User Authentication

I sistemi di autenticazione rappresentano una delle prime misure di protezione contro gli attacchi che abbiamo visto nei capitoli precedenti, in quanto consentono di verificare l'identità di un utente, che prova ad accedere al sistema → **hanno, quindi, l'obiettivo di verificare che un utente sia effettivamente chi dichiara di essere nel momento in cui accede ad un servizio**. I sistemi di autenticazione, inoltre, permettono di garantire anche altre proprietà di sicurezza ed in particolare per garantire:

- la **proprietà di autorizzazione** e di conseguenza anche la **riservatezza** → questo perchè, nel momento in cui abbiamo verificato l'identità di un utente, e quindi gli abbiamo associato un username, possiamo associarli anche i privilegi che ha all'interno del sistema dell'organizzazione, ovvero possiamo andare a definire quali sono le attività, che l'utente può svolgere all'interno del sistema dell'organizzazione;
- **l'accountability** (o responsabilizzazione) → ci permette di sapere cosa fa ciascun utente all'interno del sistema e di conseguenza se una certa attività sospetta o un qualche comportamento anomalo è associato ad un determinato utente autenticato, possiamo verificare se effettivamente l'attività sospetta o il comportamento anomalo corrisponde ad un attacco informatico.

"**Come facciamo a verificare l'identità di un utente?**" La verifica dell'identità, di solito, si basa sulla combinazione di tre aspetti:

1. Qualcosa che la persona **conosce** (come una password);
2. Qualcosa che la persona **possiede** (come una smart card o un token);
3. Qualcosa che la persona **è**, quindi una **caretteristica fisica** (come l'impronta digitale o la retina dell'occhio).

Focalizziamoci sul primo aspetto, ovverosia i sistemi di autenticazione basati su password. Questi sistemi sono **largamente utilizzati** per diversi motivi, tra cui:

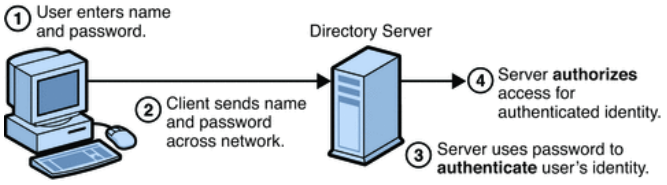
- **facilmente implementabili**:
- **costi** di implementazione e manutenzione **molto bassi**.

Un sistema di autenticazione basato su password si compone di due fasi:

1. una **fase** iniziale di **registrazione** → in cui l'utente, tipicamente, sceglie un username ed una password (ovverosia il segreto). L'username e la password

vengono salvati sul Server di autenticazione gestito dall'authentication service;

2. una **fase** di **autenticazione** → nel momento in cui l'utente deve essere autenticato, semplicemente ripresenta username e password e il servizio di autenticazione controlla che essi siano presenti nel file delle password salvate creato dell'authentication service. Se username e password corrispondono, allora l'utente è loggato.



Questo sistema, nonostante sia molto conveniente in termini di implementazione e costi, non rappresenta il sistema di autenticazione più sicuro. Infatti, recenti studi hanno dimostrato che una delle principali cause di attacchi di Data breach riguarda l'utilizzo di credenziali rubate (o comunque crackate) dagli attaccanti → quindi, una delle cause principali di attacchi deriva dal fatto, che gli attaccanti sono in grado di rubare e/o crackare le credenziali degli utenti di un'organizzazione e naturalmente questo ha effetti negativi:

- per gli **utenti**, in quanto molto spesso quando gli attaccanti riescono ad ottenere le credenziali dell'utente, riescono ad accedere anche ad altre informazioni sensibili dell'utente (come ad esempio i dati della carta di credito e codice fiscale) e di conseguenza l'attaccante può compiere altre tipologie di attacco;
- per l'**organizzazione**, in quanto subisce un danno alla reputazione ed economici.

