Alma Mater Studiorum · Università di Bologna

Campus di Cesena - Scuola Di Scienze

Corso di Laurea in Scienze e Tecnologie Informatiche

Sicurezza dei sistemi di autenticazione basati su credenziali, raccolta ed analisi di un vasto insieme di password

Relatore: **Presentata da:**

GABRIELE D'ANGELO **MICHELE PROVERBIO**

III Sessione

Anno Accademico 2013-2014

"Il Mercury Computer chiuso in una cassa di cristallo, coi suoi cavi sottili, protetto con ogni precauzione, respirava e viveva."

Stranger in Paradise - Isaac Asimov

Prefazione

Nell'era dell'informazione in cui viviamo, ogni persona viene sdoppiata in due entità: oltre alla parte reale e materiale vi è un alterego fatto di dati ed informazioni che vive nei database e nei server che compongono il web. Il collegamento tra le due parti è dato dagli account e dalle chiavi di autenticazione, che si presumono essere segrete. Al giorno d'oggi ogni persona possiede almeno un account, sia esso una web mail, un profilo su un social network o l'account di online banking. Quasi la totalità di questi sistemi effettua l'autenticazione dell'utente tramite l'immissione di una password, e la sicurezza di quest'ultima è essenziale per la protezione delle informazioni personali, dei dati bancari e della propria identità.
Col passare del tempo le informazioni personali che diamo in custodia al web crescono sempre più velocemente e diventano sempre più personali e delicate, ma purtroppo l'importanza che viene data alla sicurezza informatica non cresce di pari passo sia da parte degli utenti, sia da parte dei fornitori dei servizi web.
Questa tesi ha come scopo quello di portare il lettore ad avere più coscienza rispetto alle tematiche della sicurezza informatica e di esporre in modo chiaro e semplice alcune delle problematiche che riguardano la sicurezza dei sistemi di autenticazione.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione alla sicurezza dei sistemi di autenticazione

Questo capitolo vuole introdurre il lettore all'argomento della sicurezza dei sistemi di autenticazione con password. Per prima cosa verrà illustrato il funzionamento del protocollo di autenticazione: il login al servizio, la comunicazione delle credenziali di accesso al server, e verrà spiegato in quale modo le credenziali vengono salvate e gestite dal server. Successivamente verranno illustrate le principali tecniche di attacco e si cercherà inoltre di presentare alcune delle tecniche da utilizzare per difendersi da questi attacchi, come applicarle e come spesso vengono utilizzate in modo errato.

1.1 Anatomia dei sistemi di autenticazione

Il protocollo di autenticazione tra un client e un server avviene attraverso lo scambio di una chiave di autenticazione; la chiave è composta da un nome utente, univoco per ogni account, e una password segreta. La chiave di autenticazione viene creata attraverso il processo di registrazione al servizio, solitamente tramite la compilazione di una form in un ambiente web, durante il quale l'utente immette un nome utente e una nuova password; durante questo processo il server si deve occupare di controllare che non esistano altri

account con lo stesso nome utente, altrimenti non sarà possibile stabilire una relazione tra l'utente e l'account appena creato. Attraverso il processo di registrazione, infatti, l'utente definisce in modo univoco la relazione tra la sua persona e l'account creato presso il servizio. L'univocità di questa relazione è essenziale ai fini della comunicazione tra client e server. Una volta che più di una persona possiede la chiave d'autenticazione il vincolo di univocità viene rotto e si presentano ovvi problemi di sicurezza.

1.1.1 Login e comunicazione con il server

Nel caso dei servizi web viene esposta agli utenti una form di login, solitamente in una pagina apposita, nella quale l'utente deve inserire le proprie credenziali per ottenere i privilegi di accesso al servizio.
La form è solitamente formata da due o più campi di input *HyperText Markup Language* (HTML) [4], uno per l'username e uno per la password; è inoltre possibile trovare altri campi per l'inserimento, ad esempio, di codici captcha [5]. Una volta premuto il tasto di invio della form di login, le credenziali vengono impacchettate in un messaggio *Hypertext Transfer Protocol* (HTTP) e inviate al server per la verifica. Una prima distinzione da fare è la modalità con cui queste informazioni vengono inserite nel messaggio. Le modalità sono due, e vengono specificate nel tag *form* della pagina html; la modalità GET [2] inserisce il contenuto dei campi input nell'*Uniform Resource Locator* (URL) a fianco dell'indirizzo del sito web precedute dal simbolo ?, ad esempio per una semplice form di login con modalità di sottomissione GET, questa è la richiesta che viene sottoposta al server *http://site.org/login.php?username=alice&password=qwerty*. La seconda modalità è la POST [1], la quale inserisce i dati nel body del messaggio http, in questo modo vengono nascosti a prima vista, ed è comunque da preferire al metodo GET.
La maggior parte dei servizi web ora incapsula tutto il traffico, compreso lo scambio di credenziali, all'interno di *Hypertext Transfer Protocol Secure* (HTTPS) al posto del classico HTTP. HTTPS aggiunge al protocollo stan-

dard un servizio di crittografia *Transport Layer Security* (TLS) che permette una comunicazione sicura tra client e server. I messaggi vengono criptati con un sistema a chiave simmetrica, la quale viene creata attraverso una prima fase di handshaking con chiavi asimmetriche, più lenta ma molto più sicura [7].
L'uso di https è essenziale per ottenere un livello di sicurezza di base, che protegga la comunicazione da osservazioni esterne.
In realtà la password viene mandata al server sotto forma di hash in quanto, come vedremo nella prossima sezione, le password vengono mantenute sul server sotto forma di hash. In generale, è buona regola di sicurezza che la password non esca mai in chiaro dal client.

1.1.2 Server e conservazione delle credenziali di accesso

Le richieste http e https inviate dai client vengono ricevute e gestite da un web server. Quando il server riceve il messaggio, lo scompone attraverso il parsing del contenuto e chiama lo script per la generazione della risposta passandogli gli oggetti estratti dal messaggio. Nel caso di una richiesta di login, lo script troverà le credenziali immesse dall'utente sotto forma di stringa in variabili fornite dal web server. Lo script può quindi utilizzare questi dati per controllare la validità della password attraverso la sottomissione di una query al *Database Management System* (DBMS), il quale risponderà con esito positivo nel caso in cui le credenziali fossero corrette e negativo altrimenti. Il database sottostante è solitamente implementato con una tabella che possiede i campi *username*, *mail*, *hash_password* e dati descrittivi come la data di iscrizione, il sesso o la nazionalità. È importante notare che le password degli utenti vengono salvate sotto forma di hash [8]; ovvero vengono prima trasformate da una funzione di hashing in un'altra stringa alfanumerica. Le funzioni di hashing sono funzioni non iniettive, pertanto non esiste una funzione inversa per poter ricavare la password dal suo hash. Come vedremo nel capitolo successivo esistono tecniche per poter ricavare la password generan-

te da un hash; perciò, è necessario usare delle accortezze nella scelta della password.
Le funzioni hash più usate per questo scopo sono md5 e sha [9][10]; la prima, pur essendo di gran lunga la più comune, espone alcune vulnerabilità [11] che lo rendono un algoritmo poco sicuro per la protezione delle password. Per questo motivo è caldamente consigliato l'utilizzo di algoritmi più sicuri come sha-256 o sha-512. Il problema purtroppo è scarsamente riconosciuto, e per questione di retrocompatibilità la maggior parte dei vecchi sistemi non è stata aggiornata, esponendo gli utenti a gravi problemi di sicurezza.

