



Cofinanziato
dall'Unione europea



PASSWORD MANAGEMENT

Sicurezza informatica

Elena Maria Dal Santo

elenamaria.dalsanto@its-ictpiemonte.it





Autenticazione e
autorizzazione

“Prozioni anonimi”



Immaginate di dover condividere un file su Google Drive con un vostro amico, in modo da poterlo modificare entrambi.

Scegliete la condivisione tramite link, comoda e veloce, e inviate il link al vostro amico.

L'amico si dimentica di dover aggiungere le sue modifiche e parte per andare in vacanza, lasciando a casa il pc su cui ha l'account Google.

“Prozioni anonimi”

Mentre si gode la bella vita, sul lettino in spiaggia, si ricorda dei vostri accordi e, dal cellulare, accede al file su Google Drive tramite il link che avevate condiviso.

Poniamo per assurdo che quel cellulare non sia collegato a nessun account Google.

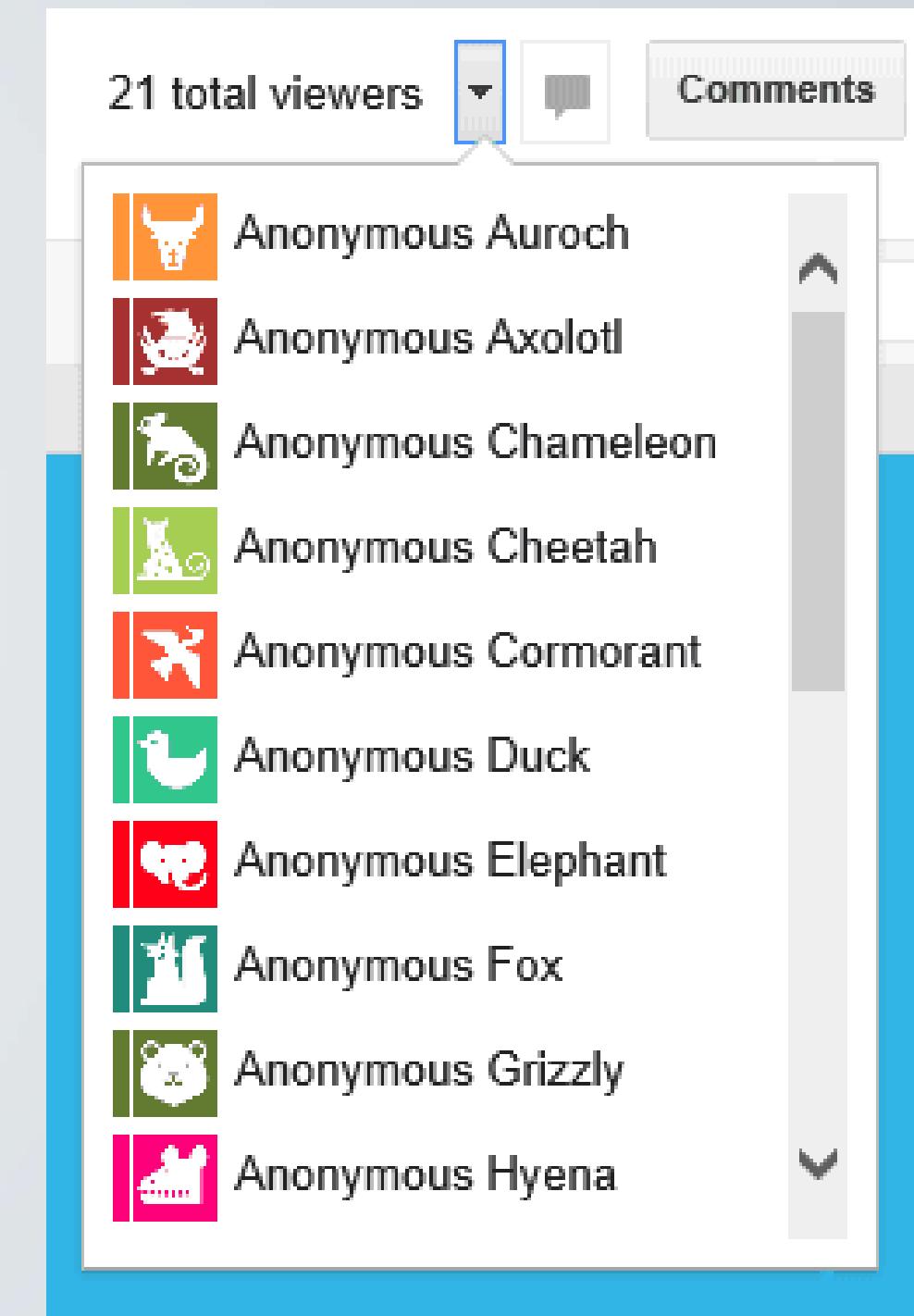
Il vostro amico potrà comunque entrare e modificare il file.



“Prozioni anonimi”

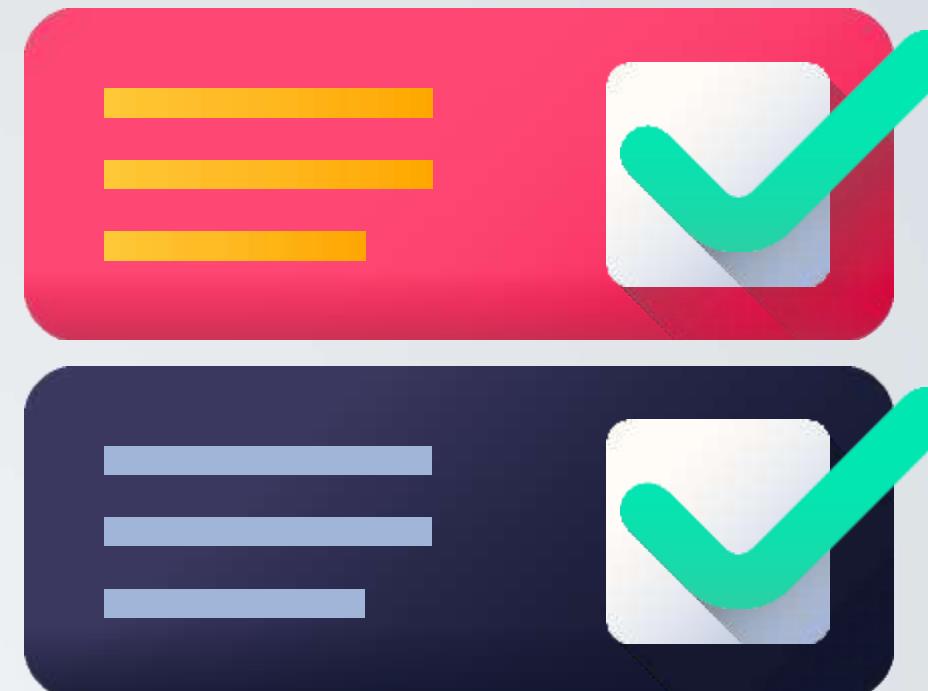
Questo perché Google Drive consente l'autorizzazione senza bisogno di **identificazione**.

- **AUTORIZZAZIONE** → ti concedo privilegi di accesso
- **IDENTIFICAZIONE** → è l'atto di indicare l'identità di una persona o una cosa
- **AUTENTICAZIONE** → è l'atto di confermare la veridicità di un'informazione (es. identità) tramite l'utilizzo di un attributo (es. password)

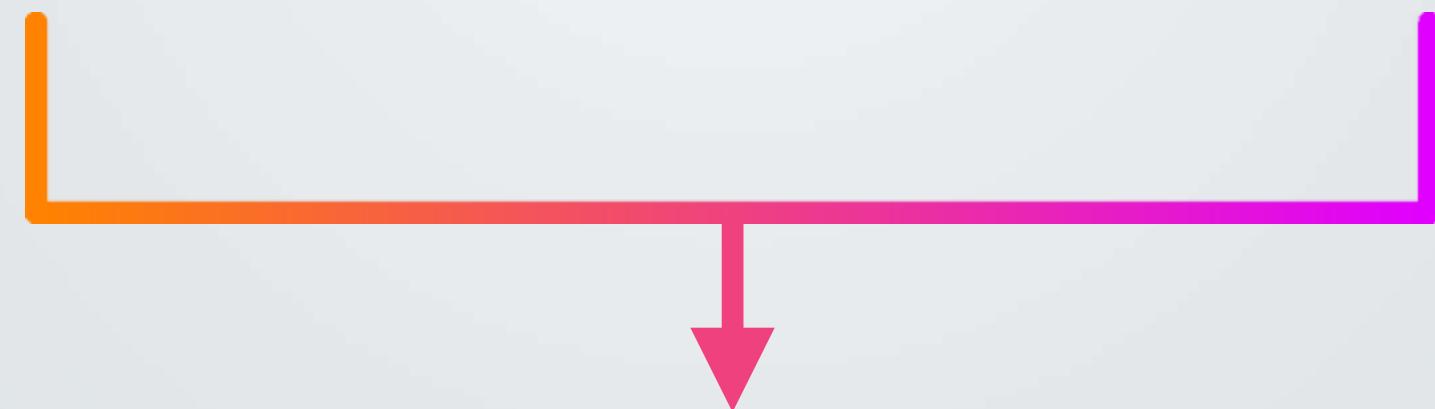




Prevenzione dalle intrusioni



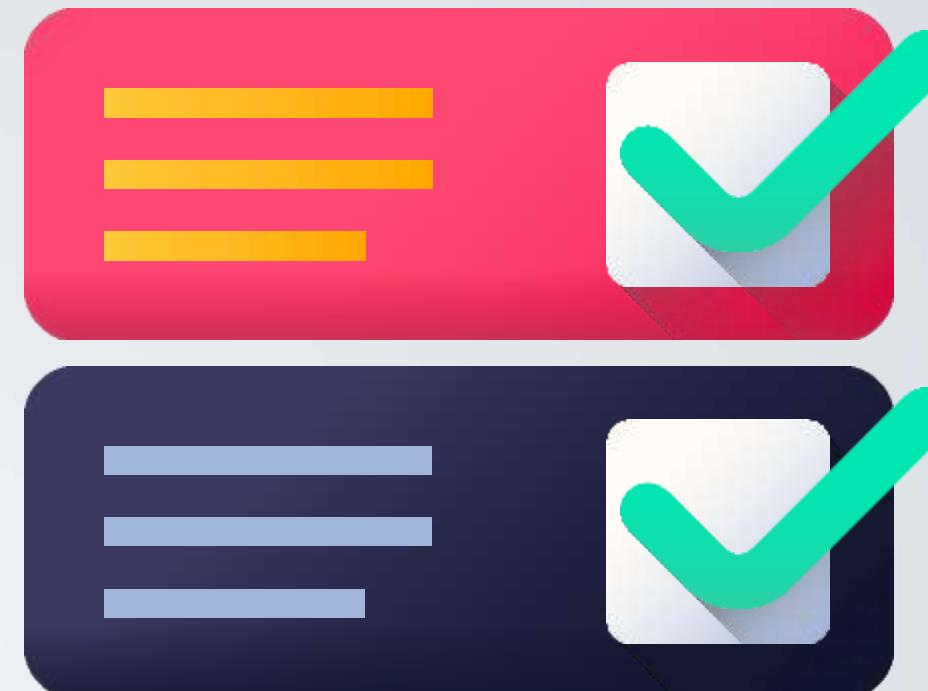
Autenticazione Autorizzazione



Controllo degli accessi



Prevenzione dalle intrusioni



Autenticazione Autorizzazione

JWT (Json Web
Token)

Oauth (Open
Authorization)

Controllo degli accessi

JWT (Json Web Token)

È un token che contiene al suo interno anche informazioni sull'utente/client

Consente **autenticazione** + eventuale scambio di **informazioni**

Molto utilizzato perché viene mappato su un JSON ed è quindi facilmente trasformabile in un oggetto.