Sorge allora spontanea una domanda: "**Perchè non è un sistema sicuro?**" Per il fatto, che gli utenti si trovano a gestire un elevato numero di account, dato che mediamente un utente si trova a gestire circa 20 account e di conseguenza, l'utente quando deve settare la password sceglie:

- **password facilmente memorizzabili** e di conseguenza facilmente indovinabili dagli attaccanti (per esempio si combina il nome di un nostro famigliare con la sua data di nascita) → in generale, gli utenti utilizzano una sequenza di numeri oppure parole molto comuni (recenti studi hanno mostrato che le password maggiormente utilizzate quest'anno sono: 123456, 12345, password);
- la **stessa password** viene **riutilizzata** per molteplici account ed in particolar modo, la stessa password viene utilizzata per account di lavoro, account di

shopping online e account di reti sociali.

> 💡 Per questi motivi, si è stimato che 2 utenti su 5 subisce un attacco di Data breach, in cui l'attaccante guadagna immediatamente il controllo dell'account degli utenti.

Le tipologie di attacco con cui si l'attaccante ottiene la password dell'utente si dividono in quattro macro-categorie:

1. **Offline Attacks** → in questo caso, l'attaccante deve ottenere l'accesso al password file, il quale è memorizzato sul Server di autenticazione o sulla macchina delle vittime ed in particolare, l'attaccante utilizza diverse strategie per provare diverse password, fino a quando trova la stessa password contenuta nel password file.
2. **Active Online Attacks** → in questo caso, l'attaccante non ha accesso al password file, ma ha solamente accesso al sito/sistema di cui vuole ottenere l'accesso. L'attaccante, quindi, con questo attacco può avere come informazioni preliminari, una serie di username che vengono utilizzati da utenti legittimi per accedere al sito/sistema ed eventualmente conoscenza su come vengono strutturate le password e quindi sulla politica di creazione delle password adottata dal sito/sistema → anche con questa tipologia di attacco, gli attaccanti provano ad autenticarsi utilizzando diverse password, fino a quando non ottengono l'accesso;
3. **Passive Online Attacks** → gli attaccanti non interagiscono direttamente con il sistema, ma intercettano la password utilizzando diverse tecniche che vedremo successivamente;
4. **Non Technical Attacks** → ovverosia gli attacchi di ingegneria sociale, i quali manipolano psicologicamente gli utenti, per portare quest'ultimi a rivelare le credenziali.

"**Sui sistemi più comunemente adottati come obiettivi di questi attacchi, dove sono salvate le password?**" Sui sistemi Windows vi sono diversi posti/locations in cui le password vengono salvate. Solitamente gli account locali ad una macchina Windows sono salvati nel cosiddetto **SAM Database**, il quale si trova in due posti:

1. solitamente si trova nella **directory \Windows\System32\config**, dove solo gli amministratori possono accedere;

2. una copia è anche salvata in una **chiave dei registri** di Windows chiamata **SAM**.

> 💡 Per ottenere una copia del SAM Database basta ottenere un accesso alla macchina della vittima come utente local amministrator e poi eleviamo i nostri privilegi come utente amministratore, possiamo aprire un Command Prompt sulla macchina della vittima e semplicemente con un comando “rage save” siamo in grado di copiare il SAM Database.

Un’altra location in cui sono salvate le password è **LSAS** (**Local Security Authority Subsystem Service**), ovvero un processo di Windows dove le password, degli utenti registrati, sono salvate nella memoria allocata per l’esecuzione di tale processo → gli attaccanti, quindi, per ottenere una copia delle password salvate nella memoria del processo LSAS utilizzano **Mimikats**, ovverosia un programma gratuito e open source per Windows, che può essere utilizzato per ottenere informazioni riguardanti le credenziali di accesso.

Sulla macchina Linux, invece, le password degli utenti che si possono collegare alla macchina, sono salvate in due file:

1. **etc/passwd** → contiene tutti gli username degli utenti;
2. **etc/shadow** → contiene l’hash delle password degli utenti.

> 💡 Sulle macchine Linux, quindi, gli attaccanti per indovinare le password devono fare il merge dei due file in un unico file e poi possono fare il cracking.