1.2 Sicurezza dei sistemi di autenticazione. Attacco e difesa

I sistemi di autenticazione descritti nella sezione precedente sono spesso bersagli di attacchi informatici atti a rubare informazioni sensibili per usi impropri, come furti di identità e raccolta di indirizzi mail per inoltrare spam o tentativi di phishing [12].
È interessante distinguere gli attacchi informatici applicati a sistemi di autenticazioni in due macro categorie: attacchi mirati, atti a indovinare la password di una singola persona, e attacchi su larga scala, che si limitano a trovare quante più chiavi di autenticazione possibili per un vasto pool di account. Dagli attacchi mirati è ovviamente più difficile difendersi, in quanto l'attaccante ha la possibilità di effettuare information gathering sulla vittima; ha inoltre la possibilità di usare una quantità molto superiore di risorse computazionali rispetto ad attacchi su larga scala [3]. In ogni caso la contromisura migliore resta la scelta di una buona password. In questa sezione andremo a illustrare alcune delle tecniche di attacco che possono mettere a rischio la sicurezza dei moderni sistemi di autenticazione. Per ognuna di queste cercheremo di presentare alcune possibili tecniche di difesa.

1.2.1 Bruteforce su login

Un primo attacco molto semplice da effettuare è un attacco di tipo bruteforce sul login. L'attaccante prepara un vasto numero di possibili password e le manda una alla volta al server web imitando dei tentativi di login da parte dell'utente. Queste liste di password vengono chiamate dizionari [13], e ne esistono di ogni grandezza, ordinate alfabeticamente o per rarità delle password; possono essere comprate sul web e vengono messe a disposizione gratuitamente da strumenti di testing come *John the Ripper* [14]. Con il progetto presentato in questa tesi è possibile creare dizionari personalizzati di questo tipo.
Essendo un attacco molto semplice, è possibile difendersi in modo altrettando semplice. La difesa più efficace e poco costosa da attacchi di questo tipo, deve essere implementata dal server del servizio web, limitando il numero di richieste di login che un client può sottomettere al server. Aggiungere un tempo di attesa di soli 100 millisecondi rende gli attacchi di tipo bruteforce completamente inutilizzabili. Un altro tipo di difesa applicabile è il captcha, un metodo per riconoscere client umani da client non umani attraverso la sottomissione al client di un problema computazionalemnte molto difficile.
Per quanto riguarda l'utente, l'unica difesa applicabile è la scelta di una password poco comune. Molti dizionari vengono costruiti a partire da collezioni di password reali, perciò è importante che sia poco probabile che la password scelta sia già stata usata da altri utenti.
Esiste anche un altro tipo di attacco bruteforce, che invece di utilizzare dizionari di attacco, tentano tutte le permutazioni di lettere minuscole, lettere maiuscole e caratteri speciali in stringhe solitamente non più lunghe di 12-14 caratteri. Per questo motivo è consigliato utilizzare una password molto lunga, sebbene questi tipi di attacchi vengano usati molto raramente.

1.2.2 Sql injection

Le sql injection sono una tipologia di attacco più articolata; permettono all'attaccante di iniettare codice sql arbitrario all'interno delle normali query sql utilizzate dal servizio per controllare le credenziali dell'utente o per effettuare la registrazione.
I punti di accesso per questo tipo di attacco sono quasi sempre le form html, come quella di login, la registrazione o la casella di ricerca interna al sito web. Una sql injection consiste nell'inserire all'interno della query preformata presente nel web service, del codice sql arbitrario in modo tale che la sintassi della stringa risultante rimanga corretta. Un attacco di questo tipo potrebbe garantire l'accesso senza conoscere nessuna chiave di autenticazione, oppure potrebbe scaricare l'intera tabella o addirittura cancellarla. Molte delle password reali che si trovano sul web sono state rubate tramite attacchi di sql injection.
Un esempio di attacco su un sistema di autenticazione vulnerabile: porzione di codice dello script che si occupa del controllo delle credenziali:

```
user = form.get("username")
pass = form.get("password")
res = db.query("select * from users where user='"+user+"'
and pass = '"+pass+"'")
```

la stringa da iniettare nel campo username (anche nel campo password, è indifferente) per accedere al servizio senza la chiave di accesso potrebbe essere questa:

```
' or 1=1 --
```

Il token "–" serve per commentare il resto della stringa.
sostituendo nella query preimpostata:

```
risultato = db.query("select * from users where user=''
or 1=1 -- ' and pass = ''")
```

Per difendersi da questi tipi di attacchi il web service deve effettuare dei controlli sul testo proveniente dall'input in modo tale da non accettare stringhe che contengono caratteri speciali delle query sql come "-", "&" e "apice". Tutti i servizi web che non effettuano controlli sintattici di questo tipo, o che li effettuano in modo errato, sono vulnerabili alle sql injection [15]. Queste vulnerabilità sono ancora molto comuni nonostante esistano degli standard di sanitizzazione dell'input, e sono dovute a negligenze nello sviluppo di applicazioni che dovrebbero garantire la sicurezza dei dati degli utenti [16].

1.2.3 Packet sniffing

Il packet sniffing riguarda la comunicazione tra client e server, e cerca di estrapolare informazioni sensibili dalla comunicazione tra i due attori [18]. L'attacco consiste nello stare in ascolto sul canale di comunicazione tra client e server per leggere il traffico prodotto dalle due parti. Nel caso in cui venga utilizzato il protocollo http e non https, nel momento in cui l'utente spedisce al server le credenziali per il login, l'attaccante può leggere il messaggio http ed estrarre le informazioni al suo interno; per esempio nome utente e password.
In questo caso l'attaccante si deve trovare in un canale di comunicazione condiviso con l'utente, ovvero collegati allo stesso gateway. Ambienti di questo tipo, e quindi rischiosi per la sicurezza delle comunicazioni possono essere connessioni wifi di bar, biblioteche e luoghi pubblici [19].
In generale, per difendersi da questi attacchi, bisogna sempre essere certi di essere collegati al servizio web tramite protocollo https, e non http; in caso contrario potrebbe essere rischioso instaurare la comunicazione.
Misure di sicurezza aggiuntive posso essere fornite dai router wifi, che possono criptare la comunicazione con gli host connesi. In questo modo gli altri host connessi allo stesso router non posso leggere facilmente il contenuto dei pacchetti.

