HÖGSKOLAN
I HALMSTAD

IT forensik och informationssäkerhet, 180 hp

KANDIDATUPPSATS

Lösenordsmönster

Att förebygga svaga lösenord

Mark Crossley och Joakim Lindell

Datateknik 15hp

Halmstad 2015-08-25

Lösenordsmönster

Att förebygga svaga lösenord

Kandidatuppsats

2015 Juni

Författare: Mark Crossley
Författare: Joakim Lindell
Handledare: Eric Järpe
Examinator: Mattias Weckstén

Sektionen för informationsvetenskap, data- och elektroteknik
Högskolan i Halmstad
Box 823, 301 18 HALMSTAD

© Copyright Mark Crossley, Joakim Lindell, 2015. All rights reserved
Kandidatuppsats
Rapport
Sektionen för informationsvetenskap, data- och elektroteknik
Högskolan i Halmstad

Tack

Ett stort tack till handledare Eric Järpe, vars visade passion för akademia har gjort att vår resa nått en längre sträcka.

Executive summary

Passwords are used more now than ever before. Their use is based on the idea that the password is only known to the user and that its secrecy prevents others from accessing potentially valuable or sensitive information. But how secret is a password in today's high tech world?

Passwords are generally converted into hashsums and saved in databases. Cracking a password requires that the process is reversed so that the actual password can be derived from the hash sum. This cracking process can be achieved by two methods. An attacker can test all the possible combinations, (brute force cracking) or the attacker can compare the password with a list of commonly used passwords (cracking with wordlists).

This paper investigates a passwords vulnerability to both brute force cracking and cracking via wordlists. It uses a modern computer's processing speeds to establish the amount of time to crack a certain password via brute force cracking. It also deploys state of the art techniques to examine a password's content. It analyses three databases from different online communities to examine any possible correlation between a user's hobby interest and their choice of password.

This paper finds that the majority of passwords won't remain secret for very long. Short passwords which consist of a small alphabet are particularly vulnerable to brute force attacks. However due to the increasing speed of modern computers even passwords which are twelve characters long are still potentially vulnerable. This paper finds that users from a variety of online communities choose common passwords which are likely to be on a wordlist and thus susceptible to cracking via word list attacks.

This paper provides suggestions on how a user can choose a stronger password.

Sammanfattning

Lösenord används allt mer frekvent i och med digitaliseringens utspridning. Användingsområdet bygger på idéen att ett lösenord är känt endast för en användare och att denna hemlighet förhindrar andra från att komma åt värdefull eller känslig information. Men hur hemligt är ett lösenord i dagens högteknologiska värld?

Lösenord är typiskt sett beräknade till hashsummor och lagrade i databaser. Att knäcka en lösenordshash görs typiskt sett genom två metoder. Antingen genom att en angripare provar samtliga möjliga lösenord upp till och med en viss angiven teckenlängd, så kallad brute force knäckning. Det andra alternativet är genom att prioritera vissa lösenord som bedöms sannolika; en ordlistattack.

Detta arbete undersöker vissa sårbarheter hos ett lösenord gentemot både brute force knäckning och ordlistattacker. Det är begränsat till den processorkraft en genomsnittlig persondator kan tänkas inneha. Arbetet utnyttjar metoder som anses state of the art i att analysera ett lösenords uppbyggnad. Det analyserar tre databasdumpar från olika communities på internet, för att undersöka eventuella samband mellan användares intressen och deras lösenord.

Arbetet finner att majoriteten av lösenord inte kommer att vara hemliga allt för länge. Korta lösenord är särskilt sårbara för brute force knäckning. Ökningen i prestanda gör även att lösenord upp till tolv tecken kan vara obekvämt sårbara. Det visas även att det finns god anledning att göra fortsatta studier på ordlistattacker baserade runtomkring användarens intresseområden.

Avslutningsvis ges råd på procedur för att öka lösenordsstyrkan.

Ordlista

Det finns några grundläggande termer som är använts genom hela rapporten. Termer är förklarade nedan.

- Hashsumma
  Hash-funktioner som MD5 eller SHA-1 används bland annat för att försvåra att lösenord som blir stulet går att läsa i klartext. Dessa funktioner ersätter en godtycklig sträng tecken mot en ny sträng, en hashsumma, bestående av fast teckenlängd och teckenuppsättning. På så vis går det inte längre att urskilja den ursprungliga teckensträngen i klartext.

- Knäcka lösenord
  I det här arbetet är knäcka lösenord refererat till som processen att generera en hashsumma som är identisk till ett lösenords hashsumma.

- Brute force knäckning
  Brute force knäckning gissar systematiskt alla möjliga teckenkombinationer för att testa vilken kombination som knäcker hashsumman.

- Knäckning med hjälp av ordlistor
  Tekniken benämns ibland "smarta" gissningar. Den jämför lösenords hashsumman med att generera hashsummor för ord enligt de som finns i en ordlista. Ordlistan innehåller allmänt ord, namn, stavelser, siffror och kombinationer av dessa, samt erfarenhetsmässigt vanliga lösenord som "qwerty", en sträng tangenter bredvid varandra på ett tangentbord.

- Databasdumpar
  Databasdumpar som har läckt ut till allmänheten inkluderar Yahoo, Sony, LinkedIn och RockYou. Databasdumpar innehåller riktiga lösenord från verkliga användare som har läckt ut till allmänheten.

- Lösenordsrymd
  Lösenordsrymdens storlek beror på två faktorer: lösenordslängd n och mängden tecken i alfabetet A. Lösenordslängd är definierad som längden av ett lösenord, medan det med alfabet avses den storlek av alfabet som används för lösenordet. De alfabet som är tillgängliga för en användare kan typiskt sett delas upp i fyra olika delar: versaler, gemener, siffror och speci-

altecken. Lösenordsrymdens storlek blir då $|A|^n$. Lösenordrymdsrymdens storlek visar antalet möjliga lösenordskombinationer.

- Lösenordsentropi
  Lösenordsentropi är ett grundläggande koncept för att undersöka ett lösenords styrka. Med hjälp av denna bedöms ett lösenords oförutsägbarhet. Den försöker att värdesätta hur svårt ett lösenord är att knäcka. Det vill säga, hur starkt ett lösenord är.