Le password, tipicamente, non vengono memorizzate in chiaro, ma vengono salvate tramite una **funzione di hash** → le funzioni di hash sono funzioni matematiche, che generano a partire da un valore in input, un valore che non ha alcun legame con il valore originario e di conseguenza non è possibile dal valore di hash risalire al valore originario. Vediamo, allora, un esempio di SAM Database estratto da una macchina Windows:

Possiamo vedere, che nel SAM Database abbiamo:

- **Username**;
- **SID** → che rappresenta il gruppo a cui appartiene l'utente;
- **2 hash in 2 formati** → il primo segue il formato LM (Lan Manager) ed il secondo segue il formato NTLM (New Technology Lan Manager). Il formato LM veniva tipicamente utilizzato nei sistemi Windows precedenti a Widows 2000 e viene ancora utilizzato nel SAM Database per motivi di compatibilità. Il formato, invece, oggi giorno più utilizzato è il formato NTLM e i due formati si differenziano per diversi aspetti:
  - si differenziano per la sicurezza → NTLM ci consente di calcolare l'hash di password lunghe 256 caratteri, mentre con LM era possibile calcolare l'hash di password lunghe al massimo 14 caratteri;
  - si differenziano anche per il fatto, che per calcolare l'hash NTLM utilizza tutti i caratteri della codifica UTF-8 (quindi fino a 65000 caratteri), mentre LM adotta un approccio diverso e le password calcolate con questa funzione di hash sono facilmente crackabili → in particolare, LM permette di fare l'hash password lunghe fino a 14 caratteri e abbiamo che:
    - tutte le lettere vengono convertite da minuscolo a maiuscolo;
    - se la password è più corta di 14 caratteri, vengono inseriti dei caratteri vuoti per riempire tutti i 14 caratteri:
    - la password di 14 caratteri viene suddivisa in due stringhe da 7 caratteri ciascuna;
    - ciascuna stringa di 7 caratteri viene quindi crittografata con un algoritmo a chiave simmetrica e combinate nuovamente

---

Una volta che gli attaccanti sanno quale tipologia di hash è stata utilizzata, possono iniziare a condurre varie tipologie di attacco:

- Una prima tipologia di attacco, che possono adottare, è l'**attacco di tipo Brute Force** → con questa tipologia di attacco, l'attaccante prova sostanzialmente tutte le possibili combinazioni di caratteri, numeri e simboli speciali che compongono la password. Successivamente ne calcola l'hash e poi va a confrontarlo con l'hash delle password contenute nel password file → l'attaccante ripete questo processo fino a quando non trova una password contenuta nel password file. Nel caso peggiore, l'attaccante per riuscire a recuperare una password è $|A|^n$. Assumiamo, allora ad esempio, una password lunga 8 caratteri: tra maiuscole, minuscole, numeri e simboli speciali possiamo avere 96 caratteri possibili e di conseguenza abbiamo $96^8 = 7.2$ quadrilioni di possibili combinazioni.

  Attraverso questa tipologia di attacco, abbiamo che sicuramente prima o poi l'attaccante troverà una o più password contenute nel password file, ma ha come svantaggio il fatto che richiede numerose risorse computazionali e risorse in termini di tempo;

- Un'altra tipologia di attacco sono i **Dictionary Attack** → approccio più "smart" utilizzato dagli attaccanti, che consiste nel prendere parole da una lista (detto **dizionario**) di parole più comuni ed utilizzate e creare l'hash di ciascuna di queste parole e confrontarlo con il valore degli hash presenti nel password file. Lo svantaggio di questo approccio, è che trova le password solamente se esse sono contenute nel dizionario e quindi, se non sono contenute nel dizionario l'attacco non ha successo. Se però la password è contenuta nel dizionario, allora gli attaccanti la troveranno in tempi molto più brevi rispetto ad un Brute Force Attack;

- abbiamo poi gli **Hybrid Attack** → combinano i due attacchi visti precedentemente, nel senso che quando un utente deve scegliere una password, tipicamente quest'ultimo prende una parola facile da ricordare e poi applica delle regole di sostituzione (per esempio sostituiscono delle lettere della parola). Gli attaccanti, quindi, con questa tipologia di attacco, vanno a prendere le parole da un dizionario (quindi da una lista di parole comuni) e poi gli applicano delle regole che catturano queste strategie di sostituzione adottate dagli utenti per formulare le proprie password. Ne faccio poi l'hash e lo confrontano con il valore degli hash presenti nel password file.