JWT (Json Web Token)

Un JWT è formato da 3 parti:

- HEADER
- PAYLOAD
- SIGNATURE (firma)

Non è cifrato, è codificato in Base64 (i bit che lo compongono vengono mostrati con caratteri ASCII)

The screenshot shows the jwt.io debugger interface. At the top, it says "ALGORITHM HS256". The "Encoded" section contains the Base64 string: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfjoYZgeF0NFh7HgQ. The "Decoded" section shows the JSON structure of the token:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

The "VERIFY SIGNATURE" section contains the code: HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
secret
) secret base64 encoded. A blue button at the bottom right says "Signature Verified".

JWT (Json Web Token)

HEADER

Composto da 2 parti:

- **alg**: algoritmo utilizzato per la firma
- **typ**: tipo di token (JWT)

The screenshot shows a web-based JWT debugger at jwt.io. The interface has two main sections: 'Encoded' and 'Decoded'. The 'Encoded' section displays the base64-encoded JWT string:
eyJhbGciOiJIUzI1NiIsInR5cCI6
IkpxVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6Ikpvag4gRG9
1IiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfjoYZge
F0NFh7HgQ

The 'Decoded' section shows the JSON structure of the token. A red box highlights the 'HEADER' field, which contains:
{
 "alg": "HS256",
 "typ": "JWT"
}
The 'PAYLOAD' field contains:
{
 "sub": "1234567890",
 "name": "John Doe",
 "admin": true
}
Below the decoded fields, there is a 'VERIFY SIGNATURE' section with code for HMACSHA256:
HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 secret
)
 secret base64 encoded

A large blue button at the bottom right says 'Signature Verified' with a checkmark icon.

JWT (Json Web Token)

PAYLOAD

Contiene i **claims**, cioè le informazioni riguardanti un'entità (solitamente il client) ed eventuali altri dati.

- Registered claims
- Public claims
- Private claims

Tutti i claims sono facoltativi.

The screenshot shows the jwt.io debugger interface. On the left, under 'Encoded', is a long string of characters: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfjoYZgeF0NFh7HgQ. A red box highlights the 'PAYLOAD:' section, which contains the following JSON:

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Below the payload, under 'VERIFY SIGNATURE', is a code snippet for HMACSHA256:

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)
```

At the bottom, a blue button indicates 'Signature Verified'.

JWT (Json Web Token)

PAYLOAD

Registered claims

Registrati all'interno di un registro (IANA Json Web Token Claim Register). Sono degli standard (es “iss” per issuer, “exp” per expiration time)

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6  
IkpxVCJ9.eyJzdWIiOiIxMjM0NTY  
3ODkwIiwibmFtZSI6Ikpvag4gRG9  
lIiwiYWRtaW4iOnRydWV9.TJVA95  
OrM7E2cBab30RMHrHDcEfjoYZge  
F0NFh7HgQ
```

Decoded

HEADER:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT:

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)  secret base64 encoded
```

Signature Verified

JWT (Json Web Token)

PAYLOAD

Public claims

Definibili a proprio piacimento, senza alcuna restrizione (es. nome, email, ecc).

Per evitare collisioni nella semantica delle keys, devono comunque poi essere registrati.

The screenshot shows the jwt.io debugger interface. On the left, under 'Encoded', is a long string of characters: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfjoYZgeF0NFh7HgQ. A red box highlights the 'PAYLOAD' section on the right, which contains the JSON: { "sub": "1234567890", "name": "John Doe", "admin": true }. Below the payload is a 'VERIFY SIGNATURE' section with code for HMACSHA256. At the bottom, a blue bar indicates 'Signature Verified' with a checkmark icon.

HEADER:
{ "alg": "HS256", "typ": "JWT" }

PAYOUT:
{ "sub": "1234567890", "name": "John Doe", "admin": true }

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)  secret base64 encoded
```

Signature Verified

JWT (Json Web Token)

PAYLOAD

Private claims

Pensati per lo scambio di informazioni specifico con quell'applicazione.

Più specifici di quelli pubblici (es. ID utente, reparto, ruolo, ecc).

Importante assicurarsi che il nome non collida con uno registrato o pubblico.

The screenshot shows the jwt.io debugger interface. In the 'Encoded' section, a long string of characters is displayed. In the 'Decoded' section, the token is broken down into its components. The 'HEADER' component contains the algorithm (HS256) and type (JWT). The 'PAYLOAD' component contains the following JSON object:

```
{"sub": "1234567890", "name": "John Doe", "admin": true}
```

A red box highlights the 'PAYLOAD' section. At the bottom of the page, a blue bar indicates that the 'Signature Verified'.

JWT (Json Web Token)

SIGNATURE

La firma viene applicata a encoded header + encoded payload + secret secondo l'algoritmo dichiarato nell'header.

Viene usata per verificare integrità dei messaggi e identità del mittente.

The screenshot shows the jwt.io debugger interface. At the top, the algorithm is set to HS256. The token is displayed in two sections: Encoded and Decoded. The Encoded section shows the raw token string: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfjoYZgeF0NFh7HgQ. The Decoded section shows the JSON structure of the token:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Below the token, there is a code editor for generating the signature:

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)
```

A red box highlights the "secret" input field. At the bottom, a blue button indicates "Signature Verified".

JWT (Json Web Token)

Header.Payload.Signature



Base64

XXXXXX.yyyyyy.zzzzzz

The screenshot shows the jwt.io debugger interface. At the top, the URL is "jwt.io". The "ALGORITHM" dropdown is set to "HS256".

Encoded: The token is shown as a single string of Base64 encoded data: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfjoYZgeF0NFh7HgQ

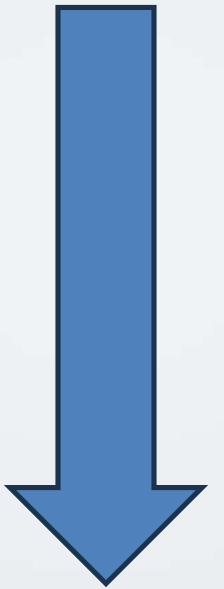
Decoded:

- HEADER:** { "alg": "HS256", "typ": "JWT" }
- PAYOUT:** { "sub": "1234567890", "name": "John Doe", "admin": true }
- VERIFY SIGNATURE**
- HMACSHA256(** base64UrlEncode(header) + "." + base64UrlEncode(payload), **secret**) **secret base64 encoded**

Signature Verified

JWT (Json Web Token)

Il token è “signed” (quindi firmato) ma le informazioni al suo interno non sono in alcun modo criptate



Non si devono inserire all'interno del token informazioni sensibili

JWT (Json Web Token)

Authorization: Bearer <token>

“Autorizza: chi ha il token <token>”

Anche per questo
deve essere
sufficientemente
breve!

Il token viene inviato come header nelle chiamate http per chiedere un'autorizzazione

Se il token è valido, viene concessa l'autorizzazione.

JWT (Json Web Token)

JSON Web Tokens - jwt.io

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA950rM7E2cBab30RMHrHDcEfijoYZgeF0NFh7HgQ
```