- Födelsedagsparadoxen / Lösenordsparadoxen
  Födelsedagsparadoxen är ett klassiskt problem inom sannolikhetsläran. Den innebär att sannolikheten är större än femtio procent för att det i en grupp om 23 slumpmässigt utvalda personer finns minst två med samma födelsedag. Det här arbetet använder tankegången bakom födelsedagsparadoxen för att beräkna sannolikheten att minst två användare har samma lösenord. Det vill säga sannolikheten för en krock eller sannolikheten att en angripare lyckas gissa ett lösenord som någon användare har.

- Moores lag
  Moores lag anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastigheten inom 10 år kommer att vara 100 gånger högre än idag, se Brumen och Cernezel [1]. Således multiplicerades antalet lösenord med 100 för att beräkna antalet lösenord som knäcks inom en viss tid om tio år.

- Segmentering
  Lösenord kan vara uppbyggda på flera olika sätt. De kan innehålla ett ord, flera ord eller inget ord alls. För att kunna analysera lösenord måste lösenord bearbetares. Lösenord måste segmenteras för att identifiera ord.

- Ordnät
  Ordnät är en lexikal databas. Det kategoriserar ord i grupper av synonymer kallas synsets. Till exempel en labrador retriever och tysk schäferhund skulle ha samma synset - hund.

Innehåll

Tack iii

Executive summary v

Sammanfattning vi

Ordlista vii

1 Introduktion 1
- 1.1 Bakgrund . . . 1
- 1.2 Lösenordshantering . . . 2
- 1.3 Lösenordsstyrka . . . 3
- 1.4 Lösenordsanalys . . . 5
  - 1.4.1 Semantik . . . 5
- 1.5 Bidrag till området . . . 5

2 Problemställning 7

3 Metod 9
- 3.1 Lösenordsentropi och lösenordsrymd . . . 9
- 3.2 Segmentering . . . 10
- 3.3 Klassificering . . . 11
- 3.4 Semantisk generalisering och kategorisering . . . 11
- 3.5 Lösenordsparadoxen . . . 11
- 3.6 Problematisering av metoden . . . 13

4 Experiment 15
- 4.1 Lösenordsrymd . . . 15
- 4.2 Lösenordsparadoxen . . . 16
- 4.3 Analys . . . 17
- 4.4 Testdata . . . 17

5 Resultat 19
- 5.1 Lösenordsrymd . . . 19
- 5.2 Lösenordsparadoxen . . . 27
- 5.3 Semantik . . . 32

5.4 Procedur för att öka lösenordsstyrkan 34

6 Diskussion **37**

6.1 Lösenordsrymd 37

6.2 Lösenordsparadoxen 40

6.3 Semantiska mönster i intresseindelade databasdumpar 41

7 Slutsats **43**

7.1 Framtida arbete 44

7.1.1 Lösenordsparadoxen 44

7.1.2 Semantiska mönster 44

8 Referenser **45**

Tabeller

1 Alfabeten som ett lösenord kan utnyttja 9
2 Lösenordsrymdens storlek som ska undersökas för en krock . . . 12
3 Antalet lösenord knäckta per sekund, minut, dag och år 16
4 Hur lösenordslängd påverkar antalet möjliga lösenordkombinationer 20
5 Hur alfabetsstorlek påverkar antalet möjliga lösenordkombinationer 22
6 Tiden som krävs för att beräkna alla hashar för en lösenordsrymd 25
7 Antalet tecken som krävs för att skapa ett lösenord som tar i genomsnitt 31 dagar att knäcka med brute force 26
8 Sannolikheten för en krock 28
9 20 mest frekvent förekommande lösenord i testdata. 32
10 20 mest frekvent förekommande segment av lösenord i testdata. . 33
11 20 mest frekvent förekommande semantiska generaliseringar av lösenord i testdata. 34

Diagram

1 Hur lösenordslängd påverkar antalet lösenordkombinationer . . . 21
2 Hur alfabetsstorlek påverkar antalet möjliga lösenord 23
3 Jämförelse av hur en ökning i lösenordslängd eller alfabet påverkar antalet möjliga lösenord. 24
4 Teckenuppsättningen för ett lösenord som tar i genomsnitt 31 dagar att knäcka med brute force 27
5 Sannolikheten för en krock 29
6 Sannolikheten för en krock med alfabet som innehåller siffror . . 30
7 Sannolikheten för en krock med alfabetet innehållande versaler, gemener och siffror 31
8 Sannolikheten för en krock med alfabet innehållande versaler, gemener, siffror och specialtecken. 31

1 Introduktion

Nio av tio användarvalda lösenord går att hitta i en vanlig ordbok. I detta arbete utforskas farorna med samt tillvägagångssätt, för hur ett lösenordsval kan undvika fallgropen att vara precis som alla andra. Denna första sektion introducerar förutsättningar och bakgrundsinformation runtomkring arbetet.

1.1 Bakgrund

Användandet av lösenord är väl utbrett. Det är en tid där ett allt bredare urval av person- och samhällsspecifika tjänster är tillgängliga elektroniskt. För att upprätthålla integritet omkring dessa privata användingsområden, krävs en process för att kontrollera en användares identitet innan tillgång ges till en specifik tjänst. Den här autentiseringen fungerar ofta med hjälp av en av tre grundläggande principer; "någonting du är", "någonting du innehar" eller "någonting du känner till". Biometrisk autentisering utnyttjar principen "någonting du är" och bygger på igenkänning av individers karaktäristiska fysiologiska egenskaper för att verifiera och identifiera användarens identitet. Smartkort eller tvåstegsverifiering beror på principen "någonting du innehar" och kräver att användare har med någonting för att verifiera sin identitet som till exempel id-kort eller mobiltelefon. Autentisering med lösenord beror på principen "någonting du känner till". Det bygger på idén att användare känner till information som är hemlig för obehöriga. Autentisering av lösenord är nu och har alltid varit ett av de mest vanligt förekommande sätten för autentisering se Loch et al [13], Tzong-Chen och Hung-Sung [24], Zviran och Haga [27], Lee et al [12], Egelman et al [6] och Creese et al [5]. Det beror på att metoden är billig och lätt att implementera. Utöver detta underlättar det att konceptet är förhållandevis enkelt för användaren att förstå. I motsats till biometrisk- eller tvåstegsverifiering krävs det heller ingen kringutrustning.