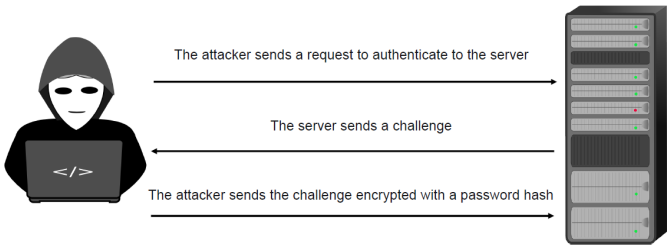
- Un modo per velocizzare il processo di calcolo degli hash è di utilizzare le **Rainbow Table** → esse permettono di avere per ogni parola presente all'interno di un dizionario il corrispondente hash calcolato secondo diversi algoritmi comunemente utilizzati per memorizzare la password all'interno del password

file → da un punto di vista della computazione, abbiamo che attraverso le Rainbow Table l'attacante risparmia tempo, dato che gli hash sono già calcolati. Dal punto di vista della memoria, le Rainbow Table richiedono un ingente spazio di memoria;

- Un'altra tecnica che sta sempre di più prendendo piede è quella del **Pass-the-Hash** → rappresenta una variante del Brute Force Attack e colpisce maggiormente i sistemi Windows, perchè quest'ultimi supportano il protocollo di autenticazione NTLM. Questo protocollo di autenticazione veniva utilizzato per autenticare gli utenti che volevano accedere a Server Windows nelle versioni precedenti a Windows 2000 ed era un protocollo di tipo challenge-and-response, ovvero: Quando l'utente si vuole autenticare, il Server di autenticazione gli invia una challenge (solitamente questa challenge è un valore completamente casuale di 16 byte). L'utente, per autenticarsi, cifra la challenge con l'hash della propria password. Il Server farà la stessa operazione che ha fatto l'utente, ovvero calcolare il valore cifrato della challenge che ha inviato e se i due valori (ovvero il valore inviato dall'utente e quello calcolato dal Sever) corrispondono, allora l'utente è autenticato con successo. Teoricamente, questo protocollo NTLM non dovrebbe più essere utilizzato per autenticarsi ad un Server Windows (oggi si utilizza un protocollo più sicuro di nome **Kerberos**), però quando l'autenticazione con Kerberos non funziona, Windows tenta di utilizzare l'autenticazione con NTLM.

Come può l'attaccante, allora, sfruttare il protocollo NTLM? L'attaccante, innanzitutto, deve ottenere accesso alla macchina di una vittima mediante, ad esempio, phishing o mediante attacchi che sfruttano delle vulnerabilità. Una volta che ha ottenuto l'accesso alla macchina della vittima, l'attaccante può:

1. installare Mimikats sulla macchina della vittima e fare il dump delle hash delle password di tutti gli utenti collegati alla macchina;
2. elevare i propri privilegi → per fare questo, l'attaccante inizia una richiesta, al Server di autenticazione, per creare un nuovo account di amministratore sulla macchina compromessa. Per confermare la creazione dell'account amministratore, l'authentication Server deve verificare la mia identità e per fare questo, il protocollo NTLM ci invia una challenge e l'attaccante firma la challenge con l'hash della password dell'utente amministratore della macchina di cui ho ottenuto il controllo (dato che con Mimikats ho il dump delle hash delle password). Cifro, quindi, la challenge con la password dell'amministratore e in questo modo l'attaccante viene autenticato come utente amministratore.