Decoded

EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)  secret base64 encoded
```

The screenshot shows the Samltool.io Debugger interface. At the top, there's a navigation bar with icons for back, forward, and search, followed by the URL "SAML - samltool.io" and the domain "samltool.io". On the right side of the header are social media links for Twitter and GitHub, and a follower count of "3,150 followers".

The main content area has a title "SAML" and a sub-section "Debugger". Below this, there are two large text boxes: "SAML ENCODED" on the left containing a long string of characters, and "SAML DECODED" on the right showing the same data as XML code. The XML code is color-coded by element type (e.g., green for tags, red for attributes). A checkbox labeled "Prettify (not editable)" is checked, and a button labeled "Expand" is visible.

At the bottom, there's a section titled "SAML INFO" which is currently empty.

JWT (Json Web Token)

Vantaggi:

- utilizza il formato **Json** → facilmente convertibile
- **piccolo** → può essere trasmesso negli header
- **signed** → sicuro perché firmato digitalmente
- **stateless** → non deve essere salvato, essendo in chiaro si possono verificare informazioni e firma. Questo implica, però, anche una certa difficoltà nel revocare il token, in caso di problemi!

JWT (Json Web Token)

Svantaggi evidenti:

- **signed** → se la chiave viene compromessa, un attaccante può usarla per generare un JWT che risulterà comunque valido
- **stateless** → è difficile revocare il token in caso di problemi. La possibilità di revocare un token va a braccetto con la creazione di una blacklist, che richiede tempo e manutenzione costante.

OAuth (Open Authorization)

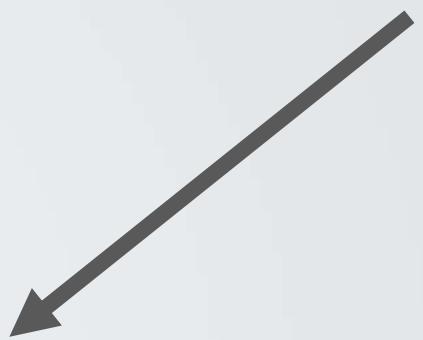
Una delle applicazioni dei JWT è nello standard OAuth

OAuth è, ad oggi, lo standard per **l'autorizzazione** online tra aziende.
L'autorizzazione funziona senza alcun bisogno di scambiarsi
credenziali o account.

Tramite l'utilizzo di un token viene consentito o meno l'accesso a determinate risorse. Solo il provider del token può verificare il token stesso.

OAuth (Open Authorization)

Questo standard prevede l'utilizzo di **access token**



Dato che rappresenta l'autorizzazione all'accesso per un determinato user (spesso questi token sono dei JWT)

Per questioni di sicurezza, gli access token hanno quasi sempre anche una scadenza (anche se non è obbligatorio)

OAuth (Open Authorization)

OAuth prevede 4 attori principali:

Resource owner

User o sistema che possiede le risorse e può fornire o meno autorizzazioni all'accesso

Client

User o sistema che richiede accesso alle risorse. Per ottenere l'accesso, deve possedere un access token.

Authorization server

Server che riceve dal client la richiesta di ottenere un access token e fornisce il token in seguito a una corretta autenticazione presso il resource owner

Resource server

Server che “protegge” le risorse. Si occupa di recepire il token e consentire o meno l'accesso alle risorse.

OAuth (Open Authorization)