Lösenord har emellertid ett antal erkända brister. En av dessa är att lösenord inte alltid nödvändigtvis är hemliga för en obehörig. Morris och Thompson [17] analyserade 3289 lösenord i 1979 och drog slutsatsen att 86% var svaga eftersom de var för korta, innehöll vanliga ord eller endast innefattade gemener. Forskning påvisar att lösenord fortfarande lider av samma brister idag. Florencio och Herley [8] studerade cirka fem miljoner lösenord och observerade att de flesta endast innehöll gemener eller siffror. Dessa observationer har fått stöd av flera

forskare se Notoatmodjo och Gilbert [18], Sasse et al [20], Tupsamudre et al [23] och Tam et al [22]. De här preferenserna producerar svaga lösenord som lätt kan knäckas, vilket i sin tur resultera i att lösenord kanske inte längre är hemligt.

1.2 Lösenordshantering

Datoradministratörers uppgift innefattar att hantera lösenord på ett säkert sätt. Kryptografiska hash-funktioner som MD5 eller SHA-1 används för att försvåra att lösenord blir stulna. Dessa funktioner ersätter varje enskilt tecken med flera andra, så att ordet inte längre består av klartext. På det här sättet svarar varje lösenord i klartext mot en viss hashsumma. Den sparas sedan i en lösenordsdatabas. Då någon autenticierar med lösenord beräknas automaktiskt hashsumman och jämgörs med innehållet i lösenordsdatabasen. Med detta förfarande försvåras för en obehörig att läsa lösenord om databasen skulle bli stulen. En nackdel med hash-funktioner för detta användingsområde är att de alltid producerar samma hashsumma för en viss teckenföjld. Lösenordet "password" skulle till exempel producera hashsumman

5f4dcc3b5aa765d61d8327deb882cf99.

För att skydda lösenordet ytterligare använder vissa datoradministratörer något som kallas salt. Att "salta" ett lösenord innebär att några slumpade tecken läggs till före och/eller efter lösenordet innan en hash beräknas med en hashfunktion.

Det finns två huvudsakliga metoder för att "knäcka lösenord". Brute force knäckning och knäckning med hjälp av ordlistor. Båda teknikerna producerar hashsummor av en viss teckenuppsättning och jämför hashsumman med hashsumman i en lösenordsdatabas. Brute force knäckning gissar systematiskt alla möjliga teckenkombinationer för att testa vilken kombination som knäcker hashsumman. En brute force knäckning är mer effektiv mot korta lösenord som innehåller ett alfabet (se Avsnitt 3.1 Lösenordsentropi och lösenordsrymd). Ett långt lösenord kan ta lång tid att knäcka med brute force knäckning eftersom det finns flera möjliga teckenkombinationer. Därför brukar en ordlista användas, så att datorn gör "smarta" gissningar. Ordlistan innehåller allmänt ord, namn, stavelser, siffror och kombinationer av dessa, samt erfarenhetsmässigt vanliga lösenord som "qwerty". En lösenordsdatabas som är skyddad med salt kräver att angripare generar lösenordshashen för varje ord med hjälp av det använda saltet. Att pröva miljarder lösenord på det sättet är ansträngande, men några

miljoner är ingen svårighet. Lösenorden bör inte vara bland de första miljonerna angriparen testar. Det betyder att lösenord inte skall vara enkla varianter på ord eller fraser som är populära.

1.3 Lösenordsstyrka

Ett koncept inom studier av lösenord är lösenordsrymd. Lösenordsrymden beror på två faktorer, lösenordslängd n och mängden tecken i alfabetet A. Lösenordsrymdens storlek blir då

$$|A|^n$$

Alfabetet är de tecken som används i lösenord. Om lösenord mellan m och n tecken (dvs lösenord vars längd är minst m bitar och högst n bitar) tillåts så blir lösenordsrymdens storlek

$$\sum_{k=m}^{n} |A|^k = \frac{|A|^m(1-|A|^{n-m+1})}{1-|A|}$$

Om en användare väljer ett lösenord som innehåller bara siffror är urvalet begränsat till 10 tecken (siffror 0-9). Ett lösenord som inkluderar siffror, symboler, små och stora bokstäver har ett betydligt större alfabet, bestående av 101 tecken. Lösenordsrymdens storlek ökar således exponentiellt i förhållande till hur långa lösenord som används och därför har ett långt lösenord betydligt fler möjligheter än ett kort lösenord. Det mest populära lösenordet i RockYou databasen var 123456, se Tupsamudre et al [23]. Det här lösenordet har en lösenordsrymd av $10^6 = 1000000$, dvs det finns 1000000 möjliga lösenord som kan skapas med 6 siffror. Den rapporterade hastigheten för att bearbeta ett klartextlösenord från ett (osaltat) hashvärde är mellan 10^9 och 10^{12} kombinationer per sekund, se Brumen och Cernezel [1]. Det mest populära lösenordet från Rock you databasen skulle därför knäckas inom 13 minuter med brute force knäckning. Det kan till och med knäckas ännu snabbare om lösenord finns med i ordlistan. Moores lag [16] anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastighet om 10 år kommer att vara 100 gånger snabbare än idag. Detta utgår enbart från Moores lag och tar inte hänsyn till möjliga förbättringar inom områden som till exempel beräkningar av hashsummor. Det betyder att det kommer att bli alltmer viktigt att ha ett lösenord som inte går att knäcka inom den första miljarden gissningar.

Datorer kan också generera lösenord, vilket ofta används vid e-handel. Microsoft generar exempelvis ett lösenord som är 25 tecken långt och innehåller tecken från varje definerat alfabet, så som versaler, gemener siffror och specialtecken. Det innebär en Lösenordsrymd med 101^{25}
= 128 243 199 501 723 361 359 275 202 020 838 997 632 567 303 002 501 olika möjliga kombinationer. För att vara bra ska lösenord helst inte innehålla några riktiga ord, så det är osannolik att lösenordet skulle finnas på någon ordlista. Ett sådant lösenord kan bara knäckas med brute force knäckning och skulle ta betydligt längre tid att knäcka än lösenordet 123456.

