Sicurezza dei sistemi e delle reti[1]

Mattia Monga

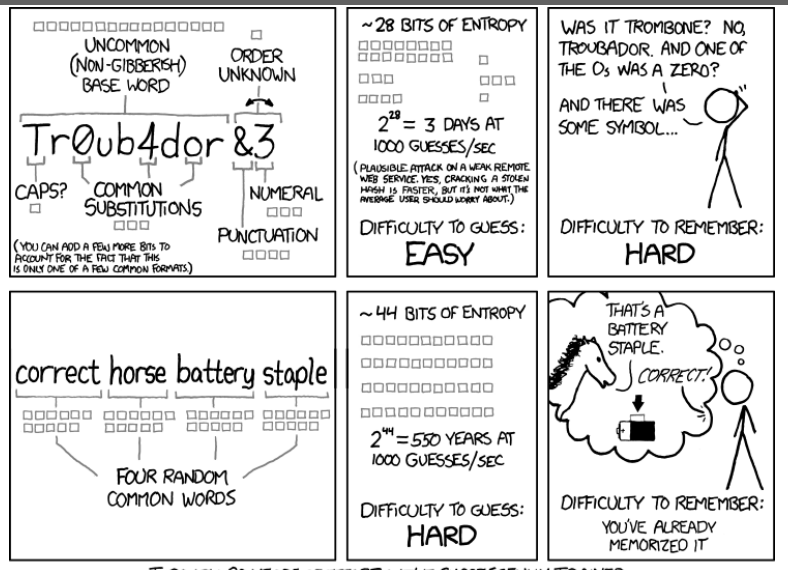
Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2014/15

Lezione XV: Autenticazione

Password guessing

- Una password può essere scelta in maniera prevedibile (anziché **del tutto casuale**) nell'insieme possibile.
- *Online guessing*: l'attaccante prova tutte le password possibili (**brute force**); si limitano i tentativi e/o si rallenta il feedback



http://xkcd.com/936/

Da `/usr/share/dict/italian`

45	3
182	4
1002	5
3106	6
5694	7
10748	8
15374	9
19126	10
20532	11
15996	12
11225	13
6931	14
3535	15
2020	16
733	17
339	18
160	19
72	20
40	21
10	22
2	23
5	24
1	25

$$26^8 = 2,088 \cdot 10^{11}$$
$$4000^4 = 2,560 \cdot 10^{14}$$

Offline guessing

Offline guessing: l'attaccante accede all'elenco dei segreti (generalmente crittati con hash) e prova elenchi di parole (**dictionary attack**); si salano gli hash per rendere impraticabile la realizzazione di *rainbow table*.

Utente	salt	stored password
Alice	42	$\mathrm{hash}(42 \vert \mathrm{password}_{\mathrm{Alice}})$

- Possibilità di **intercettazione**
- Utilizzo in occasioni differenti
- **distribuzione iniziale delle credenziali** (si fanno scadere al primo accesso)

Riassumendo

Per i servizi critici è fondamentale curare il **controllo degli accessi**

- Autenticazione e autorizzazione sono logicamente distinte
- Le credenziali sono un elemento critico per la sicurezza di tutto il sistema di controllo.

Credenziali generalizzate

Alice può provare la sua identità mostrando

- qualcosa che **sa** (password tradizionale)
- qualcosa che **ha** (authentication token)
- qualcosa che **è** (biometria)

È naturalmente possibile (e spesso desiderabile) avere autenticazioni **a piú fattori**.

Client inaffidabili

La diffusione del malware ha reso spesso inaffidabili i client

Chi garantisce che la schermata di login non sia un *cavallo di Troia* capace di memorizzare/rubare le credenziali?