Il modo per verificare quanto una password è robusta ad attacchi di Brute Force è di utilizzare il concetto di **entropia** → l'entropia ci permette di misurare il livello di casualità presente all'interno di una password, dato che un livello di casualità più alto comporta chiaramente una password più difficile da indovinare o crackare per un attaccante. La formula che viene utilizzata per misurare la robustezza di una password è $log2(|A|^n)$, dove:

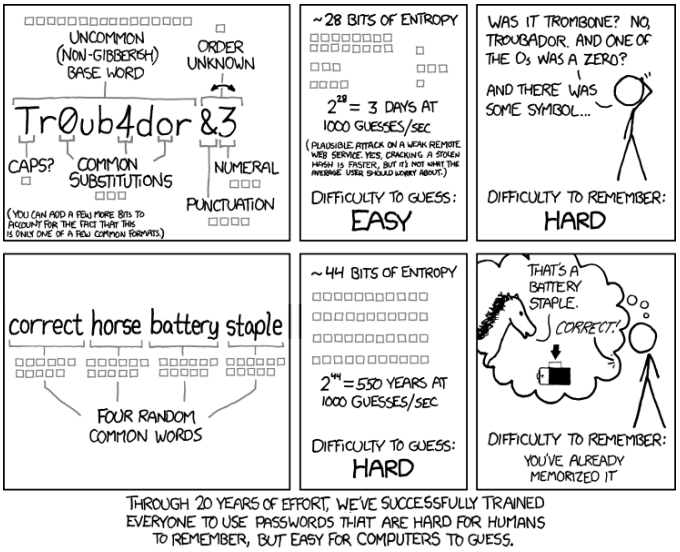
- A = insieme di simboli che formano la password;
- n = lunghezza della password.

Quindi, per esempio:

- se la password contiene lettere minuscole, allora il numero di caratteri che la compongono (ovvero |A|) sarà: |A| = 26;
- se supponiamo, inoltre, che la lunghezza della password sia 8;
- allora l'entropia sarà $log2(26^8) = 37.60$ bit e quindi la password è debole.

> 💡 Una password forte ha una entropia di almeno 60 bit

La misura basata sull'entropia per stimare la sicurezza della password è una buona misura? La risposta è **no**, perchè non tiene conto dei pattern maggiormente utilizzati dagli utenti, come ad esempio il fatto che alcuni caratteri della password vengono sostituiti con certi simboli speciali oppure che i numeri vengono inseriti in fondo alla password e molto spesso si ottengono password difficilmente ricordabili per l'utente. Scegliere, invece, quattro parole casuali come password, comporta l'ottenimento di una password facilmente ricordabile per l'utente e difficilmente crackabile per gli attaccanti, dato che ha un alto livello di entropia. Questo concetto può essere ben espresso attraverso la seguente figura:



Per stimare la robustezza di una password, rispetto agli attacchi visti precedentemente, possiamo utilizzare un'alternativa proposta da Dropbox chiamata **zxcvbn** → essa tiene conto dei pattern, che gli utenti seguono per creare una password. L'assunzione che viene fatta, è che una password tendenzialmente è una combinazione di una o più stringhe che adottano i pattern detti precedentemente (quali ad esempio: date, caratteri ripetuti, parole della tastiera) e poi va a stimare il numero di tentativi, che un attaccante dovrebbe fare per indovinare la stringa, utilizzando un dizionario con elencate in ordine decrescente le parole più comunemente utilizzate, che va a formare la password e che adotta uno specifico pattern.

Sorge spontanea una domanda: "**Come facciamo a proteggerci da queste tipologie di attacchi?**" Le possibili contromisure sono:

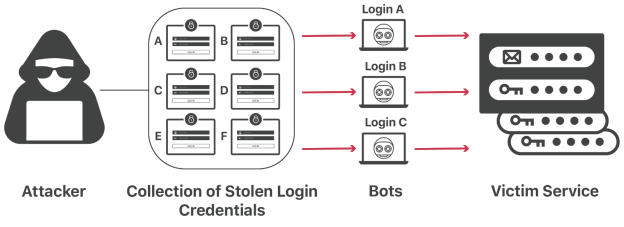
- abbiamo detto, che questi attacchi richiedono l'accesso al password file e di conseguenza dobbiamo garantire, che soltanto gli amministratori abbiano i privilegi per accedere al password file;
- abbiamo visto che è possibile elevare i propri privilegi e quindi ottenere l'accesso come amministratore. In questa eventualità, l'unica cosa da fare è associare un meccanismo di log, il quale tiene traccia di chi accede al password file, in modo da vedere se ci sono attività inusuali;
- un'altra contromisura è di aumentare i tentativi che gli attaccanti devono provare ed in questo senso, quello che si può fare è di non memorizzare all'interno del file solamente l'hash della password, ma aggiungere alla password anche un valore casuale (detto **salt**) e calcolare l'hash della password più il valore casuale → attraverso questo meccanismo, l'attaccante non avrà più successo con gli