Come funziona?

PRE: l'authorization server identifica il client associandoci un client_id e un client_secret

1. Il client richiede all'authorization server l'autorizzazione ad accedere a una determinata risorsa, fornendo client_id e client_secret come prova di autenticazione.
2. L'authorization server autentica il client e verifica gli **scopes** (le motivazioni per cui l'accesso a una risorsa dovrebbe essere concesso. Questi scope sono decisi e forniti dal resource server)

OAuth (Open Authorization)

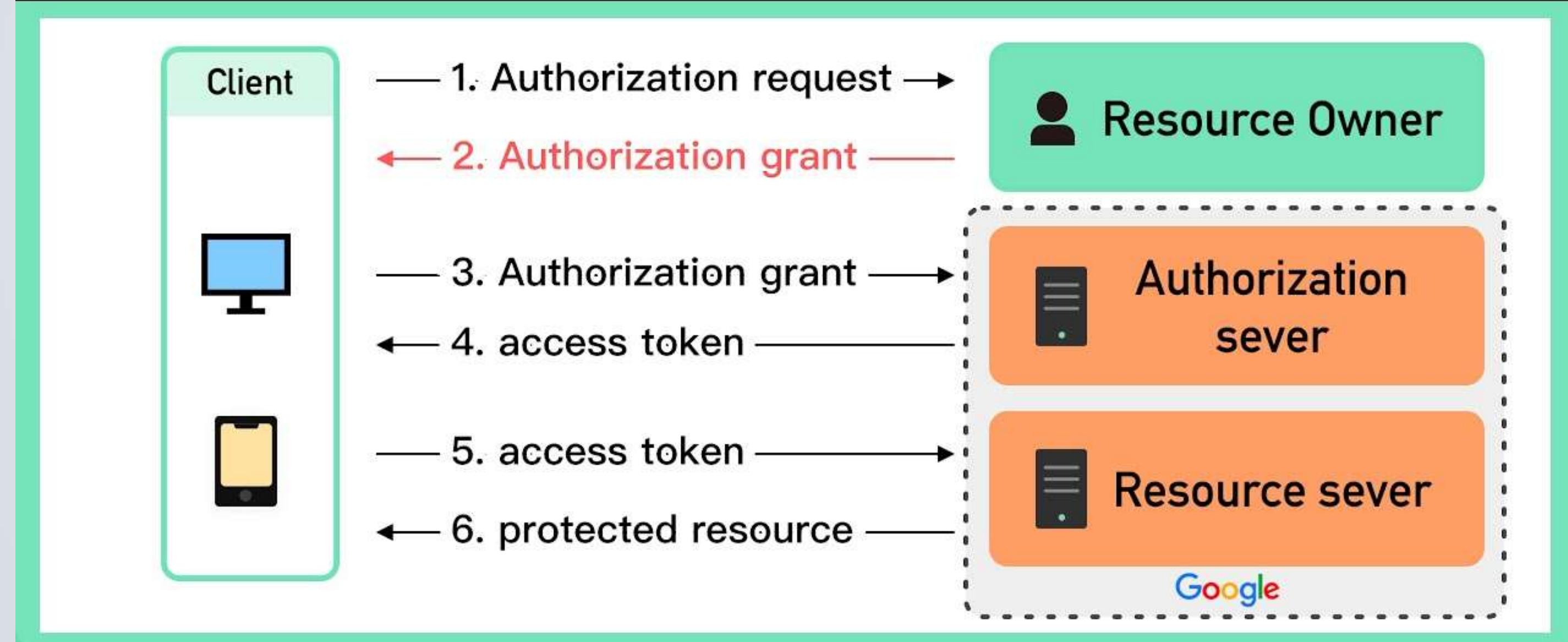
Come funziona?

3. Il resource owner interagisce con l'authorization server per autorizzare l'accesso
4. L'authorization server risponde al client inviandogli il token
5. Con quel token, il client può richiedere al resource server l'accesso alle risorse.

NB non sempre viene restituito direttamente il token di accesso, possono essere restituiti codici univoci di accesso con cui richiedere un token o anche un refresh token (con scadenza più lunga e utilizzabile per richiedere un nuovo access token). A ogni client sono associate delle **grant** che stabiliscono il numero di step che dovrà compiere per ottenere l'access token.

OAuth (Open Authorization)

| But What Is OAuth 2 Really About?



OAuth (Open Authorization)

A differenza di JWT, OAuth è uno standard **stateful**, in quanto richiede una connessione all'authentication server per ottenere e verificare i token.

Possono essere utilizzati JWT, ma non è obbligatorio.

Il provider di OAuth token sarà anche l'unico in grado di verificarli.

OAuth vs. JWT

Casi d'uso

JWT è molto usato per le API

OAuth può essere usato nel web, per le API, per applicazioni e risorse

Token

JWT definisce un formato token

OAuth definisce un protocollo di autorizzazione che può, eventualmente, fare uso di JWT

OAuth vs. JWT

Facilità d'utilizzo

JWT è più facile da imparare e da usare

OAuth può risultare più complicato, essendo più strutturato

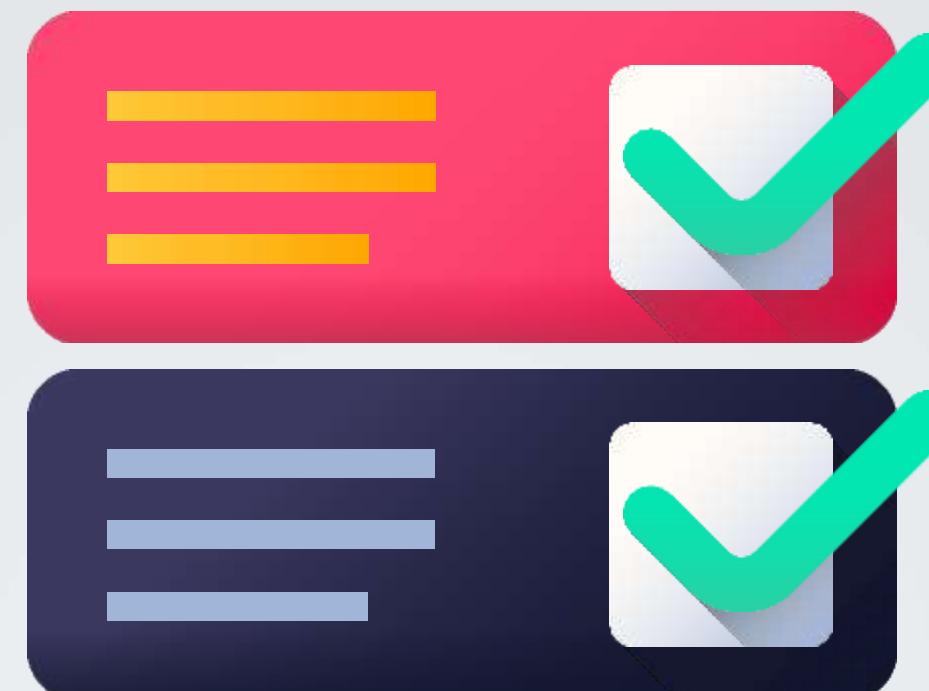
Memorizzazione

Stateless. L'unico storage che può usare è quello del client, il server non memorizza niente.

Stateful. Il server può memorizzare i token e le informazioni. Può ovviamente usare la memorizzazione anche lato client.



Prevenzione dalle intrusioni



Autenticazione

Autorizzazione

Crittografia

Controllo degli accessi → **PASSWORD**

Una **password** è una sequenza di lettere, numeri e simboli che viene usata per accedere in modo esclusivo a una risorsa informatica.

La password è il legame tra la nostra identità e la risorsa informatica stessa, e permette alla risorsa di identificarcici in modo univoco.



Chi trova una password...



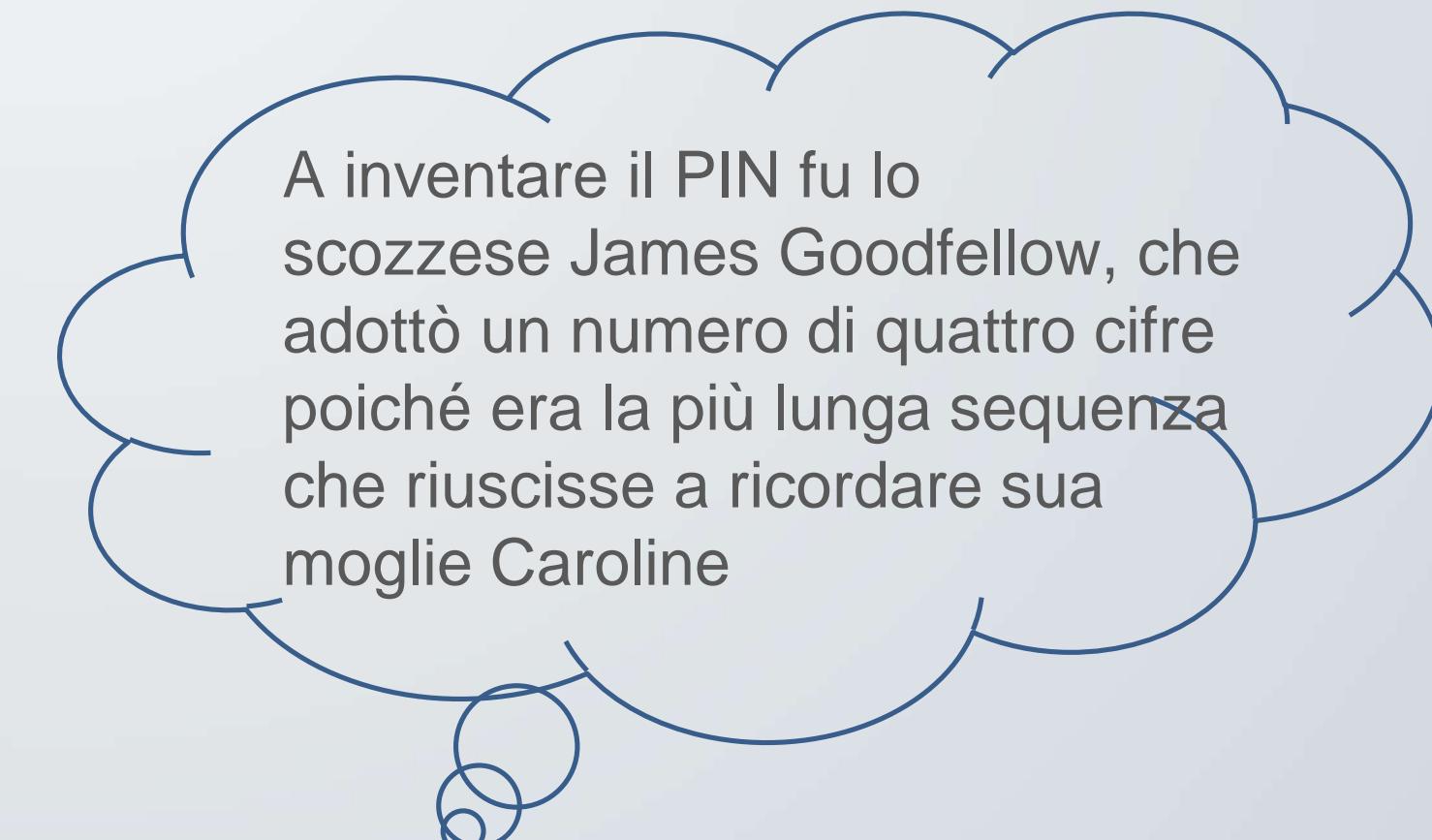
PIN e PUK

Il **PIN (Personal Identification Number)** è una sequenza di 4 cifre che viene utilizzata per verificare che una persona che accede a un dispositivo abbia le autorizzazioni necessarie a compiere quell'operazione.

Di fatto, è una password numerica, che conferma la proprietà sul dispositivo.

Dove?

- Cellulare
- Bancomat
- App