Nuförtiden värderar många webbsidor användarens lösenord med en styrkemätare enligt mått så som "weak", "okay" eller "strong". Bedömningen är framförallt baserad på lösenordets längd men några webbsidor kollas även om lösenord innehåller en siffra och eller en stor bokstav. Till exempel Iloveyou bedöms som ett starkt lösenord av många webbsidor medan 1loveYou skulle bedömas som ett starkt lösenord av ännu fler webbsidor. Båda metoderna ignorerar dock hur majoriteten av lösenord knäcks. Det finns mjukvaror som är specialiserade på att knäcka lösenord, se Veras et al [25]. Mjukvaror använder ordlistor som innehåller populära lösenord, (t.ex. iloveyou), och ordens genererade hashsummor. Mjukvaror jämför lösenordets hashsumma med alla hashsummor på listan. Mjukvara kan till och med "mangla" lösenord. Det vill säga mjukvaran förändrar lösenord med en viss kombination av nummer, symbol och stora bokstäver. Mjukvaror skulle lätt knäcka 1loveYou eftersom iloveyou är ett populärt lösenord och finns på ordlistan. Ett bra bedömningsverktyg skulle därför kolla på lösenordets innehåll och inte bara längd och teckenuppsättning. En beräkning av lösenordsrymd är inte tillräcklig för att bedöma ett lösenord. En bra lösenordsstyrkemätare borde ta hänsyn till lösenordets innehöll (ord, mönster och så vidare), se Zhang-Kennedy et al[26]. Den borde också ta hänsyn till det faktum att människor måste komma ihåg lösenord, se Florencio och Herley [9]. Det vill säga Tr0ub4dor&3 är ett starkt lösenord, men det är möjligen krångligare att komma ihåg. En bra styrkemätare borde inte bestraffa ett lösenord som correcthorsebatterystaple, som medan det bara innehåller ett alfabet, är tillräckligt långt och består av flera ord för att höja entropin.

1.4 Lösenordsanalys

Datorn kan hantera allt större datamängder på grund av ökningen i omfattning av datalagring och kraftfullare datorer. Det finns stora mängder av lösenord som har läckt ut i form av databasdumpar från tjänster som Yahoo, Sony, LinkedIn och RockYou, se Veras et al [25]. Mängden lösenord är så stor att den kan klassificeras som Big Data, se Mcafee och Brynjolfsson [15]. Jakobsson och Dhiman[11] påpekar att det är människor som oftast väljer lösenord och Marshalls [14] forskning visade att människors hobbyintresse påverkar valen. Marshalls kartläggning visar att 11% av de undersökta använder fotbollslag som lösenord. Marshalls arbete baserade sig på 125 enkäter. Databasdumpar som har läckt ut till allmänheten gör det möjligt att analysera betydligt större dataset. Analysering av databaser har ytterligare en fördel jämfört med kartläggningar som grundas på enkäter. Det är osäkert om människor svarar sanningsenligt när de besvarar en enkät. Datasets innehåller dock riktiga lösenord som har läckt ut till allmänheten. Det skulle vara intressant att försöka hitta mönster, samband och trender som stöder Marshalls observationer att människors hobbyintresse påverkar lösenord i analysen av datasets.

1.4.1 Semantik

Flertalet spännande riktningar vad det gäller förståelse av lösenord har utforskats under 2014. Veras et al [25] har visat att samma sorts språkanalys som till exempel används av mobiltelefoner för att gissa nästa ord användaren vill skriva, också kan appliceras för att analysera lösenordstrender. Detta visar vilka sorters lösenord som är populära, exempelvis, är det verb eller substantiv, är det namn på städer eller människor?

1.5 Bidrag till området

Detta arbete börjar med att beräkna de matematiska begränsningarna i olika lösenordsval. Det visar hur detta påverkar lösenordsrymden. Arbetet beräknar även sannolikheten för att två lösenord är identiska, givet en likartat utspridd lösenordsrymd. Vidare redogörs för teori omkring lösenordsentropi.

Slutligen utforskas data tillgänglig ifrån publicerade databasläckor. Det ger en förhållandevis kvantitativ inblick i användarvalda lösenord. Arbetet testar en tes i att kombinera Marshall's [14] resultat om att användare väljer någonting

personligt, tillsammans med Veras' et al [25] arbete för automatiserad textanalys av lösenord. Detta genom att undersöka eventuella trender i tre databasdumpar från applikationer med olika intresseområden.

Den här kunskapen och förståelsen är viktig inom IT-säkerhet för att klura ut lösenord och för att förstärka lösenord. Förmågan att knäcka ett lösenord kan vara viktig i ett fall, eftersom det innebär tillgång till information, vilket kan innebära mer bevismaterial, se Rogers och Seigfried[19]. Utöver detta så kan förståelse för hur människor skapar lösenord också vara till hjälp vid bedömning av lösenordssäkerhet. Detta ger underlag till en bättre lösenordpolicy, se Jakobsson och Dhiman[11].

2 Problemställning

Forskning har visat att ju längre ett lösenord är desto längre tid krävs för att knäcka lösenordet, se Morris och Thomson [17], Tam et al [22]och Tupsamudre et al [23]. Moderna datorer bearbetar dock enorma antal kombinationer varje sekund, se Brumen och Cernezel [1]. Detta arbete undersöker relationen mellan olika antal tecken och alfabetsuppsättningar och hur de påverkar den tid som krävs för att knäcka ett lösenord idag genom metoden brute force. Det vill säga: hur påverkar lösenordsrymdens storlek den tid det krävs för att knäcka ett lösenord med brute force knäckning?

Svaret på den här frågan kan ha betydelse inom lösenordssäkerhet. Det kan ge vägledning för hur företag eller enskilda personer kan skapa starka lösenord. Svaret kan även visa på ett tydligt sätt hur lösenordsrymden påverkar den tid som krävs för att knäcka ett lösenord. Ett företag till exempel kan ha en lösenordspolicy som kräver att alla lösenord tar genomsnitt 30 dagar eller mer att knäcka med brute force knäckning. Företaget kan då använda svaret till den här frågan för att stipulera den minsta lösenordsrymdens storlek som uppnår säkerhetsbehov.

Med åtanke till att enligt Veras et al [25] är upp till 90% av lösenord är mellan 8 och 12 tecken långa, hur stor är risken att minst två användare råkar välja samma lösenord, givet en teoretisk förutsättning att chansen i lösenordsvalet är lika stor för alla lösenord i lösenordsrymden.

En ordlisteattack använder textsträngar ur olika källmaterial. Exempelvis kan detta vara ur en ordbok, eller ur en ordlista genererad utifrån tidigare lösenordsdumpar. En idé tidigare har varit att sortera dessa på frekvens utifrån mest använda lösenord. En tanke är, om semantiska kategorier såsom att 'fotboll' generellt kan tillhöra kategorien 'sport', om dessa genereras för en lösenordsdump, hur ser fördelningen ut i sådant fall?