1.2.4 Hash reversing

Come descritto in precedenza, le password vengono salvate sul server sotto forma di hash per proteggerle in caso di attacchi informatici. Tuttavia esistono diverse tecniche per riuscire a ricavare la password generante da un hash. È sempre possibile usare un approccio di tipo bruteforce: vengono generate password casuali che vengono passate alla funzione di hashing controllando che il risultato coincida con l'hash. Per quanto semplice, questo tipo di attacco impiega un'enorme quantità di tempo e risorse computazionali.
Una variante dell'attaco bruteforce si serve di dizionari contenenti una vasta collezione di password, solitamente ordinate per ricorrenza, ovvero la probabilità che vengano utilizzate. Come descritto nella sezione sul bruteforce su login, questo è un attacco molto veloce, che riesce subito a trovare le password più comuni.
Tuttavia esiste un altro tipo di attacco molto più efficace che fa utilizzo di rainbow table [20], ovvero di liste di coppie di valori (hash, password) precomputate. Nel caso in cui la lista venga mantenuta ordinata per hash, la complessità computazionale per ricavare una password dal suo hash diventerebbe logaritmica rispetto alla dimensione della lista; nel caso di una rainbow table di 100 milioni di coppie, si riuscirebbe a trovare una password in soli 26 passi. Un computer moderno riuscirebbe a decifarre un intero database di chiavi di autenticazioni in qualche decina di ore.
L'approccio difensivo da utilizzare contro tutte le tecniche descritte in questa sezione è la scelta di una password lunga, ma che sia soprattutto poco probabile che sia già stata usata in precedenza da altri utenti, o dallo stesso utente su più servizi diversi. In generale una volta che una password viene scoperta è prudente pensare che quella password sia stata aggiunta ad una rainbow table.
Quasi la totalità dei furti di identità avviene attraverso l'utilizzo di questa tecnica, perciò è molto importante difendersi da essa efficacemente.
Oltre alla scelta di una buona password da parte dell'utente, esiste una tec-

nica di difesa applicabile dal web server chiamata salting [21], che consiste nell'appendere alla password una stringa casuale prima di passarla alla funzione di hashing. La stringa, chiamata salt, viene poi salvata nel database accanto all'hash; durante il processo di login il server riceve la password immessa dall'utente dalla form e la passa alla funzione hash appendendo il salt recuperato dal server. Questa tecnica rende inutilizzabile attacchi che fanno uso di rainbow table. Tuttavia se la stringa di salt è troppo corta, l'attaccante può comunque precompilare una rainbow table con tutte le possibili combinazioni di salt e password. Per esempio nei vecchi sistemi unix le password degli utenti venivano salvate sotto forma di hash con 12 bit di salt, ovvero 4096 possibli combinazioni. Con l'aumento delle capacità di memoria e di prestazione dei sistemi è stato necessario aumentare la lunghezza del salt.

Capitolo 2

Ambiente di acquisizione di password reali

L'obiettivo principale di questa tesi è la progettazione e lo sviluppo di un sistema software per l'acquisizione di una collezione di password reali più vasta possibile, in modo da poter effettuare degli studi statistici sulle password scelte dagli utenti al fine di proteggere la riservatezza dei nostri dati tramite la scelta di password più sicure.
Le password che devono essere raccolte vengono solitamente condivise sul web a seguito di attacchi informatici rivolti a servizi web, e sono spesso sotto forma di file di testo chiamati "data leak" o "dump" contenenti le chiavi d'autenticazione, solitamente nella forma di coppie *email:password*. Questi file vengono quindi raccolti ed elaborati dal sistema, e le password estratte vengono salvate in un database.
La prima sezione di questo capitolo si occupa di illustrare i canali di comunicazione dai quali il sistema preleva le password; successivamente verrà presentata la progettazione e l'implementazione dei vari moduli software che compongono il sistema automatico di acquisizione di password.

2.1 Ricerca di data leaks

Molto spesso, a seguito di attacchi informatici, vengono rilasciati pubblicamente sul web interi database di chiavi di autenticazione. Ci si riferisce ad essi come data leak, oppure database dump (spesso abbreviati rispettivamente in "leak" e "dump").
L'acquisizione di password avviene attraverso la ricerca di questi leak, rilasciati sotto forma di file di testo sulla rete attraverso dei servizi di file sharing, quali pastebin, filetube e il network torrent. Alcuni dei leak più importanti possono arrivare ad avere qualche milione di chiavi; per esempio l'attacco a RockYou che nel 2009 scoprì più di 32 milioni di password, o a Zhenai nel 2011 con circa 3 milioni di account compromessi.
La maggior parte delle password raccolte nella collezione deriva da leak di grandi dimensioni, ma lo sforzo maggiore deriva dalla ricerca di piccoli leak che vengono costantemente rilasciati sul web, ed è su questa ricerca che viene concentrata l'attenzione di questo documento.

2.1.1 Canali di comunicazione di rilascio

2.1.2 Twitter

Quando un file di testo viene caricato su pastebin viene automaticamente reso pubblico e indicizzato; questo rende possibile la scrittura di software capace di leggere ogni nuovo file indicizzato pubblicamente, alla ricerca di pattern di interesse; nel nostro caso, pattern che suggeriscano la presenza di possibili chiavi di autenticazione. Un software già esistente che esegue questa ricerca è @PastebinLeaks [22]; uno strumento automatico che utilizza Twitter per comunicare pubblicamente i nuovi pastebin i quali vengono riconosciuti come possibili leak. I tweet prodotti da @PastebinLeaks presentano una piccola introduzione e un url abbreviato al file di testo.
Una copia di ogni tweet viene salvato nel database e il modulo di acquisizione

utilizza il link in esso contenuto per scaricare il file di testo, il quale viene analizzato alla ricerca di chiavi che vengono estratte e inserite nella collezione.

2.1.3 RSS

Spesso i leak vengono condivisi sul web su servizi di condivisione file, per esempio filetube, o su piattaforme sociali nell'ambito della sicurezza informatica, come forum e blog. Per il monitoraggio automatico di questi canali viene utilizzato il servizio *Rich Site Summary* (RSS) per leggere eventuali nuovi contenuti. RSS è un formato di distribuzione di contenuti web che utilizza XML e viene spesso usato per la presentazione di meta dati riguardanti la pagina; nell'esempio di un blog di informazione, il titolo dell'articolo, la data, i tag e un breve riassunto. Applicazioni che fanno uso degli stream RSS prendono il nome di RSS feeder, i quali permettono all'utente di vedere quando è stato pubblicato qualcosa di nuovo sui siti di interesse senza dover visitarli uno per uno. Nel progetto viene utilizzato un semplice feeder per essere notificati all'inserimento di nuovi contenuti sui canali di condivisione che decidiamo di monitorare.
Gli url alle pagine RSS sono salvati nel database e possono essere aggiunti anche dall'interfaccia di amministrazione. Alcuni servizi, come filetube, prevedono l'aggiunta di una stringa di ricerca al link RSS per filtrare preventivamente i contenuti di interesse; per esempio: *http://www.filestube.to/file/rss.rss?q=password%20leak* e *http://www.filestube.to/file/rss.rss?q=database%20dump*.

2.1.4 Google Alerts

Il servizio Alerts di Google [23] permette all'utente di utilizzare il potente crawler di Google per essere notificati quando un nuovo contenuto di interesse viene scoperto sul web. Il servizio alerts prevede che l'utente si autentichi con il suo account Google e immetta delle parole chiave di ricerca per attivare un nuovo "alert". È inoltre possibile specificare la frequenza con cui si vuole

essere notificati, il tipo di fonti (blog, forum, news, etc.), la lingua e il modo con cui si vuole essere notificati. Il servizio prevede l'invio di notifiche via mail oppure sfruttando un flusso RSS da poter eventualmente aggiungere al proprio feeder RSS. La nostra piattaforma utilizza quest'ultima modalità in quanto, come spiegato nella sezione precedente, abbiamo già a disposizione un modulo di acquisizione da canali RSS.