attacchi di Brute Force e con i Dictionary Attacks, perchè nel momento in cui l'attaccante va a prendere una parola comune, nel caso del Dictionary Attack, ne calcola l'hash e la confronta con l'hash della password, l'attaccante non troverà mai una corrispondenza tra l'hash della parola comune e l'hash della password nel password file (soprattutto se vado a modificare la grandezza del salt);

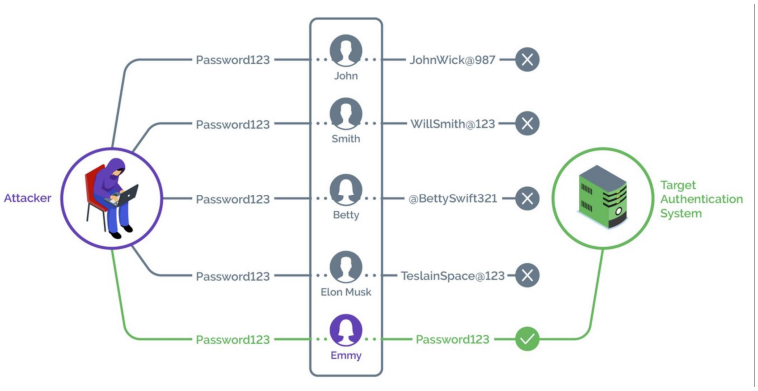
- nel momento in cui l'attaccante è riuscito a compromettere la password, l'utente deve avere a disposizione un meccanismo veloce per resettare la password, in modo da fermare l'attaccante.

Passiamo adesso agli attacchi condotti online, in cui l'untente non ha accesso al password file, ma quello che può sapere è tipicamente una lista di username che hanno accesso al sito/applicazione a cui gli attaccanti vogliono accedere. In questo caso, gli attaccanti tipicamente provano parole da un dizionario e tentano di combinare gli username noti con le parole di tale dizionario e le provano tutte fino a quando non riescono l'accesso. Un'altra strategia può essere quella di creare delle password (supponendo che l'attaccante abbia conoscenza di chi siano le vittime e che abbia accesso ai profili sulle reti sociali delle vittime) utilizzando informazioni pubbliche presenti sulle reti sociali delle vittime → naturalmente non vi è la garanzia che l'attaccante riesca a recuperare le password delle vittime.

Un'altra tecnica utilizzata è il **Credential Stuffing** → sfrutta il fatto, che gli utenti utilizzino le stesse credenziali per accedere a diversi siti online. Queste credenziali sono ampiamente disponibili agli attaccanti, in quanto le credenziali cadute vittime di Data breach vengono vendute nel Dark Web o comunque rese pubbliche. Questo attacco, quindi, consiste innanzitutto nel creare una lista dei file contenenti le milioni di coppie <username-password>, che sono state rese pubbliche in vari Data breach. Successivamente, gli attaccanti provano tutte queste coppie <username-password> per autenticarsi a diversi siti fino a quando non hanno successo → per riuscire a condurre questi tipi di attacchi, gli attaccanti utilizzano dei bots, ovverosia macchine sotto il controllo dell'attaccante e che generano in parallelo molteplici richieste di autenticazione utilizzando le coppie <username-password> presenti nella lista.



Una variante del Credential Stuffing è il **Password spraying** → il Password spraying sfrutta il fatto che gli utenti utilizzino la stessa password per autenticarsi in diversi siti, ma anche il fatto che utilizzino patter comuni per creare tali password. Gli attaccanti, quindi, prendono una o più password comunemente utilizzate dagli utenti e le utilizzano per autenticarsi presso diversi account, che hanno accesso al sito/applicazione obiettivo dell'attacco.