I tastierini

La protezione del "tastierino" che **cambia ad ogni login** è puramente apparente. Un esempio di falsa sicurezza, che tra l'altro impedisce all'utente di utilizzare meccanismi automatici di memorizzazione delle password

Anti-malware

Contro client alterati è difficile proteggersi, ma protezioni piú efficaci sono:

- In ogni sessione viene comunicata solo parte della password
- Two-factor authentication (2FA)
- One-time password (OTP)

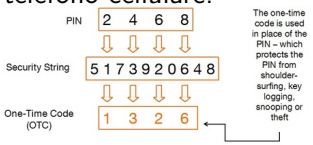
Altri attacchi

Nessuna di queste misure protegge da:

- Man in the Browser il client alterato non si limita ad intercettare le credenziali, ma è in grado di manipolare i dati della transazione
- Session hijacking l'attaccante è in grado di manipolare una sessione già in corso

2FA

Due (o piú) meccanismi di autenticazione: una password e il possesso di un oggetto fisico: p.es. un security token o un telefono cellulare.



È importante che i due fattori siano effettivamente indipendenti: web browsing con uno smart-phone e sms?

Security token

I piú diffusi si basano su synchronous dynamic password: la password cambia ad intervalli regolari, per esempio ogni minuto. La sincronia è un fattore critico

One Time Password

In generale si tratta di password **utilizzate una volta soltanto** e pertanto non riutilizzabili da un eventuale intercettore. L'effettivo possesso di un security token è generalmente verificato tramite una one time password generata dal token stesso o richiesta *out of band*.

Riassumendo

Le autenticazioni possono combinare piú fattori

- Segreti
- Oggetti fisici
- Parametri biometrici

Lo schema di Lamport

Leslie Lamport nel 1981 ha proposto uno schema per autenticazione tramite OTP che non prevede la necessità di sincronizzazione temporale.
Si basa sull'esistenza di una **funzione di hash** H sicura (non invertibile).

Lo schema di Lamport

1. Alice e Bob concordano un segreto W
2. Bob conserva $H(\dots H(H(W))\dots) = H^n(W)$ e n
3. Autenticazione
   1. Alice comunica la propria *username*
   2. Bob risponde con il numero n
   3. Alice comunica $x = H^{n-1}(W)$
   4. Bob verifica che $H(x) = H^n(W)$ e decrementa n

Lo schema funziona n volte, poi bisogna cambiare W.