2.2 Architettura dell'ambiente di acquisizione

Il sistema di acquisizione di password è organizzato come segue: tutte le informazioni e i dati (relativi a password, RSS e link) sono mantenuti in un *Database Management System* (DBMS) relazionale, il quale viene utilizzato, letto e modificato dai programmi che si occupano dell'acquisizione di password e della reportistica. Il software è scritto per la maggior parte in python ed è stato organizzato in moduli, ognuno con la sua funzione. Gli script possono essere lanciati manualmente oppure attraverso l'utilizzo di un servizio web per la gestione della piattaforma; dalla quale è anche possibile aggiungere link RSS, Google alerts e nuovi account Twitter da seguire.
La piattaforma è completamente portabile su qualsiasi sistema con un interprete python e un web server con supporto *Common Gateway Interface* (CGI).
La macchina utilizzata in locale per lo sviluppo dell'applicazione è una macchina con sistema operativo Linux Ubuntu [24], python 2.7.6, MariaDB come server MySQL e Apache 2.4.7 come web server.

2.2.1 Database

La base di dati consiste di 5 tabelle:

1. Passwords
2. rss_feeder

3. rss_links
4. tweetFollower
5. tweets

Nella tabella *Passwords* viene mantenuta l'intera collezione di password. La struttura della tabella è la seguente: una stringa *password* per contenere la password e un'intero *occurrences* che mantiene la frequenza di utilizzo di quella password. È presente un indice ordinato sul campo *password*, il quale è anche chiave primaria. Il campo occurrences viene incrementato nel caso in cui il modulo di estrazione trovi una password già presente nella collezione. Rss_feeder e rss_links contengono rispettivamente la lista dei feeder RSS e la lista dei link estratti dai vari feeder. Similmente a quanto accade per gli RSS feeders, le due tabelle tweetFollower e tweets contengono la lista di account Twitter da cui estrarre i tweet che verranno poi salvati nella seconda tabella.

2.2.2 Architettura software

L'ambiente di acquisizione è strutturato in moduli software, ognuno dei quali offre diverse funzionalità. La catena di acquisizione è composta principalmente da tre fasi: web scraping, estrazione dei dati di interesse e aggiornamento del database. La prima fase consiste nell'analizzare le risorse del web e scaricare il materiale di interesse; la seconda fase estrae le password dal materiale scaricato e l'ultima fase consiste nell'inserimento delle password nella collezione. Per la seconda e la terza fase sono state sviluppate delle librerie che raccolgono funzionalità comuni, come l'inserimento di una password nella collezione, l'estrazione di una password da una stringa http o l'esecuzione di una query sul database. Le due librerie, *passdb* e *parser*, verranno descritte più dettagliatamente nelle prossime sezioni. La prima fase, generalizzata come "web scraping ", viene eseguita dai moduli specifici per ognuna delle sorgenti dati Twitter e feed RSS. È inoltre presente una libreria contenente delle funzionalità di carattere più generale, come funzioni di logging, risoluzione url abbreviati etc. etc.

2.2.3 Middleware per l'accesso al database

Per l'accesso al database è stato creato un layer intermedio che si interfaccia con le API mySql di python e che offre al livello superiore delle funzioni ottimizzate per la scrittura nel db. La necessità della creazione di un middleware è nata dal problema dell'inserimento di un gran quantitativo di password nel database. Inserire una password alla volta si è rivelato altamente inefficiente a causa dell'elevato overhead prodotto dal DBMS. Per ogni query infatti viene creato dal DBMS un nuovo ambiente di esecuzione: vengono aggiornati file e allocate strutture dati in memoria le quali saranno poi deallocate al termine dell'esecuzione delle query.
L'overhead generato viene sommato per ogni query effettuata, e con molte migliaia di password estratte in pochi secondi diviene impossibile per l'inserimento stare al passo con l'estrazione. Per questo sono state prese in considerazione varie soluzioni: l'approccio che si è scelto utilizzare è quello di inserire un alto numero di record in una query di grandi dimensioni. Per fare ciò è stato sufficiente concatenare i nuovi elementi nel campo *values* della query INSERT [25].

2.2.4 Moduli di alimentazione

I moduli di alimentazione hanno il ruolo di raccogliere quanto più materiale possibile dai canali di comunicazione predefiniti. Il materiale raccolto deve essere il più significativo possibile, ovvero deve essere massimizzato il numero di chiavi di autenticazione contenute nei file di testo scaricati, ma è spesso richiesto l'intervento manuale per distinguere materiale significativo dai falsi positivi.

La raccolta da canali RSS viene gestita come segue. Gli URI (Uniform Resource Identifier) [26] vengono inseriti manualmente in una lista mantenuta nel database. L'inserimento di un nuovo URI nella lista degli RSS può avvenire tramite la console mysql o tramite l'interfaccia web gestionale. Questi URI vengono letti dal software di polling che scarica la pagina RSS

in formato XML relativa all'URI specificato. Dalla pagina XML vengono estratti gli oggetti e vengono lette le sue proprietà; quelle che ci interessano sono il titolo e il link al file. Per prima cosa deve essere recuperato dal database l'indice dell'ultimo oggetto già considerato in passato dallo stream RSS; in questo modo vengono prese in considerazione solo le nuove entry. Per memorizzare l'ultimo oggetto letto dallo stream, viene salvato nel database il suo link sotto forma di hash. Per la costruzione dell'hash viene applicata una funzione di hashing md5 prima dell'inserimento nel database.

Dopo che l'hash dell'ultima entry viene letto dal sistema, si può procedere con l'inserimento nel database di nuovi record. Per ogni oggetto vengono salvate le proprietà "titolo" e "link" in un nuovo record nella tabella rss_links. Le entry della tabella rss_links devono essere controllate manualmente per decidere quale file scaricare e quale invece scartare.

L'acquisizione da Twitter risulta leggermente più complicata. L'accesso ai dati di un account Twitter, come la sua timeline per esempio, deve essere effettuato tramite l'utilizzo delle API Twitter [30]. Quest'ultime sono organizzate attraverso un'architettura Representational State Transfer (REST) [27]. La comunicazione con le RESTful API di Twitter avviene attraverso il protocollo HTTP tramite l'invio di richieste GET e POST. Per ottenere accesso ai dati è prima necessario autenticarsi ad un account. Per il processo di autenticazione, le API di Twitter utilizzano il sistema OAuth [28], un protocollo di autenticazione open e sicuro. Le chiavi OAuth, che vengono rilasciate previa registrazione della app al servizio e vengono inserite in ogni richiesta HTTP alle API. L'implementazione del software è avvenuta mediante l'utilizzo della libreria tweepy [29], che nasconde allo sviluppatore l'uso del protocollo HTTP per interfacciarsi alle API di Twitter, favorendo un approccio ad oggetti.

Tramite l'uso delle API è quindi possibile ottenere l'accesso alla timeline dei profili Twitter così da poter scaricare il loro contenuto e, come nel caso degli stream RSS, inserire le nuove entry nel database. A differenza dei canali RSS non è necessario usare l'hash del link per tenere traccia dell'ultimo og-

getto già inserito, in quanto ogni tweet possiede già un numero identificativo univoco all'interno dello stream. Appena scaricati, i tweet vengono analizzati e viene estratto l'URL della risorsa dal testo del messaggio. L'URL, previa risoluzione dell'abbreviazione, viene utilizzato per scaricare il contenuto del pastebin; il quale viene passato al parser per la ricerca di pattern riconosciuti. Le chiavi estratte dal parser vengono successivamente inserite nel database.