Att genomföra en brute force attack kan ta lång tid i de fall stora teckenluppsättningar ska täckas upp. Att ha ytterligare metoder för att uppnå effektiviserade ordlisteattacker skulle kunna erbjuda större attackvinklar att ta hänsyn till. Att hitta antydan av mönster i semantiska kategorier skulle kunna vara en början till att konkretisera en sådan attackvinkel.

En tanke är att det går att uppskatta en beteendemodell för hur användare väljer lösenord. Detta skall utforskas genom att besvara följande frågor.

1. Hur påverkar lösenordsrymdens storlek den tid det krävs för att knäcka ett lösenord med brute force knäckning?

2. Hur kan lösenordslängd och antal tecken i det alfabetet som lösenordet utnyttjar påverka sannolikheten för att minst två lösenord valda med jämn utspridning råkar vara samma?

3. Hur är vissa semantiska kategorier av lösenord fördelade i frekvens i applikationer med ett visst intresseområde?

3 Metod

Denna sektion ger en bild av hur problemställningen ska undersökas. Fokus ligger på generella angreppssätt, och inte specifik mjukvara eller programkod.

3.1 Lösenordsentropi och lösenordsrymd

Lösenordsentropi är ett grundläggande koncept för att undersöka ett lösenords styrka. Med hjälp av denna bedöms ett lösenords oförutsägbarhet. Den försöker att värdesätta hur svårt ett lösenord är att knäcka. Det vill säga, hur starkt ett lösenord är. Den utnyttjar lösenordsrymd och gör en beräkning för ett lösenords stryka som är representerad i bits. Lösenordsrymdens storlek beror på två faktorer: lösenordslängd och alfabet. Lösenordslängd är definierad som längden av ett lösenord, n, medan det med alfabet avses den storlek av alfabet som används för lösenordet. De alfabet som är tillgängliga för en användare kan typiskt sett delas upp i fyra olika delar, se Tabell 1.

Tabell 1 visar alfabet som ett lösenord kan utnyttja.

Alfabets-uppsättning	Beskrivning	Antalet tecken
G	Alfabet med gemena bokstäver a–ö	29
V	Alfabet med versala bokstäver A–Ö	29
S	Alfabet med siffror 0–9	10
Sp	Alfabet med specialtecken som ;?£	33

Tabell 1: Alfabeten som ett lösenord kan utnyttja. Tabellen visar alfabetet som är typiskt tillgängliga för en användare och antalet möjliga tecken i varje alfabetsuppsättning. Där G representerar gemener, V representerar versaler, S står för siffror, och Sp står för specialtecken

Lösenordsrymd är summan från varje alfabet. Den potentiella lösenordsrymden är beräknad med formlen nedanför där L står för lösenordsrymd och n står för lösenordslängd.

$$L = (|G| + |V| + |S| + |Sp|)^n \tag{1}$$

Till exempel ett lösenord med 8 tecken har lösenordsrymd $(29+29+10+33)^8 = 101^8 = 10828567056280800$ bitar.

3.2 Segmentering

Lösenord kan vara uppbyggda på flera olika sätt. De kan innehålla ett ord, flera ord eller inga ord alls. För att kunna analysera lösenord måste de bearbetas. Detta då ett lösenord måste segmenteras för att identifiera de ord lösenordet innehåller. Metoden som används är baserad på Veras et al [25], vars metodik anses state-of-the-art, se de Carnavalet et al [2].

Låt oss bryta ner denna metod i steg.

Segmenteringsfasen är inspirerad av Jakobsson och Dhimanoch Dhiman's arbete [11], som har fokuserat på att segmentera just lösenord. Denna metod använder en kombination av ordböcker och korpus (en samling med väldigt många olika meningar som definerar språkbruk för en algoritm). Med hjälp av två källor för att segmentera lösenord är det dels möjligt att använda ordböcker till att hitta de ord som är rättstavade och lämpliga att utföra automatisk analys på. Eventuellt överblivna tecken klassificeras som "övrigt" (luckor). På dessa utförs endast analys av exempelvis teckenuppsättning. Till exempel skulle ett lösenord som påskharen2015^, kunna brytas upp i exempelvis både

Luckor {2015, ^} Ord {påsk, har, en}

och

Luckor {2015, ^} Ord {påsk, haren}

På så vis används ett korpus eller ordlista. Genom en N-gram analys, se Veras et al [25], kan sedan påvisas att det är större chans att lösenordet avser ordet 'haren' i relation till det tidigare ordet, 'påsk', eftersom dessa sannolikt oftare finns i samma mening.

Ingen av dessa metoder tar hänsyn till ord som inte finns i ordböcker, till exempel egennamn eller artister. Därför behöver det även genereras en uppsättning egennamn, ordlistor för artister och så vidare. Metoden baseras på den algoritm som föreslås av Veras et al som skapade ordlistor för att komplettera ordböcker.

Denna metod bör betraktas som känslig mot det faktum att ingenting hindrar lösenord från att avvika från det språk som är dominant i datasetets geografiska härkomst. Alltså, även om ett datasetet är från Sverige, så kan ett lösenord vara skrivet på ett annat språk. Det kan leda till att korta ord på ett annat språk får en hög false-positive, och därmed segmenteras inkorrekt. Ett false-positive

innebär att ett kort ord, med exempelvis två tecken, kan råka förekomma i en slumpmässig sträng av tecken, och leda till ett felaktigt identifierat ord där det egentligen inte finns något.

3.3 Klassificering

Innan ett ord kan kategoriseras så finns det nytta i att göra en bedömning av ordklass. Genom att matcha ordningsföljden på segment gentemot uppsättningar ordföljder genererade från litteratur kan en uppskattning av sannolikheten för ordklass göras.

3.4 Semantisk generalisering och kategorisering

Utifrån ett ordnät klassifieras segment. Detta genom att slå upp namnet på den samling synonymer som ordet tillhör. Då det inte finns lika kompletta ordnät för adjektiv som för substantiv och verb, så begränsas arbetet till dessa. Det är samma begränsning som tidigare arbeten gjort, exempelvis Veras et al [25].