S/KEY

Lo schema di Lamport è stato implementato da Neil Haller, Phil Karn e John Walden in S/KEY.
Usa numeri a 64 bit + 2 bit di parità.
Stringhe casuali di 8 caratteri sono difficili da utilizzare per un utente umano, è prevista una mappatura su 2048 parole da 1 a 4 caratteri: i 66 bit diventano una sequenza di 6 parole
(`TAG SLOW NOV MIN WOOL KENO` ↔ `0x3F3BF4B4145FD74B`).


```
{
"AGO",  "AID",  "AIM",  "AIR",  "ALL",  "ALP",  "AM",   "AMY",
"AN",   "ANA",  "AND",  "ANN",  "ANT",  "ANY",  "APE",  "APS",
"APT",  "ARC",  "ARE",  "ARK",  "ARM",  "ART",  "AS",   "ASH",
"ASK",  "AT",   "ATE",  "AUG",  "AUK",  "AVE",  "AWE",  "AWK",
"AWL",  "AWN",  "AX",   "AYE",  "BAD",  "BAG",  "BAH",  "BAM",
"BAN",  "BAR",  "BAT",  "BAY",  "BE",   "BED",  "BEE",  "BEG",
"BEN",  "BET",  "BEY",  "BIB",  "BID",  "BIG",  "BIN",  "BIT",
"BOB",  "BOG",  "BON",  "BOO",  "BOP",  "BOW",  "BOY",  "BUB",
"BUD",  "BUG",  "BUM",  "BUN",  "BUS",  "BUT",  "BUY",  "BY",
"BYE",  "CAB",  "CAL",  "CAM",  "CAN",  "CAP",  "CAR",  "CAT",
"CAW",  "COD",  "COG",  "COL",  "CON",  "COO",  "COP",  "COT",
"COW",  "COY",  "CRY",  "CUB",  "CUE",  "CUP",  "CUR",  "CUT",
"DAB",  "DAD",  "DAM",  "DAN",  "DAR",  "DAY",  "DEE",  "DEL",
"DEN",  "DES",  "DEW",  "DID",  "DIE",  "DIG",  "DIN",  "DIP",
"DO",   "DOE",  "DOG",  "DON",  "DOT",  "DOW",  "DRY",  "DUB",
"DUD",  "DUE",  "DUG",  "DUN",  "EAR",  "EAT",  "ED",   "EEL",
"EGG",  "EGO",  "ELI",  "ELK",  "ELM",  "ELY",  "EM",   "END",
"EST",  "ETC",  "EVA",  "EVE",  "EWE",  "EYE",  "FAD",  "FAN",
"FAR",  "FAT",  "FAY",  "FED",  "FEE",  "FEW",  "FIB",  "FIG",
"FIN",  "FIR",  "FIT",  "FLO",  "FLY",  "FOE",  "FOG",  "FOR",
"FRY",  "FUM",  "FUN",  "FUR",  "GAB",  "GAD",  "GAG",  "GAL",
"GAM",  "GAP",  "GAS",  "GAY",  "GEE",  "GEL",  "GEM",  "GET",
"GIG",  "GIL",  "GIN",  "GO",   "GOT",  "GUM",  "GUN",  "GUS",
"GUT",  "GUY",  "GYM",  "GYP",  "HA",   "HAD",  "HAL",  "HAM",
"HAN",  "HAP",  "HAS",  "HAT",  "HAW",  "HAY",  "HE",   "HEM",
"HEN",  "HER",  "HEW",  "HEY",  "HI",   "HID",  "HIM",  "HIP",
"HIS",  "HIT",  "HO",   "HOB",  "HOC",  "HOE",  "HOG",  "HOP",
"HOT",  "HOW",  "HUB",  "HUE",  "HUG",  "HUH",  "HUM",  "HUT",
"I",    "ICY",  "IDA",  "IF",   "IKE",  "ILL",  "INK",  "INN",
"IO",   "ION",  "IQ",   "IRA",  "IRE",  "IRK",  "IS",   "IT",
"ITS",  "IVY",  "JAB",  "JAG",  "JAM",  "JAN",  "JAR",  "JAW",
"JAY",  "JET",  "JIG",  "JIM",  "JO",   "JOB",  "JOE",  "JOG",
"JOT",  "JOY",  "JUG",  "JUT",  "KAY",  "KEG",  "KEN",  "KEY",
"KID",  "KIM",  "KIN",  "KIT",  "LA",   "LAB",  "LAC",  "LAD",
"LAG",  "LAM",  "LAP",  "LAW",  "LAY",  "LEA",  "LED",  "LEE",
"LEG",  "LEN",  "LEO",  "LET",  "LEW",  "LID",  "LIE",  "LIN",
"LIP",  "LIT",  "LO",   "LOB",  "LOG",  "LOP",  "LOS",  "LOT",
"LOU",  "LOW",  "LOY",  "LUG",  "LYE",  "MA",   "MAC",  "MAD",
"MAE",  "MAN",  "MAO",  "MAP",  "MAT",  "MAW",  "MAY",  "ME",
```

Vulnerabilità

Non protegge da Man in the Browser o Session Hijacking. Piú grave è l'attacco **con n piccolo**

1. Bob conserva il numero n
2. Mallory impersona Bob e manda ad Alice un $n' < n$
3. Alice risponde con $H^{n'-1}(W)$
4. Mallory potrà usare $H^{n'-1}(W)$ per sostituirsi ad Alice quando Bob arriverà a conservare n'

Mitigato rendendo Alice edotta su n corrente, in modo che possa insospettirsi per eventuali $n' \ll n$

Paper Lamport

Funziona bene con modalità "carta e penna".