2.2.5 Parsing ed estrazione

Il parser ha il compito di riconoscere dei pattern conosciuti all'interno delle stringhe e di estrarre le password da esse. I pattern principali presi in considerazione possono essere del tipo (molto comuni nei file provenienti da pastebin) http come http://username:password@members.url.com/ e http://username:password@url.com/members/; oppure del tipo user@host.dom:password o user@host.dom;password; in questo caso sono stati presi in considerazione solo due tipi di divisori: ":" e ";", essendo utilizzati nella quasi totalità dei casi. Il riconoscimento di password all'interno di file senza pattern fissi oltre ad essere un compito molto complicato è anche rischioso, in quanto è possibile che vengano estratte stringhe che si rivelavano non essere delle password. I file nei quali non vengono riconosciuti dei pattern vengono comunque mantenuti, ed è possibile agire manualmente creando delle funzioni di parsing ad hoc.

2.2.6 Analisi statistica

L'analisi statistica della collezione di password è un problema molto dispendioso in termini di risorse. Con un numero di record dell'ordine delle decine di milioni bisogna ottimizzare gli accessi al database, per esempio raggruppando analisi statistiche che interessano proprietà comuni, come la lunghezza o l'analisi dei caratteri. Le funzioni di analisi vengono raggruppate in sottogruppi, ognuno dei quali effettua una query al database specifica per quel problema e distribuisce i dati estratti alle varie procedure. Le funzio-

ni di analisi possono restituire liste di password, indici numerici o grafici; quest'ultimi vengono salvati in formato png [31] e possono essere utilizzati in pagine web o fogli di reportistica. La creazione dei grafici viene gestita tramite le librerie plotly [32].

2.2.7 Ambiente web gestionale

L'ambiente di acquisizione è gestito tramite un'interfaccia grafica che espone diverse funzionalità. Per prima cosa vengono presentate delle statistiche di base sul contenuto del database, come il totale di password raccolte, il numero di password uniche e la data dell'ultimo aggiornamento delle statistiche. Dopo la prima parte introduttiva sono presenti le form per la gestione dell'acquisizione dei dati dal web. Tramite le form è possibile inserire nuovi stream RSS da seguire, nuovi account Twitter o nuovi Google alerts.

Il servizio web usa le tecnologie di markup HTML e CSS per l'interfaccia client, mentre il servizio lato server è stato sviluppato in python utilizzando il template engine Jinja2 [33]. La scelta della tecnologia lato server python è dovuta alla facilità di integrazione con il resto dell'ambiente software.

Capitolo 3

Analisi dell'utilizzo di password

In questo capitolo verranno presentati gli studi effettuati sulla collezione di password raccolte, con l'intento di studiare la scelta di password da parte degli utenti.
Prima della presentazione dei dati è necessario puntualizzare alcuni problemi riguardanti la natura delle password contenute nella collezione che si riflettono inevitabilmente nei risultati delle analisi. La quasi totalità delle password raccolte derivano da database nei quali le chiavi d'autenticazione vengono mantenute sotto forma di hash, e le liste di password in chiaro vengono prodotte tramite il reversing di questi hash. Come descritto nel capitolo 1 il reversing degli hash può avvenire attraverso l'utilizzo di varie tecniche di bruteforce, ma le due tecniche utilizzate nella pratica sono gli attacchi a dizionario e le rainbow table. L'uso di queste due tecniche permette la risoluzione della maggior parte degli hash in poco tempo, ma le password più complicate (non presenti nei dizionari) non vengono tradotte. Questo significa che l'insieme di password sul quale verranno svolti gli studi non rispecchia fedelmente l'insieme di password reali utilizzate dagli utenti, pertanto alcune statistiche saranno alterate da questo fatto. Mediamente circa il 95% del database viene tradotto con successo, pertanto l'errore propagato è minimo. Inoltre una parte significativa del materiale sorgente ha radici geografiche appartenenti all'Europa dell'est e alla Russia non rispecchiando l'effettivo

partizionamento geografico del bacino di utenti del web. Nessuna sorgente asiatica è stata considerata per problemi riguardanti la differenza di alfabeti utilizzati. Questi fatti agiscono su alcune delle statistiche riportate di seguito.

3.1 Statistiche

Fino ad ora sono state raccolte **24,391,953** password.
La collezione contiene **5,257,174** password uniche, il che vuol dire che mediamente ogni entry è stata trovata **4,64** volte.
Il **43%** delle password, ovvero **2,299,835**, ha più di un'occorrenza, mentre il restante **56%** (**2,957,339**) è costituito da password che sono state trovate una sola volta.
Di seguito verranno presentate le 20 password più comuni con le rispettive occorrenze:

#	password	occorrenze
1	qwerty	499691
2	123456	199672
3	123456789	66688
4	qwertyuiop	60701
5	1qaz2wsx	26395
6	qazwsx	24815
7	111111	24557
8	12345678	23615
9	qwe123	23022
10	1q2w3e4r	22869
11	qwer1234	17685
12	1Q2W3E4R5T	17601
13	1234567	16921
14	123qwe	16524
15	1234567890	16003
16	12345	15762
17	123321	15100
18	123123	15086
19	Klaster	13832
20	password	13504

Tabella 3.1: Lista delle 20 password più comuni

I dati appena presentati sono indicativi di una scelta di password troppo comuni da parte degli utenti.
Si nota subito la predominanza di password keyboard bound, ovvero combinazioni di tasti che formano pattern molto semplici sulla tastiera, come "qwerty", la più comune, formata dalle prime cinque lettere della tastiera; oppure serie numeriche banali, come "123456", "123456789" o "111111". La necessità di dover ricordare molte password porta alla scelta di chiavi che richiedono poco sforzo mnemonico, se non nessuno, come le combinazioni keyboard bound appena viste. Inutile dire che si tratta di un grande pericolo

per la riservatezza dei propri dati.
Un altro aspetto molto interessante è la rapidità con cui il numero di occorrenze cala. Dal grafico in figura 3.1, che mostra le occorrenze delle prime 1000 password, si nota un andamento esponenziale inverso.

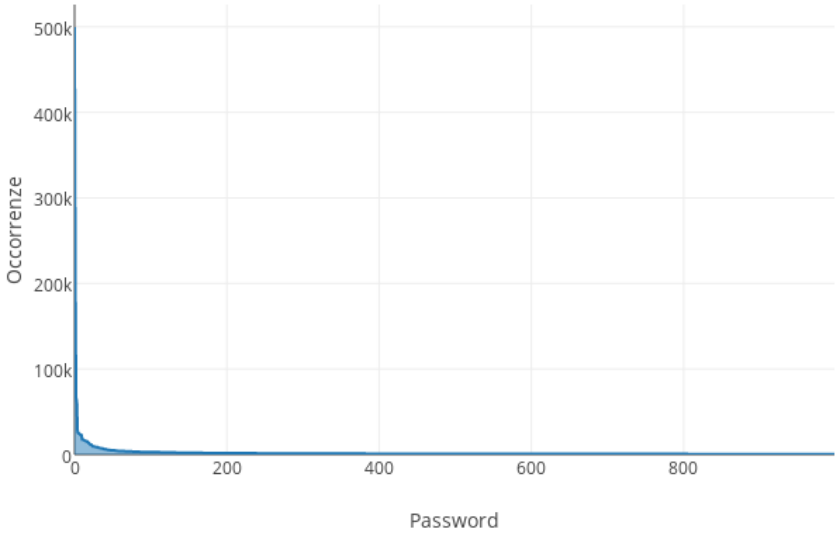

Figura 3.1: Distribuzione delle password in base alle occorrenze (limitato alle prime 1000)

Le password più comuni hanno una predominanza talmente forte che con le sole prime **4** password della lista (lo **0,000000761%** delle password totali) possiamo coprire il **3,5%** della totalità della collezione.