3.5 Lösenordsparadoxen

Födelsedagsparadoxen är ett klassiskt problem inom sannolikhetsläran. Den innebär att sannolikheten är större än femtio procent för att det i en grupp om 23 slumpmässigt utvalda personer finns minst två med samma födelsedag, se Suzuki et al[21].

Med formeln i (2) beräknas sannolikheten att åtminstone två människor har samma födelsedag.

$$P(A(n), B(d)) = 1 - (\frac{d(d-1)....(d-n+1)}{d^n}) \approx 1 - \left(\frac{d-1}{d}\right)^{n(n-1)/2} \quad (2)$$

där P = sannolikheten, $A(n) = \{$det är n personer i gruppen$\}$ och $B(d) = \{$det är d dagar på "året"$\}$

Det här arbetet använder tankegången bakom födelsedagsparadoxen för att beräkna sannolikheten att två användare har samma lösenord, det vill säga sannolikheten för en krock. Antalet dagar ersätts därför med lösenordsrymd för att producera en ekvation (3).

$$P(A(n), B(l)) \approx 1 - \left(\frac{l-1}{l}\right)^{n(n-1)/2} \quad (3)$$

där P = sannolikheten, $A(n)$ = {det är n lösenord i lösenordsdatabasen} och $B(l)$ = {lösenordsrymdens storlek är l}

Lösenordsrymdens storlek beräknas med hjälp ekvation (1).

Arbetet ska beräkna krocksannolikheten för de lösenordsrymdskombinationer som finns i Tabell 2.

Lösenordslängd	Alfabet	Lösenordsrymd
6	S	10^6
6	V,G,S	68^6
6	V,G,S,Sp	101^6
8	S	10^8
8	V,G,S	68^8
8	V,G,S,Sp	101^8
10	S	10^{10}
10	V,G,S	68^{10}
10	V,G,S,Sp	101^{10}

Tabell 2: Lösenordsrymdens storlek som ska undersökas för en krock. Det vill säga sannolikheten att minst två användare har samma lösenord, baserad på lösenordslängd och lösenordsalfabet.

De olika lösenordslängder och alfabet som undersöks ger en graf som visar hur de olika faktorerna påverkar sannolikheten att en krock uppstår. Lösenordslängd representerar en spridning som omfattar den övervägande majoriteten av lösenord, till exempel över 98% av lösenord inom RockYou, se Chou et al[4]. Alfabetets kombinationer versaler, gemener, siffror och specialtecken (V,G,S,Sp) representerar den maximala potentialen medan alfabetet med enbart siffror representerar det minsta alfabet. Analysen av databasen RockYou visade dock att mindre än 4% av lösenord innehöll specialtecken, medan 16% använder bara siffror. Kombinationen av alfabeten versaler, gemener, och siffror omfattade ca 98%, se Tupsamudre et al [23].

Bortsett från vid de lägsta bitlängderna, så passerar lösenordsrymden den storlek som datatyper i moderna programmeringsspråk klarar av att hantera. Därför finns det ett behov av att använda bibliotek som utökar eller arbetar runt eventuella begränsningar i datatyper.

Den här analysen ska visa potentialen för olika lösenordsrymder. Den ska visa hur många kombinationer det finns för ett lösenord med hänsyn till längd och alfabet. Lösenordsrymden har en svaghet för applikationer med användarvalda lösenord. Det tar inte hänsyn till att människor väljer efter mönster, det vill säga ord eller en viss kombination av bokstäver som qwerty eller "repeats" som aaaaaa, samt även sekvenser som abcdef. Utifrån lösenordsrymdens storlek sett till samma teckenlängd och samma antal tecken i alfabetet kan bedöma iloveyou och zpnfbsnw som lika starka (samma entropi). Undersökningar har dock visat att iloveyou är mer sannolikt än zpnfbsnw, se Veras et al [25]. Lösenordsrymdens storlek är användbar för att visa den högsta potential som finns hos ett lösenord. Det krävs dock en skickligare styrkemätare för att bedöma ett lösenords styrka.

3.6 Problematisering av metoden

De olika lösenordslängder och alfabet som undersöks ger en bred representation av de olika lösenord som användare väljer idag. De lösenordslängder, mellan sex och 10 tecken, som undersökts representerar en spridning som omfattar en övervägande majoritet av lösenord. Spridningen motsvarar till exempel över 98% av lösenord inom RockYou, se Chou et al [4]. Undersökningen fokuserar på tre olika alfabetsuppsättningar, med olika omfattning. Det första alfabetet består av en kombination av versaler, gemener, siffror och specialtecken (V,G,S,Sp) och representerar den maximala potentialen. Alfabet nummer två består av enbart siffror och representerar det minsta alfabet. Det tredje valet av alfabet består av versaler, gemener och siffror, vilket omfattade ca 98% av lösenorden inom RockYou, se Tupsamudre et al [23]. Analysen av databasen RockYou visade även att mindre än 4% av lösenord innehöll specialtecken, medan 16% använder bara siffror.

Detta arbete ska undersöka de möjliga kombinationer som finns med fyra olika alfabetsuppsättning och olika lösenordlängd. Det vill säga hur många olika lösenord kan skapas med en viss längd och alfabetsuppsättning. Ett möjligt alternativ till fördjupning hade varit att använda Markovkedja. Markovkedja är en statistisk modell som gör antaganden om vilka tecken som har störst sannolikhet av att följa en viss bokstav eller ord. Till exempel har bokstaven q en stor sannolikhet att följas av u på engelska. På grund av tidspress beslutades att arbetet skulle fokusera på olika teckenuppsättning och inte göra en fördjupning inom Markovkedja.

Att testa födelsedagsparadoxen på lösenord har vissa brister. Dels att korrekt standardprocedur för säkerhet använder sig av både användarspecifik och applikationstäckande salt i hashsumman för lagrade lösenord, vilket innebär att det inte är urskiljbart om två eller fler användare har samma lösenord. Då måste fortfarande var användare testas specifikt. Konceptet har dock viss relevans vid automatiskt genererade lösenord. En framtida idé hade varit att istället applicera födelesdagsparadoxen på hashsummor.