- ...



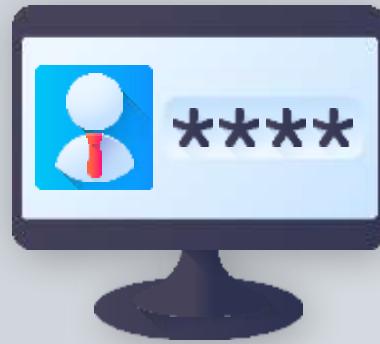
PIN e PUK

Il **PUK** (Personal Unblocking Key) è il codice di 8 cifre usato per sbloccare un dispositivo precedentemente bloccato (es. se inserite il PIN sbagliato per tre volte di fila)

Se si immette 10 volte il PUK consecutivo, la SIM viene bloccata definitivamente.



Tecniche di intrusione



Default password

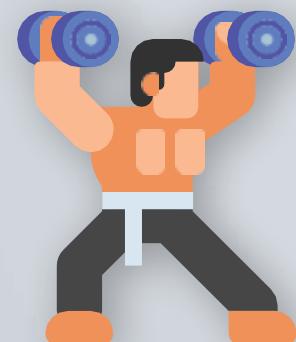
Es. <admin><admin>



Social engineering



Attacchi a dizionario



Attacchi a forza bruta

JohnTheRipper, AirCrack, ...



Attacchi “brute force”

L'**entropia** è una funzione matematica che stima quanto una password sia casualmente scelta. Ci serve per avere un'idea della complessità della password.

AAAAAAA

tanta ripetizione = poca entropia

RICATTARE

alcune ripetizioni = entropia media

J3SIVEK\$3

stringa casuale = entropia massima

Cos'è una password casuale?



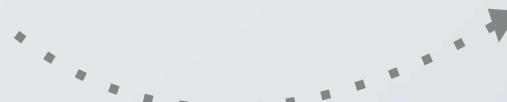
Una password generata casualmente è una stringa di caratteri di lunghezza L estratti per **selezione casuale** da un insieme di simboli Σ .

Quante sono le password di lunghezza L ?



Lo **spazio delle chiavi** è il numero di password possibili lunghe L , ed è

$$K = |\Sigma|^L$$



Le sbarrette verticali in questo contesto non indicano il valore assoluto, ma la **cardinalità** dell'insieme, ovvero quanti elementi contiene l'insieme.

Cos'è una password casuale?

Come si misura quanto è robusta una password?

$$H = \log_2 |\Sigma|^L = L \log_2 |\Sigma|$$

$$= L \frac{\log |\Sigma|}{\log 2}$$

$$\log_a x^k = k \cdot \log_a x$$

$$\log_b x = \frac{\log_k x}{\log_k b}$$



Entropia per simbolo ($L = 1$)

Set di simboli	Cardinalità	Entropia
Numeri (0-9)	10	3.332 bit
Esadecimale (0-9, A-F)	16	4 bit
Alfabeto latino (a-z)	26	4.7 bit
Alfanumerico (a-z, 0-9)	36	5.17 bit
Alfanumerico case sentive (a-z, A-Z, 0-9)	62	5.954 bit
ASCII stampabile tranne spazio	94	6.555 bit
Codice binario (1 byte)	256	8 bit

Esempio realistico

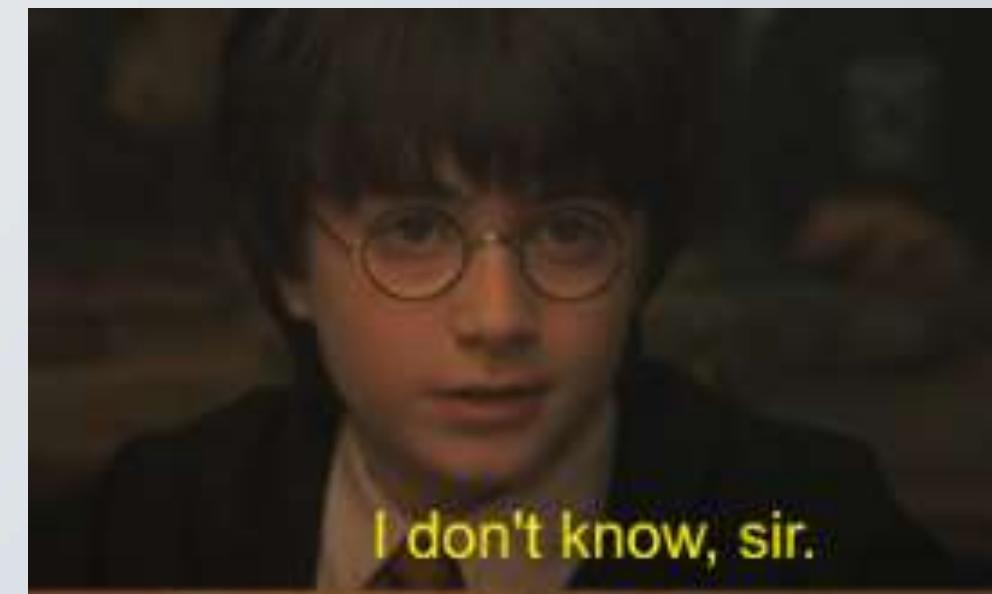
Supponiamo di generare casualmente (ovvero, ogni simbolo è equiprobabile) una password di lunghezza $L = 12$ con alfabeto $I \Sigma I = 94$, composto dai caratteri ASCII stampabili.

P6zfAm8@o#e4

L'entropia della password è calcolata con l'equazione

$$H = \log_2 K = \log_2 94^{12} = 12 \log_2 94 = 12 \frac{\log 94}{\log 2} = 78.655 \text{ bit}$$

Sono "buoni" 78 bit di entropia per una password?



Robustezza delle password *

< 28 bit

molto debole, potrebbe bastare per tenere lontano il vostro gatto.

28-35 bit

debole, tiene alla larga la maggior parte delle persone, ma non Luigino, l'hacker del piano di sopra. Potete usarla, al massimo, come password di login del pc.

36-59 bit

ragionevole, si può usare per la rete e in azienda.

60-127 bit

forte, potete usarla per proteggere i milioni di euro del vostro conto in banca.

128+ bit

overkill, manco fossi Tony Stark.

* Allo stato attuale della potenza computazionale

Tempi di attacco

Quando si effettua un attacco brute force, la lunghezza della chiave non è nota.