La lunghezza è un aspetto importante delle chiavi d'autenticazione, sia per difesa contro attacchi bruteforce sia per la correlazione che ha con la rarità di una password. In generale password più lunghe sono anche password più sicure.

La figura 3.2 mostra la relazione tra il numero di chiavi e la loro lunghezza. La statistica non tiene conto delle occorrenze delle singole password.

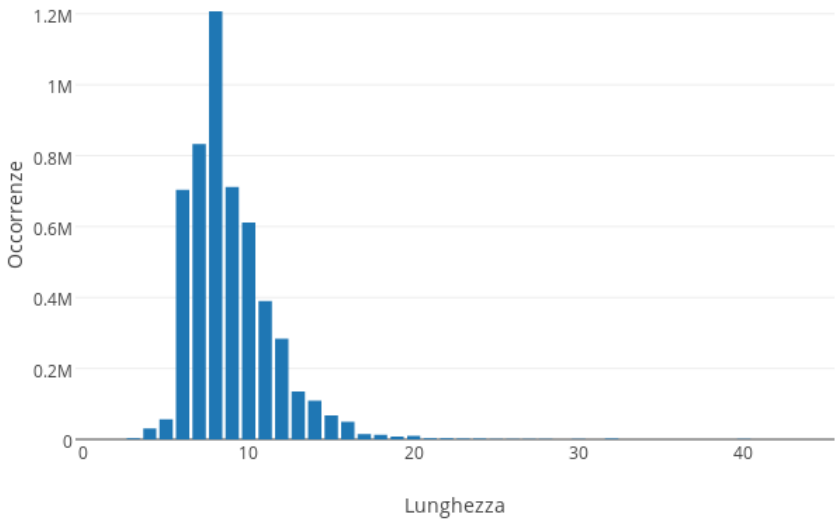

Figura 3.2: Distribuzione delle password in base alla lunghezza

Dalla figura 3.2 si nota che la maggior parte delle password è concentrate in un intervallo di lunghezza dai 6 ai 9 caratteri, assolutamente insufficienti contro attacchi di tipo bruteforce.
Le chiavi con 5 caratteri o meno sono solo l'**1,73%** (**90,846**) della totalità, e le chiavi con più di 10 caratteri coprono il **20,9%** (**1,100,744**) della collezione. La restante parte che va dai 6 ai 10 caratteri copre circa il **77,4%** della totalità delle password.

Di seguito verranno presentate delle statistiche riguardanti la composizione sintattica delle password. Lo studio più interessante è quello rivolto ai caratteri speciali, ovvero caratteri che non appartengono agli alfabeti a-z e 0-9, come la punteggiatura o simboli matematici. Da tempo ormai molti servizi web pretendono che vengano inseriti dei caratteri speciali all'interno

delle nuove password come ulteriore misura tutelativa dell'utente, e il loro studio può essere molto utile agli algoritmi di brute forcing per affinare la ricerca delle password.
La lista dei caratteri speciali dal più comune al più raro vengono presentati in figura 3.3.

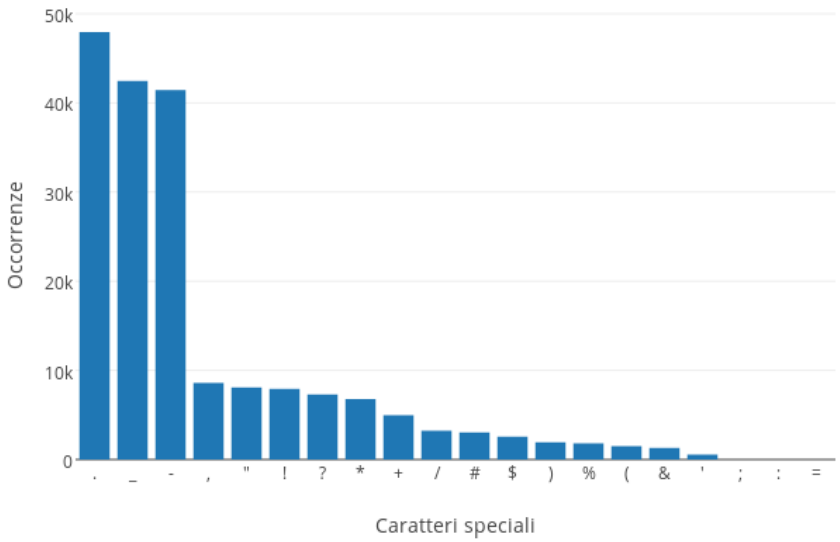

Figura 3.3: Occorrenze per caratteri speciali

I 3 caratteri predominanti sono il punto, e i simboli "-" e "_"; quest'ultimi vengono usati soprattutto per dividere più parole nella stessa password. Per il resto dei caratteri l'andamento è abbastanza lineare.
Ritornando all'affinamento della ricerca di password da parte di algoritmi di bruteforce, un metodo per ridurre drasticamente il tempo di esecuzione potrebbe essere l'uso dei soli primi 3 caratteri speciali, in quanto coprirebbero il **68,84%** delle combinazioni totali.

Analogamente a quanto fatto per i caratteri speciali, la figura 3.4 presenta la distribuzione dei caratteri alfabetici. La statistica comprende sia le lettere minuscole che le maiuscole.

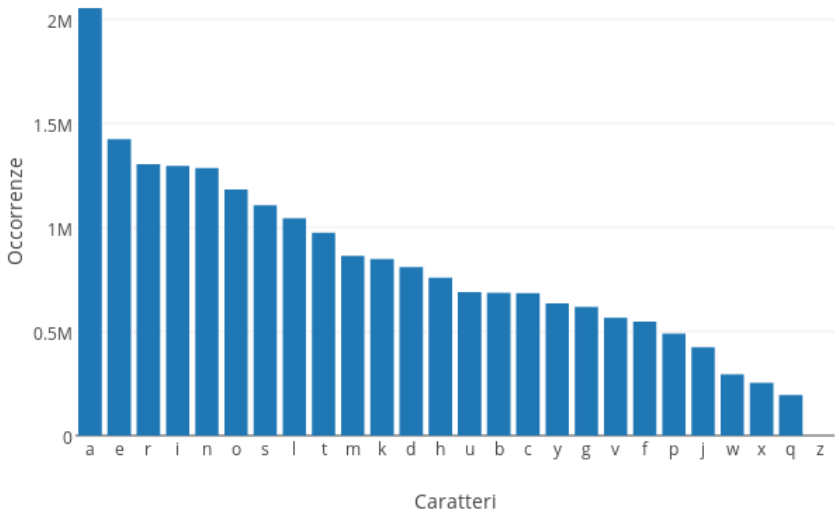

Figura 3.4: Occorrenze per caratteri alfabetici

La distribuzione è abbastanza lineare, con le vocali prevedibilmente nelle prime posizioni, e la lettera *a* con un buon distacco sulle altre lettere.