Segmentering av lösenord är en viktig del av arbetet. Det är viktigt att kunna segmentera ett lösenord för att identifiera om lösenordet innehåller ett ord, flera ord eller inga ord alls. Arbetet kan då analysera ord och undersöka om det finns en koppling mellan lösenordsval och hobbyintresse. Metoden som används är baserad på Veras et al [25], vars metodik anses state-of-the-art, se de Carnavalet et al [2]. Dock inriktar sig arbetet endast på engelskspråkiga ord, i och med att testdata är på engelska. Dock kan det fortfarande finnas lösenord som är utformade enligt andra språk, och då fungerar inte segmenteringen korrekt. Om detta vore överrepresenterat, så skulle det dock märkas i resultatet.

Lösenordspolicies vid skapande av användarlösenord kan skapa variationer i lösenord. Detta arbete tar inte höjd för att utforska eventuella skillnader mellan semantiska kategorier härstammande från testdata med olika lösenordpolicies. För att ha en så jämn jämförelse som möjligt används dock testdata utformade enligt samma lösenordspolicy.

4 Experiment

Denna sektion ger fördjupad experimentspecifik information runtomkring hur problemuppställningen angrips med metoden. Den mjukvara som använts för att utföra delsteg i metoden specificeras.

4.1 Lösenordsrymd

Lösenordsrymdens storlek visar antalet möjliga lösenordskombinationer med hänsyn till lösenordets längd och alfabet. Lösenordsrymdens storlek är beräknad enligt ekvation (1). Lösenordsrymd var beräknat för alla lösenordslängder och alfabet i Tabell 2. Den beräknade lösenordsrymden visas i Tabell 4.

En modern dator bearbetar 10^9 lösenord per sekund, se Brumen och Cernezel [1]. Beräkningen kommer från ett experiment som utfördes på en Windows 7 desktop dator. Datorn hade en AMD Radeon HD 7970 graphical processing unit (GPU) installerad och körde lösenordsknäckning med mjukvara Hashcat för att bearbeta hashade lösenord.

Tabell 3 visar antalet lösenord som knäcks inom en viss period baserat på en dator med samma specifikation som citerades av Brumen och Cernezel. 1 miljard lösenord knäcks efter en sekund så efter en minut är 60* 1 miljard = 60 miljarder lösenord knäckta. En liknande beräkning utfördes för att beräkna antalet lösenord som knäcks inom en timme, en dag och ett år. Moores lag anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastigheten inom 10 år kommer att vara 100 gånger högre än idag, se Brumen och Cernezel [1]. Således multiplicerades antalet lösenord med 100 för att beräkna antalet lösenord som knäcks inom en viss tid om tio år.

Antalet lösenord som knäcks per sekund, minut, timme, dag och år finns i Tabell 3.

Tid	Antalet lösenord	Antalet lösenord om 10 år (enligt moores lag)
1 sekund	1000000000	100000000000
1 minut	60000000000	6000000000000
1 timme	3600000000000	360000000000000
1 dag	86400000000000	8640000000000000
1 år	31536000000000000	3153600000000000000

Tabell 3: Antalet lösenord knäcks per sekund, minut, dag och år. Tabellen visar beräkningen för idag och om tio år. Beräkningen är baserad på ett experiment som utfördes på en Windows 7 desktop dator som hade en AMD Radeon HD 7970 graphical processing unit (GPU) installerad och körde lösenordsknäckning med mjukvara hashcat för att bearbeta lösenord. Moores lag anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastigheten inom 10 år kommer att vara 100 gånger högre än idag, se Brumen och Cernezel [1]. Således multiplicerades antalet lösenord med 100 för att beräkna antalet lösenord som knäcks inom en viss tid om tio år.

I arbetet beräknas den förväntade tid som krävs för att knäcka vissa lösenord (se Tabell 7 för listan). Ekvationen nedan beräknar tiden som krävs för att knäcka alla möjliga lösenordskombinationer med hänsyn till lösenordslängd och alfabetet (det vill säga lösenordsrymdens storlek).

$$T = \frac{L}{A}$$

T=Tiden som krävs uttryck i någon tidsenhet L=Lösenordsrymd A=Antalet lösenord som i genomsnitt knäcks inom tidsperioden.

För att uppskatta en lämplig tidsenhet har lösenordsrymden storlek först dividerats med det genomsnittliga antalet lösenord som knäcks på ett år. Om svaret var mindre än ett, dividerades lösenordsrymden storlek med det genomsnittliga antalet lösenord som knäcks varje dag. Om svaret fortfarande var mindre än ett dividerades den med det genomsnittliga antalet lösenord som knäcks varje timme, och så vidare fram till svaret gavs.

4.2 Lösenordsparadoxen

För att utföra beräkningar hos teckenlängder om nio och tio tecken, så räcker inte datatyper i majoriteten av moderna programmeringsspråk till. I och med detta används biblioteket GMP - GNU Multiple Precision Arithmetic Library,

se Granlund [10]. Det tillåter teoretiskt sett hantering av godtyckligts stora tal med endast internminnet i datorsystemet som begränsning. I sin tur har GMP ett från problemställningen i detta arbete avvikande användningssyfte. Det är i första hand avsett för beräkningar av väldigt stora heltal. I detta arbete finns ett intresse av att beräkna sannolikeheter.

Beräkningarna är utförda under en 64-bitars arkitektur, med i7-3930k som CPU.

4.3 Analys

I Avsnitt 3 Metod beskrevs tillvägagångssättet för att göra en semantisk analys på databasdumpar. De tre stegen som förklarades var segmentering, klassificering och kategorisering.

För att utföra segmentering används en algoritm publicerad av Veras et al, som syftar till att de segment som uppdelas ska ha en så lång totallängd som möjligt. För att genomföra en klassificering, och avgöra om ett ord är ett verb eller ett substantiv, så används verktyget NLTK 3.0.2. För att genomföra en generaliserad kategorisering av vilken ordsamling ett segment tillhör, så används WordNet 3.0.

Mjukvaran har installerats som paket ifrån Ubuntu 12.04.5's standardkatalog.

4.4 Testdata

Tre stycken engelskspråkiga databasdumpar används för att utforska de frågeställningar som beaktas. Samtliga dataset är av liknande omfattning, omkring tiotusen användarlösenord. En fördel i omfattningen hos dessa dataset är att de är av en storlek som hamnar inom ramarna för detta arbete.

Var en av dessa härstammar från internetapplikationer med ett specifikt användingsområde. Den första av dessa, Spel-databasen, kommer från ett diskussionsforum för tv- och dataspel. Den andra är ett diskussionsforum för hälsa och motion. Den tredje, och sista, är från en bokaffär med inriktning på militärhistoria.