Per combattere questa tipologia di attacchi online, ci impongono che:

- la password deve rispettare determinati requisiti (come per esempio, che la password deve avere una lunghezza minima e un certo numero di caratteri speciali);
- la password deve essere cambiata ad intervalli temporali regolari (30/60/90 giorni).

Queste misure non sono misure effettivamente efficaci contro gli attacchi online, in quanto imporre questi requisiti porta gli utenti ad adottare le strategie viste precedentemente (pattern comuni, parole comuni, password cambiate solamente di un carattere), ovverosia comportamenti prevedibili per gli attaccanti.

Le effettive contromisure verso queste tipologie di attacchi sono le seguenti:

- **Meccanismi di blocco** → blocco dell'utente dopo diversi tentativi di accesso non riusciti;
- **Introdurre un ritardo** tra i tentativi di login consecutivi falliti;
- **Monitoraggio della sicurezza** → ovvero monitorare i login per rilevare attività insolite;
- **Notificare all'utente i dettagli del tentativo di login**;
- **Lista nera delle password** → ovverosia verificare se la password è presente in una lista di parole comuni.

Gli attacchi **passivi online** sono tipicamente di due tipi:

1. **Attacchi Man-in-the-middle** → ovverosia l'attaccante ha accesso alla rete ed è in grado di intercettare il traffico (generato dal browser) della vittime verso il sito in cui si vuole autenticare. Questo è possibile, generalmente, per il fatto che non è stato adottato alcun meccanismo di cifratura per proteggere la comunicazione tra il browser e il sito;
2. **Installare un Keylogger sulla macchina della vittima** → esso è in grado di catturare tutti i caratteri digitati sulla tastiera, i quali poi vengono salvati su un file locali e periodicamente il file viene inviato all'attaccante attraverso un canale di C&C.

Infine, possiamo trovare gli attacchi di ingegneria sociale, i quali possono essere:

- phishing;
- shoulder-surfing;
- dumpster-diving.

Autenticazione multi-fattore

In concomitanza con i sistemi di autenticazione basati su password, quello che generalmente si fa è: Si autentica l'utente sulla base di una password (quindi sulla base di un segreto) e poi sulla base di un **secondo fattore**. Il secondo fattore può includere:

- codici PIN o una stringa di caratteri, spesso inviati all'utente tramite SMS o email;
- un token di sicurezza che l'utente deve collegare fisicamente al proprio dispositivo (ad esempio tramite USB);
- dettagli biometrici (come ad esempio, la scansione dell'impronta digitale o il riconoscimento facciale);
- un'applicazione si un dispositivo fidato (come quelli forniti da Google).

L'approccio quello più comune, è quello che combina la password con una OTP (One Time Password), ovverosia password generate da un dispositivo in possesso dall'utente. L'autenticazione a 2 fattori deve essere fatta in tutte quelle attività online compiute dall'utente, che possono comportare un impatto negativo per l'utente stesso (per esempio, se l'utente vuole accedere all'online banking, è il caso che sia

richiesta l’autenticazione a 2 fattori). In particolar modo, vi sono due metodi ampiamente utilizzati per ottenere la OTP:

1. metodo basato su **SMS** → attraverso questo metodo, ogni volta che l’utente accede, quest’ultimo riceve un messaggio di testo al numero di telefono registrato, che contiene una OTP → questo metodo è vulnerabile ad attacchi di SIM swapping, ovverosia l’attaccante grazie alle informazioni della vittima che è riuscito a ricavare, può intestarsi una nuova SIM a nome della vittima e di conseguenza il messaggio per autenticarsi arriva direttamente all’attaccante → di conseguenza è più sicuro il metodo basato su TOTP;
2. metodo basato su **TOTP** → in particolare, all’utente viene chiesto di scansionare un’immagine QR utilizzando un’applicazione specifica per smartphone e successivamente questa applicazione genera la OTP per l’utente.