È stato svolto uno studio sulla presenza di informazioni personali degli utenti all'interno delle loro password; quelle che abbiamo deciso di prendere in considerazione sono informazioni che possono essere trovate facilmente, come il nome e la data di nascita.
Tramite la ricerca di numeri di 4 cifre che compongono numeri nell'intervallo 1900-2099 sono state riscontrate **19,161** corrispondenze pari allo **0,35%** del totale; il che indica che le date siano un aspetto rilevante ma non predominante nella scelta di una password.
L'uso della data di nascita all'interno della propria password è una scelta controversa. Essendo una informazione personale è sicuramente un elemento di debolezza quando si tratta di difendersi da attacchi mirati a scoprire la password di un utente in particolare, in quanto può essere utilizzata in tec-

niche di bruteforcing intelligenti che utilizzano le informazioni riguardanti la vittima per ridurre il numero di tentativi necessari per indovinare la chiave. Tuttavia può anche essere un ottimo elemento da aggiungere alla propria password per aumentarne la sicurezza. La propria data di nascita è infatti immediata per quanto riguarda lo sforzo mnemonico, e può essere vista come una tecnica di salting.
Lo stesso studio svolto sulle date di nascita è stato fatto utilizzando una lista di 4,347,667 nomi propri di persona costruita dal gruppo SkullSecurity [34] raccogliendo nomi reali dal social network Facebook. Le corrispondenze trovate sono **91,778** pari al **2.11%** del totale, ovvero circa 4,5 volte maggiore delle date di nascita.

Uno studio interessante è stato svolto comparando la nostra collezione con dei dizionari precompilati utilizzati dai software di password cracking per svolgere attività di bruteforce. Le wordlist prese in considerazione sono un dizionario ufficiale di Cain [35] composto da 306,706 password e un dizionario ridotto di John The Ripper [36], contenente sole 3,546 password. Lo studio è stato svolto cercando il numero di corrispondenze che le parole dei due dizionari trovavano nella collezione.
Con il dizionario di Cain sono state trovate **13,273** (**4,33%**) corrispondenze nella collezione, una percentuale inaspettatamente bassa, che potrebbe indicare una differenza sostanziale nel materiale sorgente usato per costruire il dizionario.
Il dizionario di John The Ripper ha avuto invece una buona corrispondenza con il **35,4%** di password trovate nella collezione.
Lo stesso procedimento è stato usato con una serie di vocabolari inglesi di diverse dimensioni presi anch'essi dalla repository FTP di John.
I 3 vocabolari, contenenti 444,678, 390,532 e 296,809 parole rispettivamente, hanno riscontrato tutti e tre una corrispondenza del circa **4%**.

Capitolo 4

Scelta di una password efficace

In questo capitolo si vuole discutere della scelta di password efficaci tramite l'utilizzo delle informazioni ricavate dagli studi statistici, e tramite l'uso di alcune tecniche per la costruzione di password sicure ma soprattutto facili da ricordare.
L'esposizione avverà simulando la scelta di una password da parte di un utente immaginario con il bisogno di attivare un account Gmail, il servizio di posta elettronica di Google.

Innanzitutto è necessario creare il corpo della password: abbiamo bisogno di trovare una stringa di caratteri, non necessariamente molto lunga, che non appartenga a nessun vocabolario ne a dizionari di nomi propri. Una stringa di questo tipo potrebbe essere un insieme casuale di caratteri senza alcun senso semantico, ma una stringa così generata non si presta bene ai nostri bisogni in quanto si tratta di un elemento difficile da ricordare. Un buon compromesso tra sicurezza e semplicità potrebbe essere l'uso di permutazioni di parole esistenti, e quindi facili da memorizzare. Immaginiamo che l'utente si chiami Alessio: vogliamo costruire una stringa apparentemente casuale a partire dal suo nome. Tenendo come primo obiettivo la semplicità della password, la permutazione più immediata da applicare è il roll, ovvero una rotazione dei caratteri in cui la parola viene gestita come una stringa

circolare. Prendendo in considerazione il nome proprio "Alessio" possiamo applicare un roll4 verso destra trasformandolo in **"ssioAle"**.

Nel capitolo precedente è stata discussa l'ipotesi di aggiungere la data di nascita alla password per allungarla, adempiendo anche al requisito dell'inserimento di caratteri numerici all'interno della chiave, ormai imposto dalla maggioranza dei servizi web. Dopo l'aggiunta della data (immaginando che il nostro utente sia nato nel 1980) la nostra password risulta essere **"ssio1980Ale"**.

Sebbene non presente in nessuno dei dizionari, ne' nella nostra collezione, la password è ancora molto fragile nei confronti di attacchi bruteforce lessicografici.
Lo strumento di difesa migliore da attacchi bruteforce è l'aggiunta di caratteri speciali. Abbiamo visto in figura 3.3 che i 3 caratteri speciali più usati sono, in ordine, il punto, l'underscore e il meno. Per aumentare al massimo la sicurezza della nostra chiave useremo due caratteri speciali distinti che sceglieremo in modo tale che siano facili da ricordare. Una buona coppia di caratteri potrebbero essere l'uguale ("=") e il punto di domanda ("?"); si tratta di una buona scelta in quanto sono caratteri raramente usati e sono vicini nella tastiera, in modo da facilitarne la memorizzazione.

Con l'ultimo passo la nostra nuova chiave è diventata **"=ssio1980Ale?"**.
In termini di sicurezza è una chiave molto sicura e praticamente introvabile dagli strumenti tecnologici attuali; le permutazioni di 13 caratteri che utilizzano i caratteri alfanumerici e caratteri speciali sono 90^{13}, ovvero $2,5 * 10^{25}$.
Sfortunatamente non dobbiamo solo fidarci della sicurezza della nostra password, ma anche della sicurezza del servizio web al quale ci iscriviamo sul quale non abbiamo nessun controllo.
Se il server salva le chiavi d'autenticazione in modo non sicuro, per esempio non utilizzando nessuna tecnica crittografica, o utilizzandola in modo erra-

to, è possibile che a seguito di attacchi informatici la nostra password per quanto sicura vada a finire nei dizionari e nelle rainbow table di strumenti di password cracking e di password recovery, rendendo inutile lo sforzo finora compiuto.
Per questo motivo è essenziale utilizzare password diverse per ogni servizio a cui ci si iscrive, ma ricordare una chiave diversa per ogni account che possediamo è pressoché impossibile. Un buon compromesso è aggiungere alla propria password una lettera che faccia riferimento al servizio web; nell'esempio possiamo aggiungere la lettera "G" all'inizio della password in quanto ci stiamo iscrivendo al servizio Gmail di Google; similmente avremmo aggiunto la lettera "F" per Facebook o "T" per Twitter.

In conclusione la password che abbiamo creato è **"G=ssio1980Ale?"**. È abbastanza facile da ricordare, in quanto usa informazioni immediate come il nome, la data di nascita e la prima lettera del servizio sul quale vogliamo effettuare il login. Ha una lunghezza di 14 caratteri e usa 3 diversi alfabeti, abbastanza da scoraggiare qualsiasi tipo di attacco bruteforce; ed è abbastanza complessa da non essere già presente in nessun dizionario esistente. Inoltre, con la tecnica dell'inserimento della prima lettera del nome del servizio web all'interno della nostra chiave, è possibile generare un set di password tutte diverse tra loro senza nessuno sforzo mnemonico.

Capitolo 5

Conclusioni

In questa tesi abbiamo discusso della sicurezza dei sistemi di autenticazione con password. Per prima cosa abbiamo descritto come vengono implementati questi sistemi, analizzando sia il lato server, sia il lato client, e abbiamo inoltre descritto i protocolli di comunicazione tra le due parti.
Abbiamo imparato come possono essere gestite le password dal client quando vengono raccolte dal form HTML e come è possibile mandarle al server tramite il protocollo HTTP e quanto sia importante l'utilizzo dell'estensione criptata HTTPS. Infine abbiamo studiato come vengono gestiti dal server i database contenenti gli account degli utenti con le rispettive chiavi d'autenticazione, e quali sono i pericoli derivanti dall'uso di funzioni di hash poco sicure come md5.

