura avviene in maniera simile
  - Il salt usato è cancellato, costringendo a un brute-forcing parziale sia utenti legittimi che attaccanti

Principali KDF

- **PBKDF2 (Password Based Key Derivation Function)**
  - Standard
  - Itera una funzione pseudorandom basata su hash (**HMAC**)
  - Usa **poca RAM**
- **bcrypt**
  - Basato sul cifrario simmetrico **Blowfish**
  - Il parametro influenza sia il costo in **tempo-CPU** che in **RAM**
  - Adatto per password hashing ma **non** per key derivation
- **scrypt**
  - Usa **PBKDF2** e lo stream cipher **Salsa20/8**
  - Permette di controllare **parallelizzazione**, costo **CPU** e **RAM**
- **Argon2**
  - Vincitore della competizione pubblica per **Password Hashing**
  - Ha varianti ottimizzate per **resistenza** a **GPU** e **side-channel**
  - Basato su **Blake2**, parametri simili a **scrypt**

Raccomandazioni NIST sulle password (2017)

- Almeno 8 caratteri
- Supporto fino almeno 64 caratteri
- Permettere tutti i caratteri ASCII
- Mai troncare la password nella verifica
- Non bisogna imporre la presenza di diversi tipi di carattere
- Non bisogna forzare cambi periodici della password, ma solo in caso di compromissione
- Non bisogna permettere il salvataggio di "suggerimenti" pubblicamente accessibili o suggerire di usare informazioni specifiche

Raccomandazioni NIST (2)

- Alla creazione o cambio della password questa va confrontata con liste per evitare debolezze:
  - Password compromesse di dominio pubblico
  - Dizionari
  - Ripetizioni o sequenze
  - Parole specifiche al contesto e derivati (nome servizio, username)
  - Altre liste di password deboli
- Quando una password debole è rifiutata bisogna fornire la motivazione e suggerire come migliorare
- Bisogna limitare il numero di autenticazioni fallibili
- Le password vanno salvate in una forma resistente ad attacchi offline, usando KDF apposite che prendano in input password, salt, e fattore di costo, per impedire la ricerca a tentativi