- Alice conserva una lista cartacea di password $(H^n(W), H^{n-1}(W), \dots)$
- Una volta usata la prima della lista la cancella

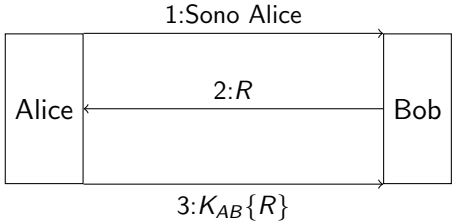
Questo schema non è suscettibile dell'attacco di n piccolo, ma la lista è naturalmente un punto critico.

Riassumendo

Lo schema di Lamport

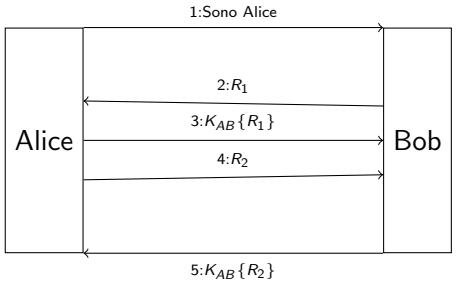
- Un meccanismo algoritmico per OTP
- Si basa sull'esistenza di funzioni di hash non invertibili
- Vulnerabile all'attacco 'n piccolo'

Challenge/Response



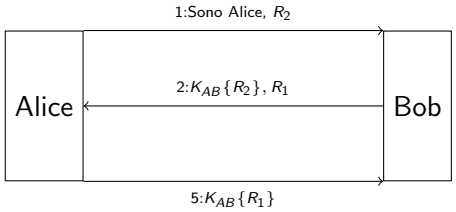
- L'autenticazione non è reciproca: qualcuno potrebbe sostituirsi a Bob.
- Offline-guessing di K_{AB} è possibile intercettando R e $K_{AB}\{R\}$

Challenge/Response con mutua autenticazione

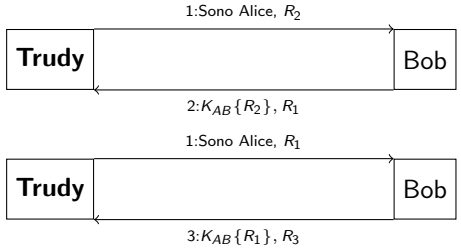


Sono necessari ben 5 scambi: si può rendere piú efficiente?

Challenge/Response con mutua autenticazione



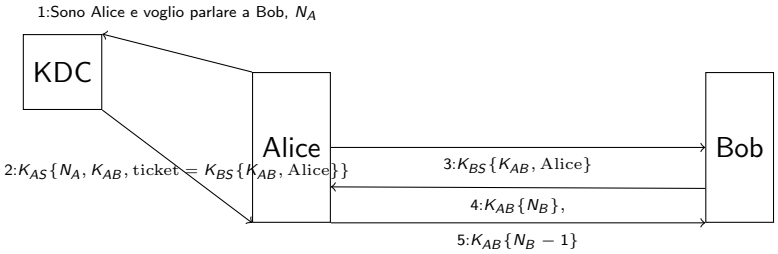
Purtroppo si presta ad un **reflection attack**



Inoltre si presta all'offline guessing (anche senza intercettazione!).

Needham-Schroeder

Per semplificare la gestione dei segreti condivisi, possono essere introdotti dei Key Distribution Center (KDC).
Needham-Schroeder [1978]



Needham-Schroeder è vulnerabile

Qualora K_{AB} sia compromesso (p.es. accedendo alla macchina di Alice) è possibile un replay attack del *ticket*, quindi occorre complicarlo ulteriormente introducendo dei timestamp, in modo che i ticket non possano essere riutilizzati. Un'evoluzione (molto diffusa) che include i timestamp è Kerberos.