Gli algoritmi utilizzati per generare le password temporanee sono:

- **HMAC-Based One Time Password (HOTP)** → questo algoritmo può essere utilizzato solamente se il Server che esegue l’autenticazione e il dispositivo utilizzato per generare la OTP condividono un segreto, che solitamente si tratta di una chiave a cifratura simmetrica. Come funziona questo algoritmo:
  - innanzitutto l’utente deve abilitare l’autenticazione a due fattori;
  - il Server di backend crea una chiave segreta per quel particolare utente e il Server condivide la chiave segreta K con l’applicazione telefonica dell’utente;
  - l’applicazione telefonica inizializza un contatore C;
  - l’applicazione telefonica incrementa prima il valore del contatore di 1 e poi genera una password unica utilizzando la chiave segreta e il contatore;
  - se il valore inviato dall’applicazione telefonica corrisponde a quello generato da Server, quest’ultimo incrementa di 1 il valore del contatore.
- **Time-based One Time Password** → l’unica differenza rispetto all’algoritmo precedente è che HOTP utilizza il concetto del contatore, mentre Time-based One Time Password utilizza il concetto del **tempo**. In particolar modo, abbiamo che l’applicazione del telefono e il Server non hanno bisogno di inizializzare il contatore e di tenerne traccia, poichè il Server e il telefono hanno entrambi accesso all’ora. L’applicazione telefonica e il Server devono essere in grado di ricavare l’ora corrente Unix (ovvero il numero di secondi trascorsi dalla mezzanotte del 1° gennaio 1970) per la generazione della OTP.

Dati biometrici

Per biometria si intende qualsiasi misura utilizzata per identificare in modo univoco una persona sulla base di tratti biologici o fisiologici. In genere, i sistemi biometrici incorporano una sorta di sensore o scanner per leggere le informazioni biometriche e confrontarle con i modelli memorizzati degli utenti accettati prima di concedere l'accesso. Nella vita normale, i sensori biometrici possono essere utilizzati:

- per sbloccare il proprio telefono;
- per il controllo dei passaporti negli aeroporti;
- per accedere a determinate aree;
- per comunicare con l'auto.

I requisiti per l'autenticazione biometrica sono:

- **Universalità** → nel senso che ogni persona dovrebbe avere questa caratteristica;
- **Distintività** → ogni persona dovrebbe presentare differenze evidenti nella caratteristica;
- **Permanenza** → la caratteristica non deve cambiare significativamente nel tempo;
- **Collezionabilità** → la caratteristica deve poter essere efficacemente determinata e quantificata.

Alcuni esempi di caratteristiche utilizzabili per il riconoscimento biometrico sono:

- la firma;
- l'impronta digitale;
- scansione della retina;
- DNA;
- riconoscimento vocale;
- riconoscimento del volto;
- riconoscimento della camminata.

"**Come funziona l'autenticazione tramite dati biometrici?**" Come per i sistemi di autenticazione basati su password, anche in questo caso vi sono due fasi:

1. una fase di **registrazione** → l’utente che si vuole registrare, ad esempio attraverso l’impronta digitale, utilizza un lettore di impronte digitali, il quale prende un campione dell’impronta digitale dell’utente e lo digitalizza, salvandolo come un vettore, ovverosia come una sequenza di bit → questo diventa il template di riferimento per autenticare l’utente;
2. una fasi di **autenticazione** → in questa fase, abbiamo che l’utente scannerizza, tramite il lettore, la propria impronta digitale che viene nuovamente digitalizzata e confrontata con il template di riferimento. Se le due impronte corrispondono l’utente viene autenticato, altrimenti gli viene negato l’accesso.

Le **limitazioni** dell’autenticazione biometrica sono:

- la precisione dell’algoritmo di corrispondenza;
- falsi positivi, ovverosia consentire l’accesso ad utenti non autorizzati;
- falsi negativi, ovverosia rifiutare l’accesso ad utenti autorizzati;
- il fatto che le impronte digitali vengono lasciate in molti punti;
- bassa accettazione da parte degli utenti.