Successivamente sono state presentate alcune delle tecniche di attacco che possono mettere a repentaglio la sicurezza delle nostre informazioni, come SQL injection, Packet sniffing e Hash reversing. Sono state presentate tecniche di bruteforcing con l'introduzione dei dizionari e delle rainbow table. Per ogni tecnica di attacco si è cercato di studiare delle tecniche di difesa e misure preventive applicabili dall'utente, e purtroppo, in alcuni casi, non dipendenti da quest'ultimo.

Dopo una introduzione alla sicurezza informatica applicata ai sistemi di autenticazione è stato presentato il progetto realizzato per questa tesi, ovvero un sistema per la raccolta di un insieme di password reali più vasto possibile. Sono stati descritti i canali dai quali è possibile ricavare chiavi d'autenticazione rilasciate sul web, e in quale modo i sistemi di acquisizione sono stati costruiti per raccoglierle. È stato descritto il database che mantiene la collezione e i problemi derivanti dalla gestione di una grande mole di dati. È stato affrontato il problema del parsing di materiale non formattato con pattern standard proveniente dal web, e si è arrivati alla conclusione di scartare tutto il materiale non conforme a pattern certi per preservare la consistenza dei dati nel database.

Successivamente sono stati presentati gli studi statistici effettuati sulla collezione. Dagli studi sono emersi diversi aspetti che mostrano come gli utenti scelgano le loro password molto spesso in modo inadeguato, probabilmente senza averne coscienza. Lo studio più interessante è stato quello delle occorrenze che ha svelato che il 43% delle chiavi della collezione è stata trovata più di una volta e che la password più comune ha raggiunto un numero notevole di occorrenze, circa 500,000.

Infine abbiamo svolto un esperimento immaginando di essere un utente di nome Alessio nato nel 1980 e di dover creare una password per l'iscrizione al servizio di posta elettronica Gmail. Tramite un procedimento incrementale siamo arrivati alla creazione di una password molto sicura, ma sempre tenendo come primo obiettivo la facilità di memorizzazione. La stringa finale ricavata è "G=ssio1980Ale?", in cui la prima lettera corrisponde alla prima lettera del nome del servizio, così facendo utilizziamo password diverse per ogni account, probabilmente l'aspetto più importante quando si parla di sicurezza dei sistemi di autenticazione.

35

Bibliografia

[1] RFC7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 4.3.3 POST. http://tools.ietf.org/html/rfc7231#section-4.3.3

[2] RFC7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 4.3.1 GET. http://tools.ietf.org/html/rfc7231#section-4.3.1

[3] D. Bellavista. ICT Security: defence strategies against targeted attacks. 2.2.2 Targeted attacks. http://amslaurea.unibo.it/6960/1/daniele_bellavista_tesi.pdf

[4] Hypertext Markup Language. RFC1866. https://tools.ietf.org/html/rfc1866

[5] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, John Langford. CAPTCHA: Using Hard AI Problems for Security http://www.captcha.net/captcha_crypt.pdf

[6] HTTP Over TLS. RFC 2818. http://tools.ietf.org/html/rfc2818

[7] Behrouz A. Forouzan. Cryptography & Network Security. 2007, Part 2, Asymmetric-Key Encipherment. McGraw-Hill.

[8] Phillip Rogaway, Thomas Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance.

[9] The MD5 Message-Digest Algorithm. RFC 1321 https://www.ietf.org/rfc/rfc1321.txt

[10] US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634. https://tools.ietf.org/html/rfc4634

[11] Xiaoyun Wang, Hongbo Yu. How to Break MD5 and Other Hash Functions. In 2005 *Advances in Cryptology - EUROCRYPT 2005*, pages 19-35, Springer.

[12] Phishing. http://en.wikipedia.org/wiki/Phishing

[13] Dictionary attack. http://en.wikipedia.org/wiki/Dictionary_attack

[14] Jhon the Ripper, openwall dictionary collection. http://www.openwall.com/wordlists/

[15] Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities. http://dl.acm.org/citation.cfm?id=2337399

[16] mysqli_real_escape_string manual reference. PHP API section 3.9.42. http://dev.mysql.com/doc/apis-php/en/apis-php-mysqli.real-escape-string.html

[17] OWASP Data validation techniques. https://www.owasp.org/index.php/Data_Validation

[18] S. Ansari, S.G. Rajeev, H.S Chandrashekar. Packet sniffing: a brief introduction. 2003 IEEE Publisher.

[19] Paul Williams. Network Security, Volume 2006, Issue 10, October 2006, Pages 13-17.

[20] Rainbow table. http://en.wikipedia.org/wiki/Rainbow_table

[21] P. Gauravaram. Security Analysis of salt-password Hashes. Published in Advanced Computer Science Applications and Technologies (ACSAT), 2012. Pages 25-30. IEEE Publisher.

[22] @PastebinLeaks. Twitter. https://twitter.com/PastebinLeaks

[23] Google alerts. https://www.google.com/alerts

[24] Ubuntu Linux. Free GNU/Linux OS. http://www.ubuntu.com/

[25] MySQL reference. 13.2.5 INSERT Syntax. http://dev.mysql.com/doc/refman/5.6/en/insert.html

[26] Uniform Resource Identifier. Wikipedia. http://en.wikipedia.org/wiki/Uniform_resource_identifier

[27] Representational state transfer. Wikipedia. http://en.wikipedia.org/wiki/Representational_state_transfer

[28] OAuth 2.0, open authorization protocol. http://oauth.net/2

[29] Tweepy. Twitter API python wrapper. https://github.com/tweepy/tweepy#tweepy-twitter-for-python

[30] Twitter REST API reference. https://dev.twitter.com/rest/public

[31] Portable Network Graphics. Wikipedia. http://en.wikipedia.org/wiki/Portable_Network_Graphics

[32] Plotly. https://plot.ly/

[33] Jinja2. Template Engine for Python. http://jinja.pocoo.org/docs/dev/

[34] Skull Security Wiki. Wordlists. https://wiki.skullsecurity.org/Passwords

[35] Cain & Abel, Password Recovery Tool. http://www.oxid.it/cain.html

[36] John The Ripper, Password Cracker. http://www.openwall.com/john/

Ringraziamenti

Vorrei ringraziare il mio relatore e ancor prima professore Gabriele D'Angelo per aver saputo trasmettere la passione, oltre che la conoscenza, di ciò che insegna e non solo, mettendo nel proprio lavoro un impegno che raramente ho trovato in un docente. Lo ringrazio anche per l'infinita pazienza e del tempo che mi ha dedicato.

Vorrei ringraziare la mia famiglia per il supporto che mi ha dato in questi tre anni.

Ringrazio i miei coinquilini Paolo, Nicola, Edoardo, Ivan, Mattia e Claudio per il tempo passato insieme.

Ringrazio gli amici che ho incontrato nel corso dei miei studi. Una dedica speciale a Giacomo e al suo umorismo.

Un ringraziamento ovviamente al gruppo CeSeNA, un ambiente perfetto dove coltivare le passioni insieme a persone veramente valide.

Ultimo, ma non per importanza, una dedica al gruppo PLOT, che riesce sempre a sollevare il morale.