Riassumendo

I protocolli crittografici possono essere molto utili

- Permettono mutua autenticazione
- Occorre fare molta attenzione alle possibilità di replay
- La gestione delle chiavi è architetturalmente la faccenda piú complicata

Il single sign-on (SSO)

L'idea di avere credenziali che permettono l'accesso a sistemi diversi è appetibile per una serie di ragioni

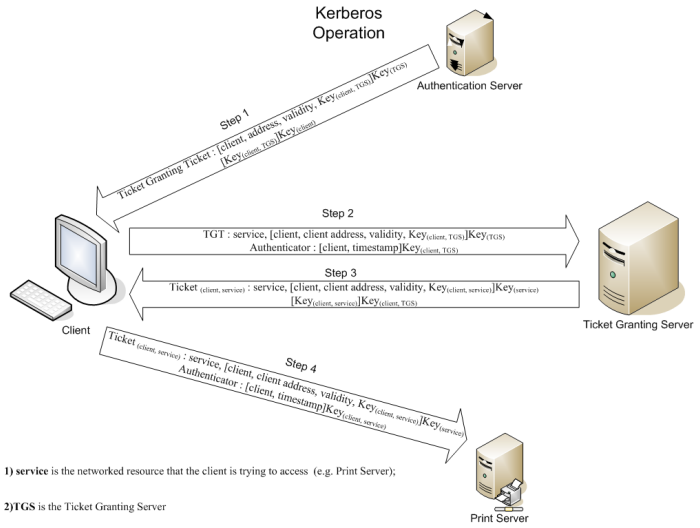
- Riduce il problema di trovare un buon segreto (sufficientemente casuale, ecc.)
- Riduce l'overhead totale di gestione degli accessi
- Permette la gestione centralizzata degli accessi, piú semplice da manutenere

(Aumenta la criticità delle credenziali, però)

Architettura di base

La maggior parte dei protocolli sono disegnati sull'impronta di Kerberos, che è una variazione con timestamp del protocollo Needham-Schroeder.
Un Ticket-Granting Server (TGS) fornisce ticket d'accesso a scadenza.

Kerberos



[Figura: `wikipedia.org`]

Il SSO sul web

Il SSO sembra particolarmente attraente in situazioni come i servizi web a bassa criticità:

- Decine di password da ricordare
- Utenti poco sensibili al problema

Ma soluzioni tipo Kerberos sono difficili da adottare, perché occorre che servizi indipendenti concordino sulla KDC.

OpenID

L'idea è che esistano siti che facciano da OpenID provider (OP) e gli OpenID Consumer (OC) accettano come credenziali quelle che gli utenti ottengono dagli OP (cui sostanzialmente gli OC delegano l'autenticazione)

Lo schema

1. Alice si connette al sito `meraviglie.org` (OC)
2. Per accedere comunica che ha un account su `faccialibro.com/openid/alice`
3. Alice accede (con credenziali tradizionali) a `faccialibro.com` (OP) e produce un **certificato d'accesso**
4. `meraviglie.org` accetta il certificato d'accesso come credenziale d'autenticazione

Problemi di OpenID

Lettura obbligatoria:
`http://www.untrusted.ca/cache/openid.html`
I principali rischi

- Facilità di allestire inganni di tipo **phishing**
- Privacy

Alternative migliori

In realtà sembra meglio avere **credenziali multiple** e gestirle con modalità come:

- PasswordSafe (`http://passwordsafe.sf.net`): un db criptato
- SuperGenPass (`http://supergenpass.com/`): un'unica password viene giustapposta ad un identificatore del sito e la vera password è ottenuta con una funzione hash. L'idea è ottima, la realizzazione ha diverse vulnerabilità: meglio usare alternative piú *crypto-savvy*, p. e.s. (`http://hpass.chmd.fr/`).

Riassumendo

Il SSO

- tipo kerberos è inadatto alla gestione di servizi web indipendenti
- le soluzioni del tipo OpenID presentano diversi problemi