Si devono testare tutte le combinazioni da una lunghezza minima l ad una massima L .

$$K^* = \text{ILI}^l + \text{ILI}^{l+1} + \text{ILI}^{l+2} + \dots + \text{ILI}^L$$

The diagram illustrates the concept of a brute-force attack. It shows a mathematical expression for the set of all possible keys, K^* , as a sum of sets of passwords of increasing lengths. The first term, ILI^l , is enclosed in a green oval and has a green arrow pointing down to the text "Tutte le password di lunghezza l ". The second term, ILI^{l+1} , is also enclosed in a green oval and has a green arrow pointing down to the text "Tutte le password di lunghezze intermedie". This pattern continues with ellipses between the terms and a final term ILI^L enclosed in a green oval, which has a green arrow pointing down to the text "Tutte le password di lunghezza L ".

Tempi di attacco

Il tempo di crack t è dato da
$$t = K^* \cdot \frac{1}{V} \cdot \frac{1}{N}$$

V è la capacità computazionale (numero di chiavi testate al secondo);
 N è il numero di calcolatori usati in parallelo.



Tempi di attacco

Supponiamo (realisticamente) $N = 1$ e $V = 500\,000$.

Otteniamo:

L	(a-z)	(a-z,0-9)	(a-z,A-Z)	ASCII
≤ 4	immediato	immediato	immediato	2 minuti
5	immediato	2 minuti	12 minuti	4 ore
6	10 minuti	72 minuti	10 ore	18 giorni
7	4 ore	43 ore	23 giorni	4 anni
8	4 giorni	65 giorni	3 anni	463 anni
9	4 mesi	6 anni	178 anni	444530 anni

“Ma sì, dai, ormai lo sanno tutti...”

Password Security Awards



And the winner is...

1 123456
2 123456789
3 picture1

[Nordpass \(2020\) 275.69.9516 passwords evaluated.](#)

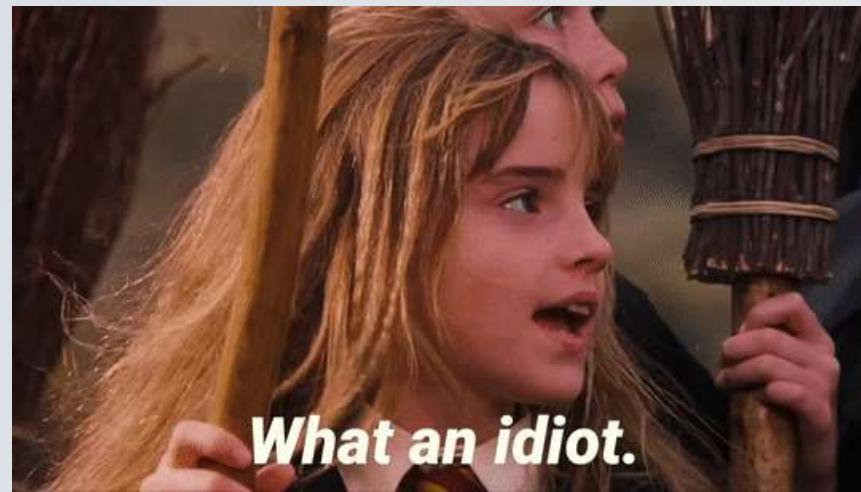
- | | |
|----------------|-----------------|
| 4 - password | 15 - 000000 |
| 5 - 12345678 | 16 - 1234 |
| 6 - 111111 | 17 - iloveyou |
| 7 - 123123 | 18 - aaron431 |
| 8 - 12345 | 19 - password1 |
| 9 - 1234567890 | 20 - qqww1122 |
| 10 - senha | 21 - 123 |
| 11 - 1234567 | 22 - omgpop |
| 12 - qwerty | 23 - 123321 |
| 13 - abc123 | 23 - 654321 |
| 14 - Million2 | 25 - qwertyuiop |

Ricerca euristica

Cosa succede se la
password non è casuale?



Ricerca euristica



Metodo stupido

AAAAAA
AAAAB
AAABA
AABAA

...

ZZZZZ

Metodo furbo

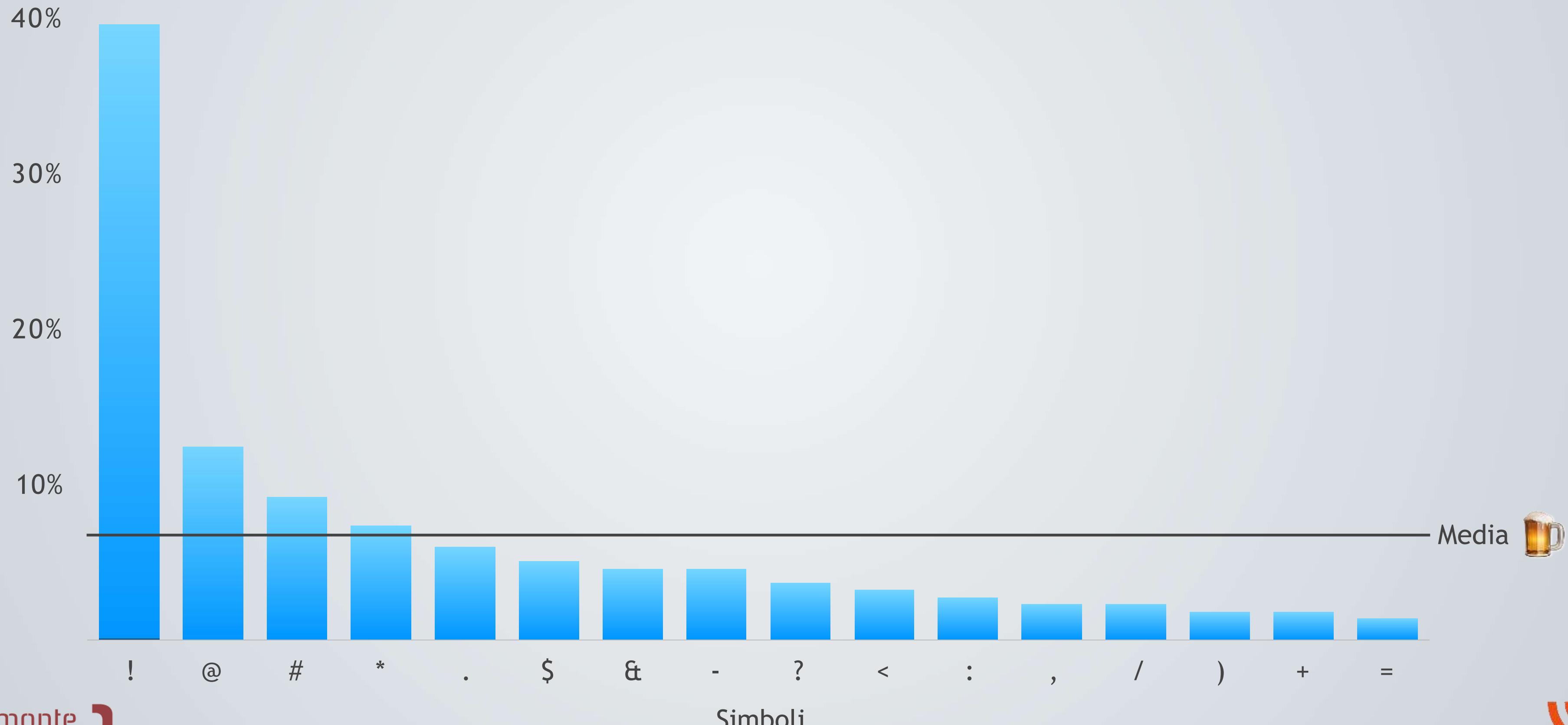
Love1!
StarWars12@
P4\$\$w0rd
qwerty#77

...

ZZZZZ

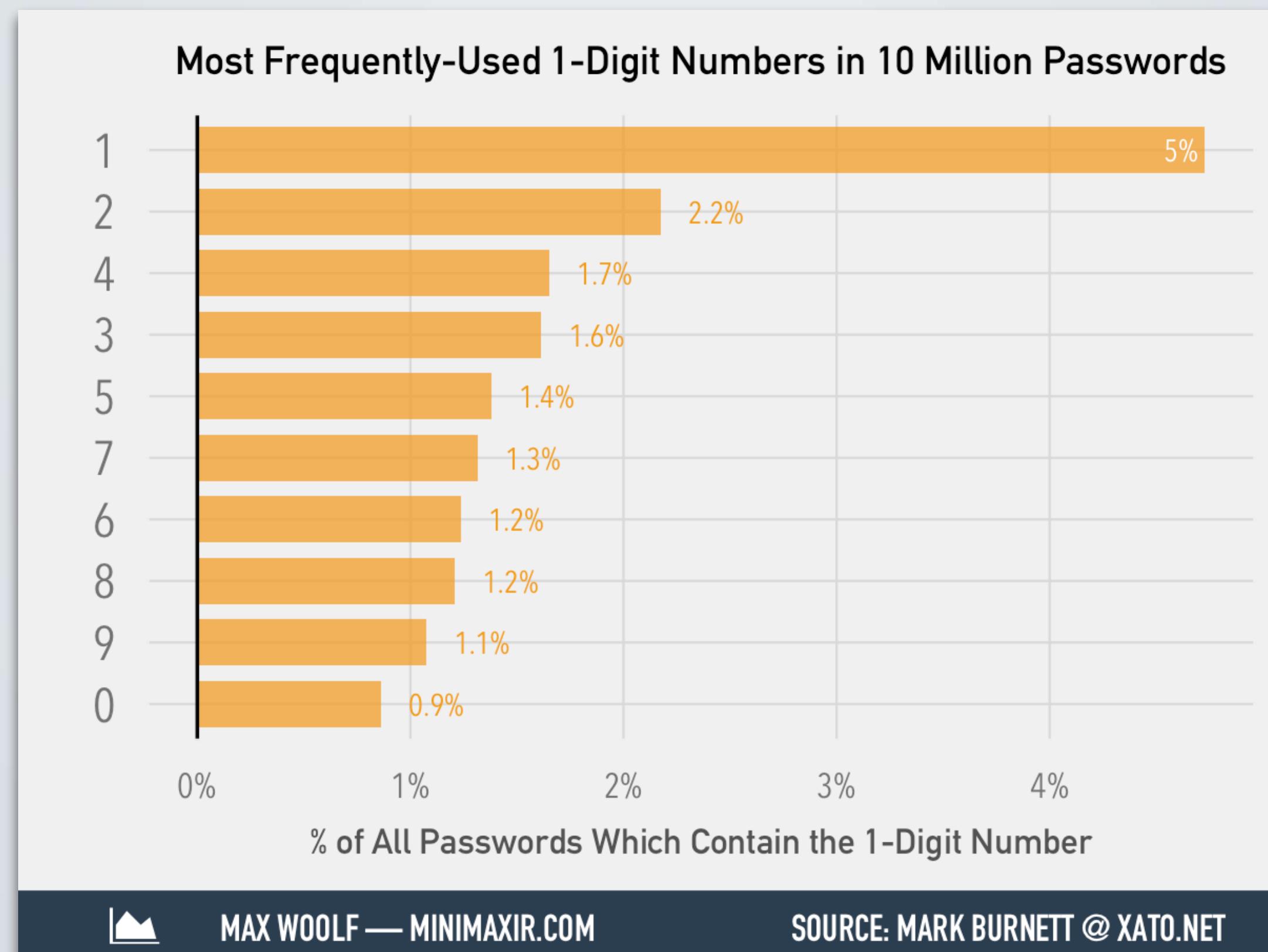
Ricerca euristica

“Aggiungo un **simbolo** alla fine così diventa sicura...”



Ricerca euristica

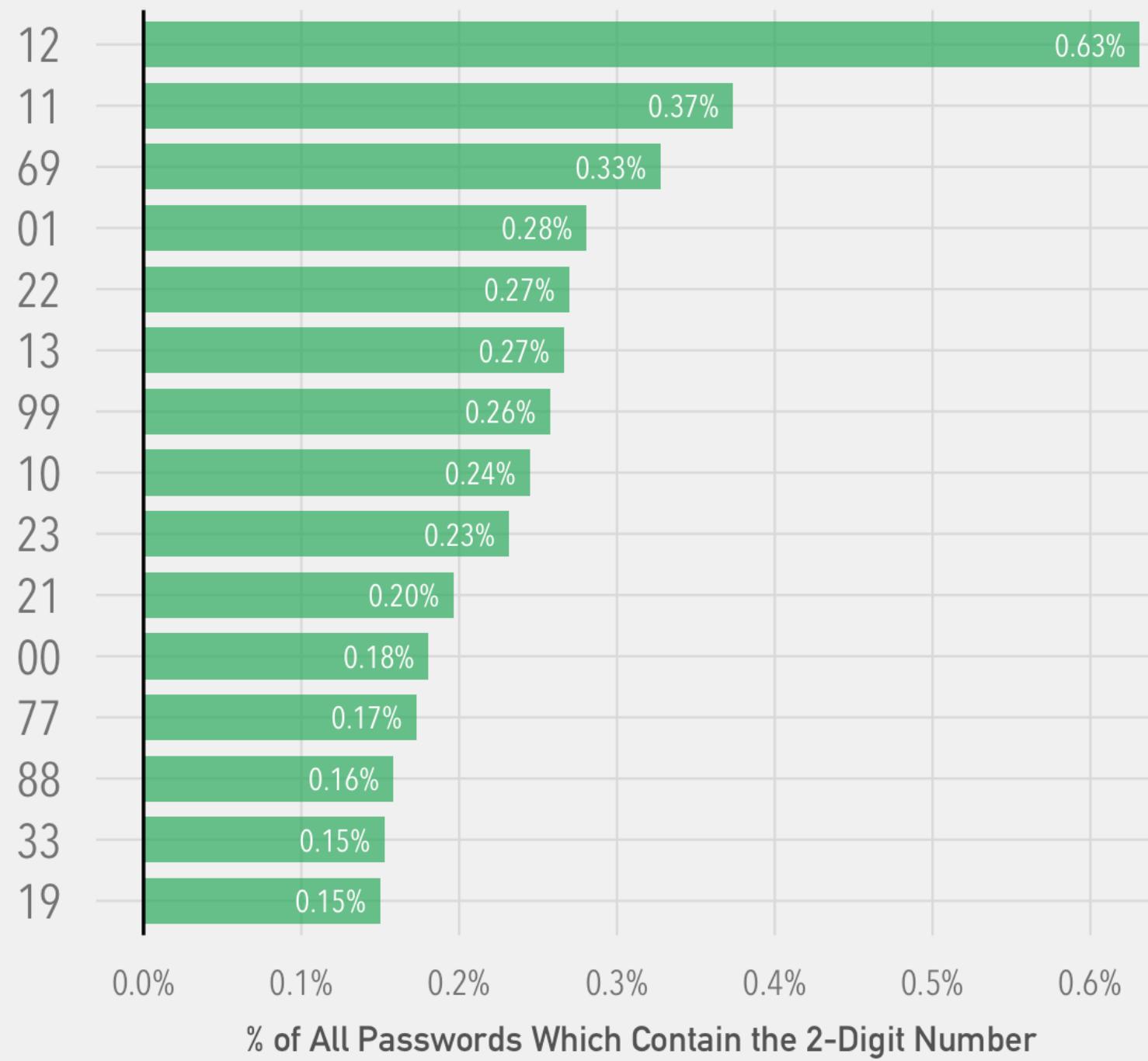
“Aggiungo un **numero** alla fine così diventa sicura...”



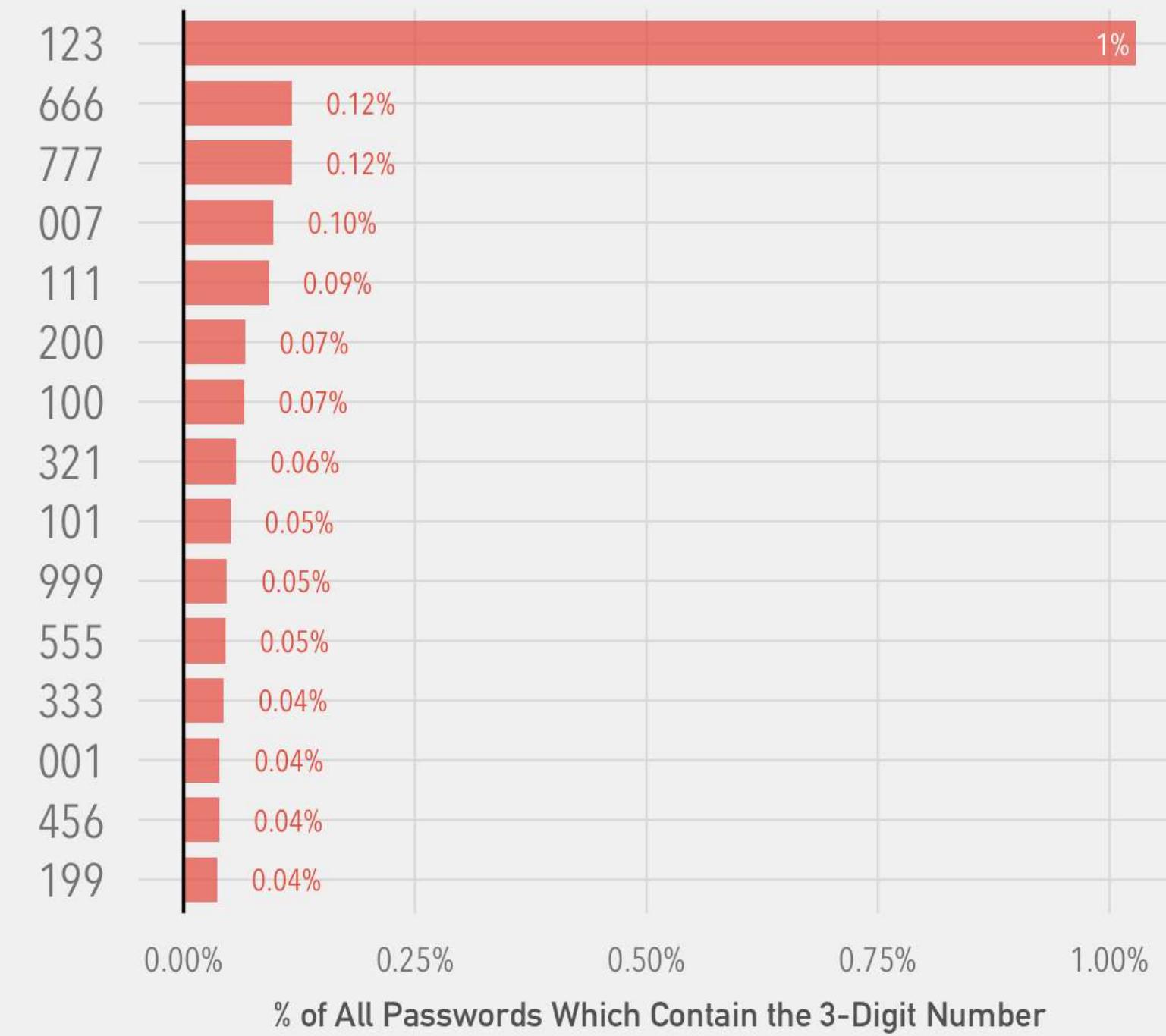
Ricerca euristica

“Aggiungo un **numero** alla fine così diventa sicura...”

Most Frequently-Used 2-Digit Numbers in 10 Million Passwords



Most Frequently-Used 3-Digit Numbers in 10 Million Passwords



MAX WOOLF — MINIMAXIR.COM

SOURCE: MARK BURNETT @ XATO.NET



MAX WOOLF — MINIMAXIR.COM

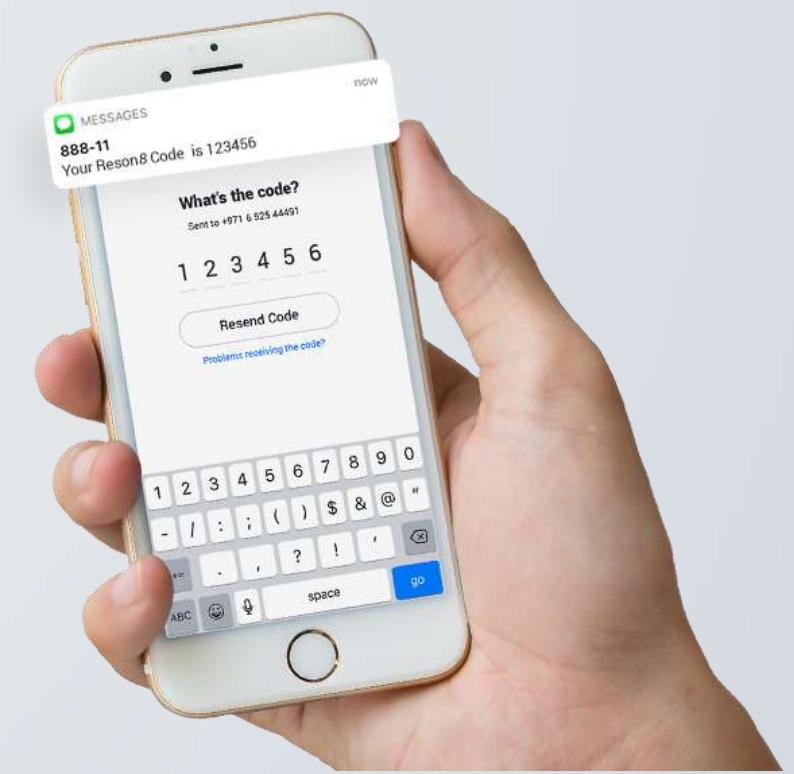
SOURCE: MARK BURNETT @ XATO.NET

One-time-password

È una password valida solo per una singola sessione di accesso o transazione.

Risolve il problema degli attacchi con replica (= qualcuno indovina la nostra password e la utilizza per eseguire l'accesso)

Non può essere memorizzata (sarebbe inutile, visto che cambia continuamente) e ha quindi bisogno di una tecnologia supplementare (**token**) per poter essere utilizzata.



One-time-password

Come vengono generate?

Tre idee principali:

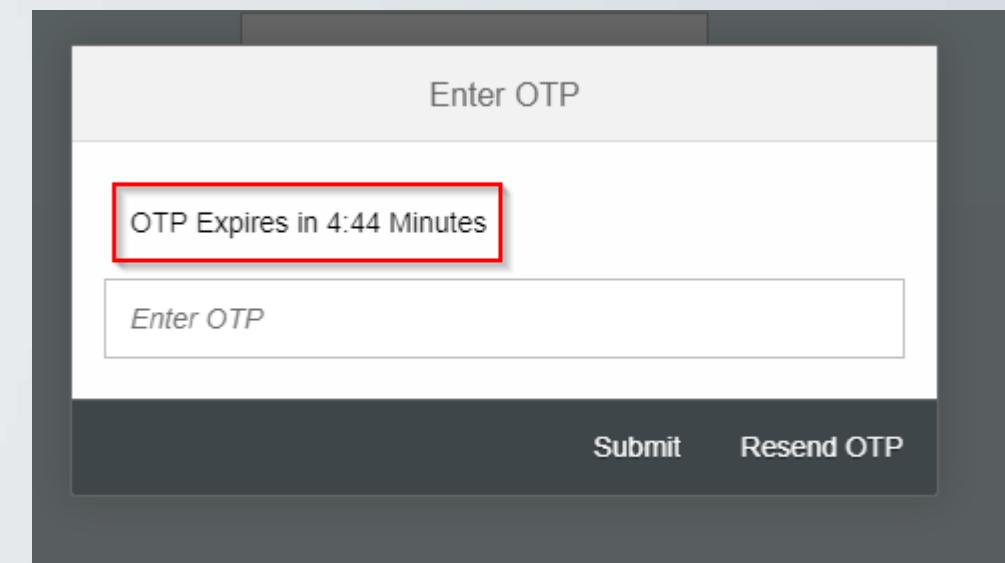
1. Sincronizzazione temporale
2. Algoritmi matematici che generano una nuova password in base alla password precedente, secondo una sequenza predefinita
3. Confronto tra input utente e valore atteso

One-time-password

Come vengono generate?

Sincronizzazione temporale

Il token ha un orologio interno, sincronizzato con l'orologio del server. La password generata è valida solo all'interno di quel determinato intervallo temporale.



One-time-password

Come vengono generate?

Algoritmi matematici che generano una nuova password in base alla password precedente, secondo una sequenza predefinita

- viene scelto un seme (valore iniziale) s
- viene applicata ripetutamente (es 1000 volte) una funzione hash $h(s)$
- viene memorizzato $h_{1000}(s)$ sul server

One-time-password

Come vengono generate?

Algoritmi matematici che generano una nuova password in base alla password precedente, secondo una sequenza predefinita

- Al primo accesso, l'utente utilizza $h_{999}(s)$
- Il sistema riceve $h_{999}(s)$ e calcola $h(h_{999}(s)) = h_{1000}(s)$
- Il sistema confronta quanto calcolato con quanto salvato. Viene cancellato $h_{1000}(s)$ e memorizzato $h_{999}(s)$

One-time-password

Come vengono generate?

Algoritmi matematici che generano una nuova password in base alla password precedente, secondo una sequenza predefinita

- Al secondo accesso, l'utente utilizza $h_{998}(s)$
- La catena di step si ripete, e può essere ripetuta altre 997 volte.
Quando arriviamo a $h_1(s)$ avremo bisogno di generare un nuovo seme **s**

One-time-password

Eventuali problemi/attacchi

- Social engineering
- Man-in-the-middle
- Prevedibilità se la generazione non è veramente casuale
- ecc ...

Per questi motivi, spesso gli OTP vengono utilizzati in combinazione con una password memorizzata dall'utente.



Password security

Bene, ma non benissimo



52%

35%

13%

riusa la stessa password per molti (ma non tutti) gli account

usa password diverse per tutti gli account

usa la stessa password per tutti gli account

Bene, ma non benissimo

13 volte

è, in media, il riuso delle password per un impiegato.

50%

delle password viene usata sia per account lavorativi e personali.

Bene, ma non benissimo

80%

dei *data breach* del 2019 sono dovuti a password riutilizzate.



FIRST REACTION: SHOCK!

Data breach

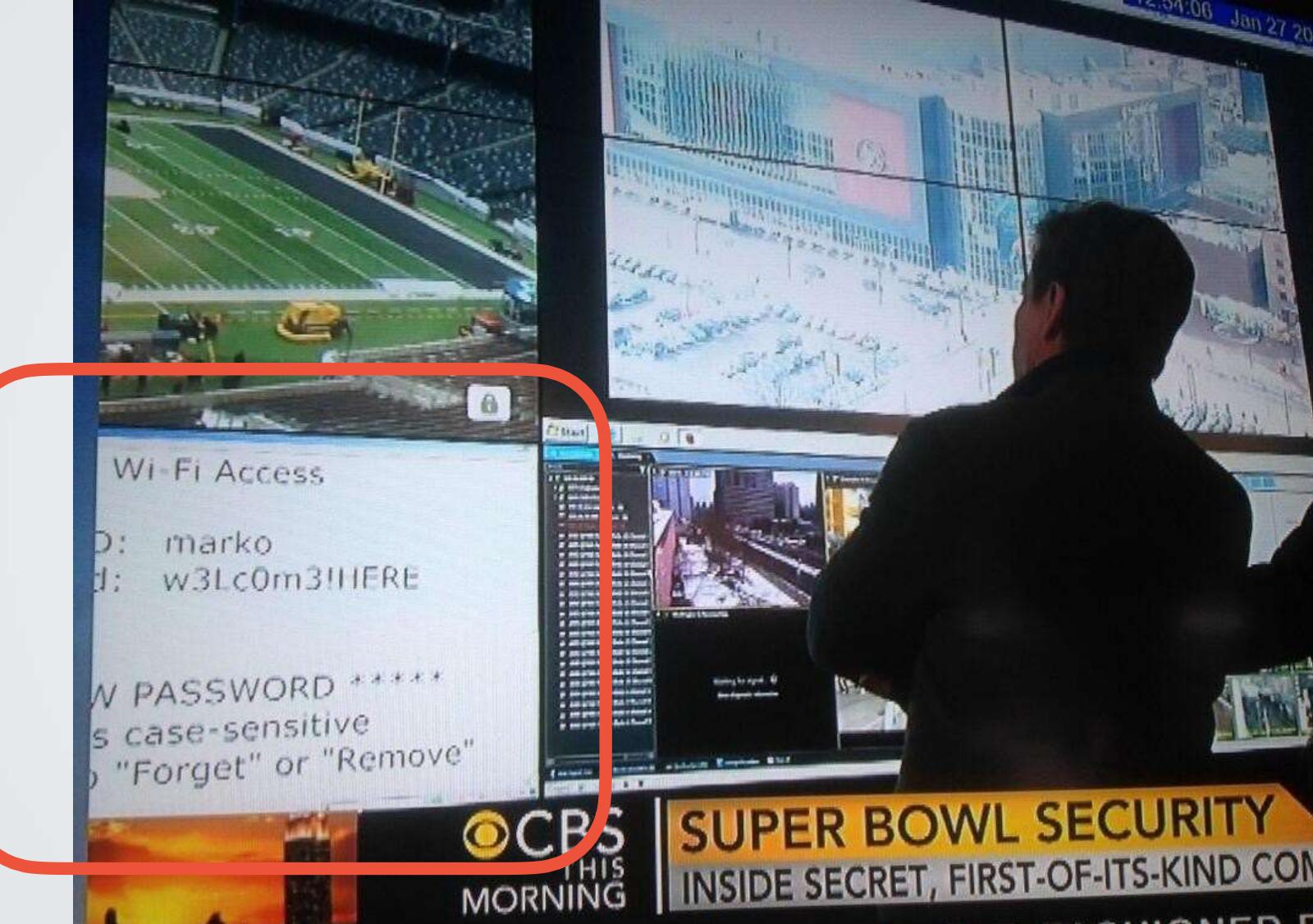
Yahoo, 2013	3 miliardi di account	
Sony, 2014	100 TB di dati	
Anthem, 2015	80 milioni di account	
Friend Finder, 2016	412 milioni di account	
Equifax, 2017	163 milioni di account	
Marriot, 2018	500 milioni di account	
Facebook, 2019	540 milioni di account	
Wattpad, 2020	270 milioni di account	

Epic fail

TVMonde, 2015



CBS, 2014



PayPal UK  
@PayPalUK Richmond, UK
The official twitter account for the fail team at PayPal UK
<http://www.paypalsucks.com>

+ Follow

Tweets Favorites Following Followers Lists

PayPalUK PayPal UK
Shop safely without paypal. [paypalsucks.com](http://www.paypalsucks.com) 9 minutes ago

PayPalUK PayPal UK
All your paypal accounts are now frozen while we clean up this mess.. 30 minutes ago

tweeterkatie Katie McGuire  by PayPalUK
Safe to say @PayPalUK has been hacked. I didn't realise they were so dirty! Should probably cancel my account! 43 minutes ago

 **@foxnewspolitics**
foxnewspolitics

@BarackObama has just passed. The President is dead. A sad 4th of July, indeed. President Barack Obama is dead

5 hours ago via web

 **John Podesta** 
@johnpodesta

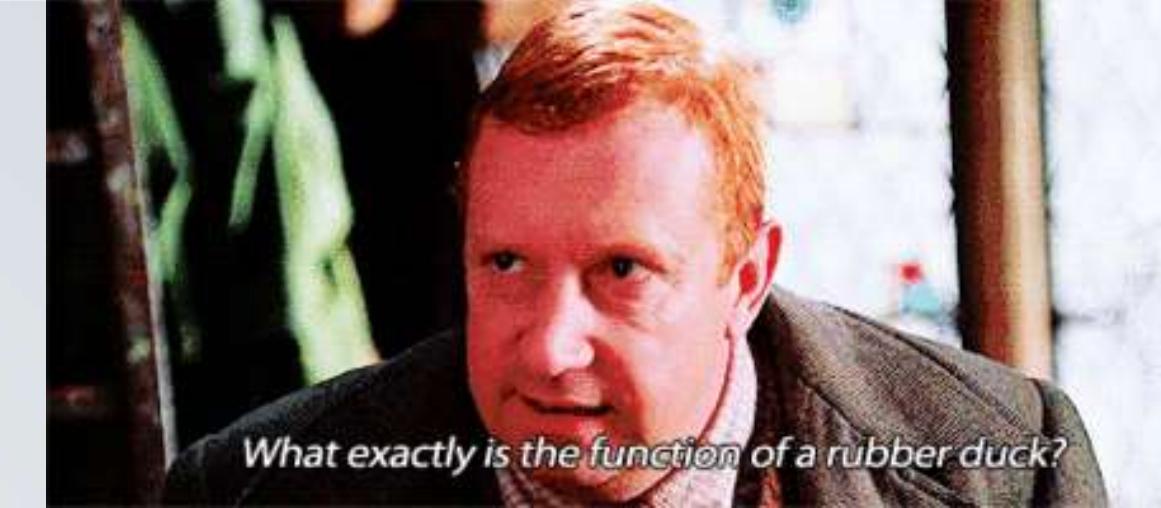
I've switched teams. Vote Trump 2016. Hi pol

7:55 PM - 12 Oct 16

54 RETWEETS 55 LIKES



Qual è, con esattezza...?



60%

Sa definire *phishing* correttamente.



55%

Sa definire correttamente *password manager*.



55%

Sa definire correttamente *autenticazione multifattore*.



32%

Sa definire correttamente tutti i termini.

Phishing



L'attaccante cerca di ingannare la vittima convincendola a fornire informazioni personali, dati finanziari o codici di accesso, fingendosi un ente affidabile in una comunicazione digitale.

Clone phishing

i link di una mail legittima vengono modificati per tentare di ingannare il ricevente.

Spear phishing

mirato ad un individuo o compagnia, alta probabilità di successo.

Whaling

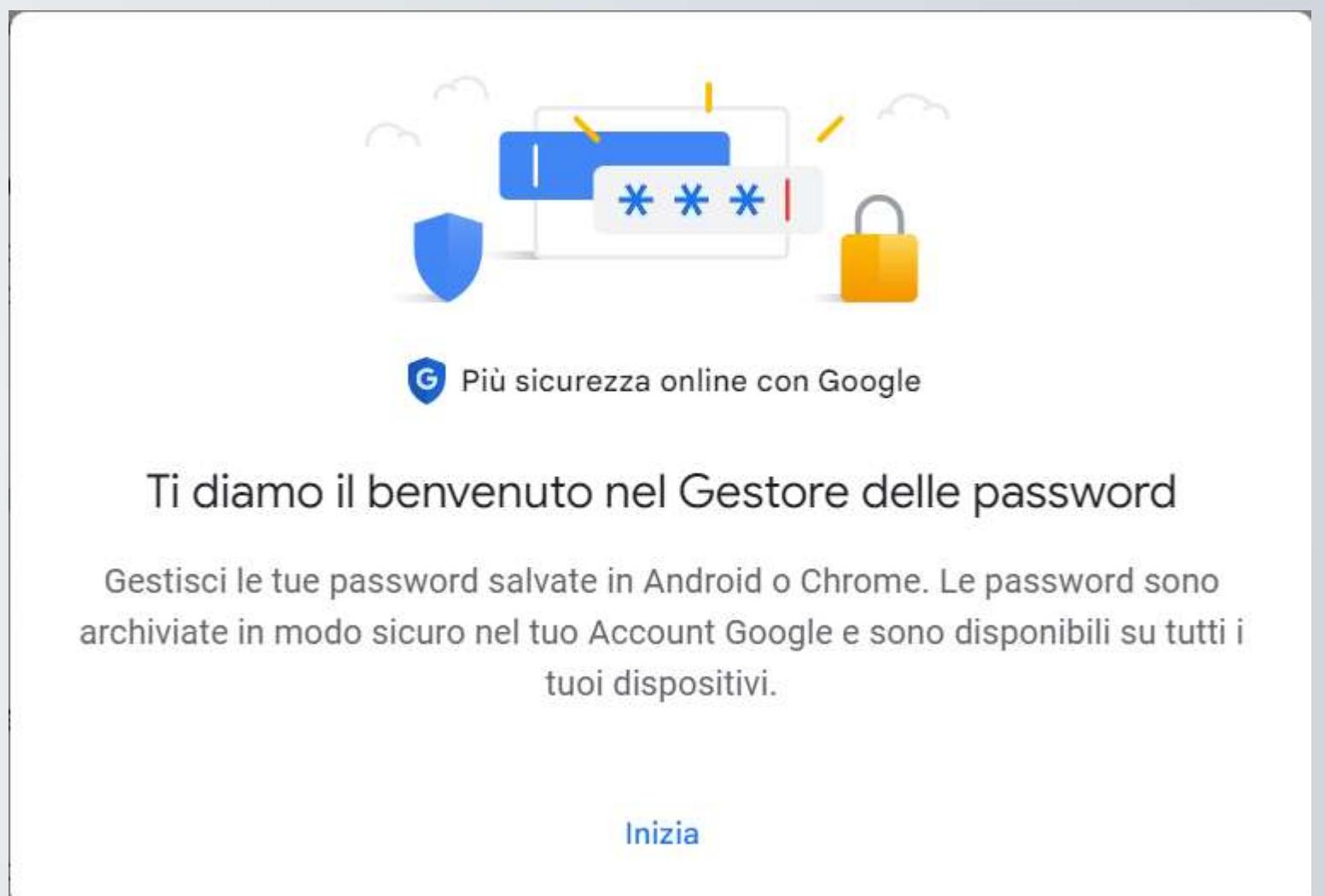
spear phishing indirizzato ad alti profili aziendali, fanno leva su azioni legali, problemi amministrativi o finanziari.

Password manager

Per **password manager** si intende un programma che consente di salvare e gestire in modo sicuro le password.

Spesso, i password manager sono anche in grado di generare password forti e sono programmati per salvare in modo sicuro anche altri tipi di dati sensibili, come ad esempio i dati di una carta di credito.

Per accedere a tutte le informazioni del password manager, l'utente deve ricordarsi solo una password, quella del password manager stesso.



Password manager

Built-in



Low level

Installazione separata

bitwarden

1Password

Mid level

High level



Password manager

Quali sono i vantaggi?

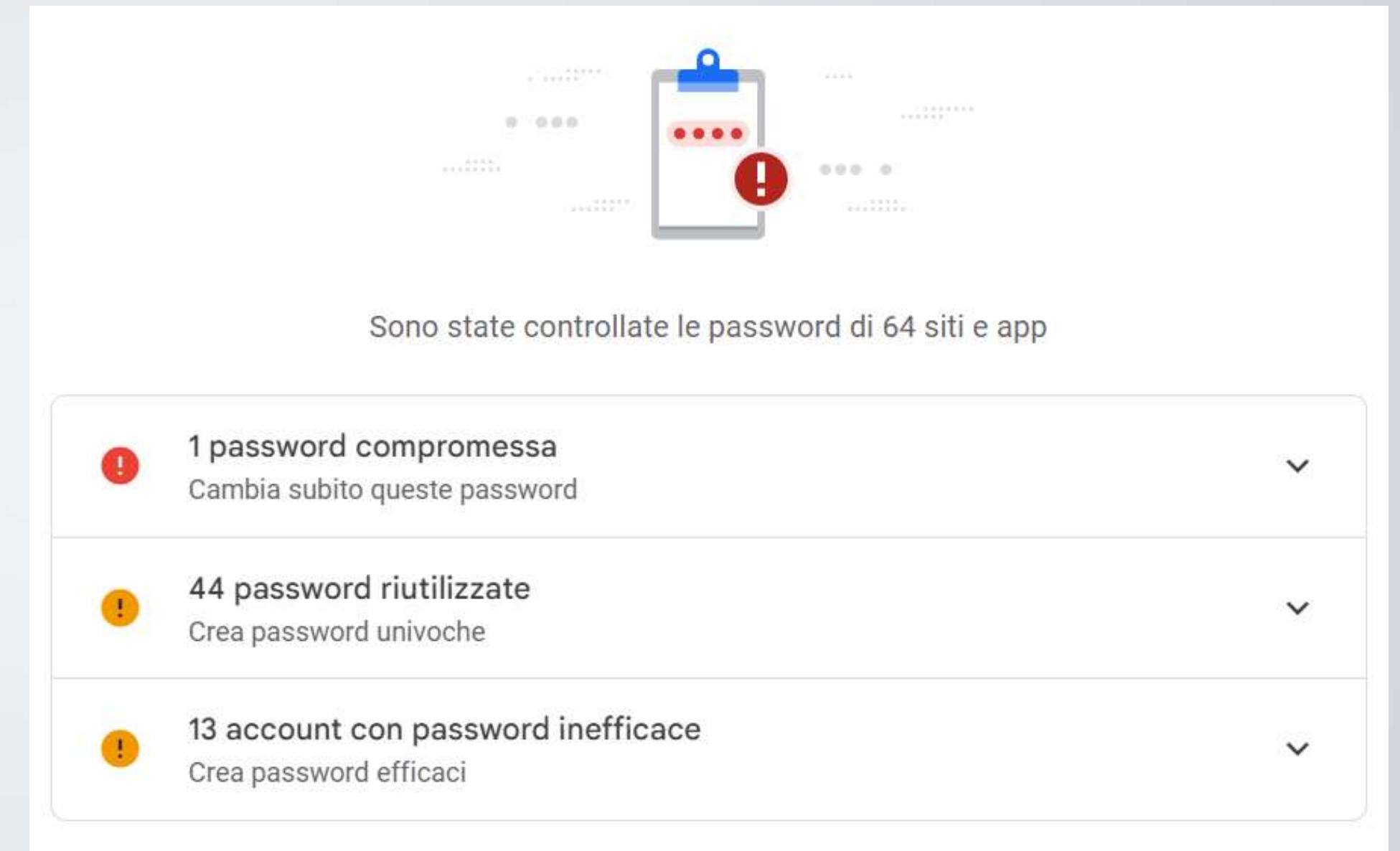
- Non c'è più bisogno di ricordarsi ogni singola password (**che dovrebbe essere diversa per ogni sito e complessa abbastanza da non poter essere indovinata**)



- Autocomplete: viene fatto solo per i siti riconosciuti e salvati come sicuri (cioè quelli in cui effettivamente ci siamo registrati). Questo ci mette al riparo eventuali tentativi di phishing: se l'autocomplete non funziona, forse è il caso di controllare il sito

Password manager

- Le password sono monitorate e possono essere mostrati degli avvisi in caso di data leak



Autenticazione multifattore



(password)

Qualcosa che
conosco



(OTP)

Qualcosa che
possiedo



(biometria)

Qualcosa che
sono