Det innebär att eventuella avvikelser i relation till mängden information ger en godtagbar chans att upptäckas. Den tid det tar för data att bearbetas hålls också ner. Ett motargument vore att i en större datamängd, så hade eventuella

avvikelser bleknat i relation till datamängdens omfattning.

Säkerhetspolicyn för dessa dataset är av liknande karaktär: Minst tre tecken, inga krav på specialtecken, versaler eller gemener. En fördel med detta är att användare ges frihet att använda lösenord de tycker passar. Det har gjorts studier som visar att en strikt lösenordspolicy inte nödvändigtvis höjer säkerheten hos autentiseringen. Eventuellt ligger en av nackdelarna dock inte nödvändigtvis i valet, utan begränsningens omfattning. Detta då en användares val av lösenord vid mer strikt lösenordspolicy inte nödvändigtvis täcks upp av denna testdata.

5 Resultat

Nedan återfinns resultat och utdata från de olika experiment som genomförts för att utforska problemställningen.

5.1 Lösenordsrymd

Tabell 4 visar hur lösenordslängdens storlek påverkar lösenordsrymd, det vill säga antalet möjliga lösenord. Antalet möjliga lösenord blir snart enormt. Antalet blir så stort att det blir svårt att presentera på ett tydligt sätt. I Tabell 4 beskrivs därför inte enbart antalet möjliga lösenord. Tabellen visar även hur många siffror som respektive lösenordsrymdens storlek består av, dvs 10-logaritmen av lösenordsrymdens storlek avrundat till närmaste mindre heltal:

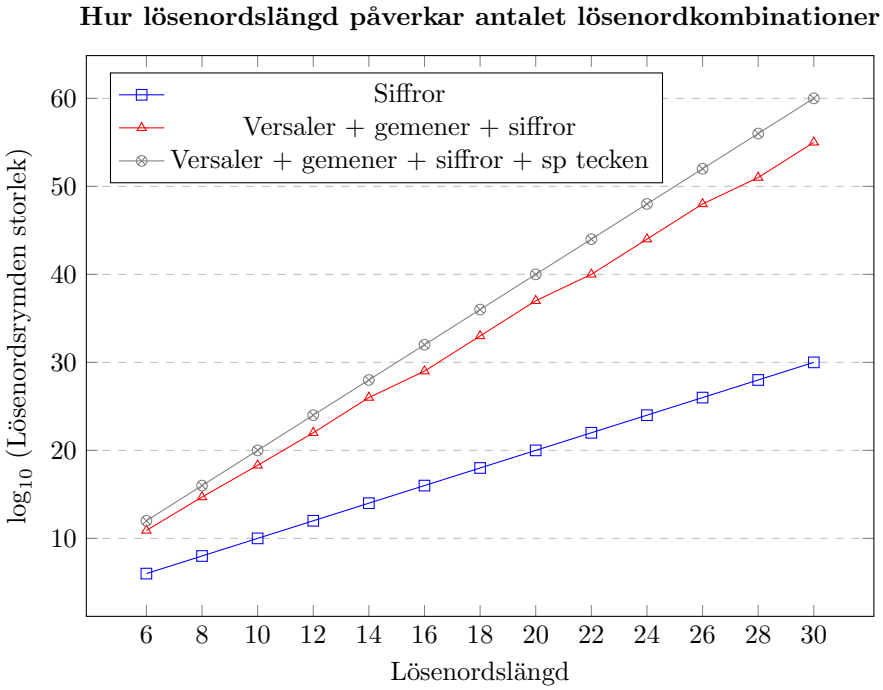
$$\log_{10}(L)$$

där L är lösenordsrymden.

I Figur 1 illustreras hur lösenordsrymdens storlek påverkas av lösenordslängden. Eftersom summorna snabbt blir så stora att de är svåra att hantera utgår grafen ifrån antalet siffror som respektive lösenordsrymdens storlek består av, det vill säga 10-logaritmen av lösenordsrymdens storlek avrundat till närmaste mindre heltal.

Lösenords-längd	Alfabet	Antalet möjliga lösenord (L)	$\log_{10}$ (L)
6	S	10^6	6
6	V,G,S	98867482624	11
6	V,G,S,Sp	1061520150601	12
8	S	10^8	8
8	V,G,S	457163239653376	15
8	V,G,S,Sp	10828567056280800	16
10	S	10^{10}	10
10	V,G,S	211392282015721 * 10^4	18
10	V,G,S,Sp	11046221254112 * 10^7	20
12	S	10^{12}	12
12	V,G,S	977477912040694 * 10^7	22
12	V,G,S,Sp	112682503013197 * 10^{10}	24
14	S	10^{14}	14
14	V,G,S	451985786527617 * 10^{11}	26
14	V,G,S,Sp	114947421323762 * 10^{14}	28
16	S	10^{16}	16
16	V,G,S	20899822769037 * 10^{16}	29
16	V,G,S,Sp	11725786449237 * 10^{19}	32
18	S	10^{18}	18
18	V,G,S	966407804840271 * 10^{18}	33
18	V,G,S,Sp	119614747568666 * 10^{22}	36
20	S	10^{20}	20
20	V,G,S	446866968958141 * 10^{22}	37
20	V,G,S,Sp	122019003994797 * 10^{26}	40
22	S	10^{22}	22
22	V,G,S	206631286446245 * 10^{26}	40
22	V,G,S,Sp	124471585975092 * 10^{30}	44
24	S	10^{24}	24
24	V,G,S	955463068527435 * 10^{29}	44
24	V,G,S,Sp	126973464853191 * 10^{34}	48
26	S	10^{26}	26
26	V,G,S	441806122887086 * 10^{33}	48
26	V,G,S,Sp	129525631496741 * 10^{38}	52
28	S	10^{28}	28
28	V,G,S	204291151222989 * 10^{37}	51
28	V,G,S,Sp	132129096689825 * 10^{42}	56
30	S	10^{30}	30
30	V,G,S	944642283255099 * 10^{40}	55
30	V,G,S,Sp	134784891533291 * 10^{46}	60

Tabell 4: Hur lösenordslängd påverkar antalet möjliga lösenordskombinationer. Tabell 4 visar hur lösenordslängd (n) och alfabet (A) påverkar antalet möjliga lösenordskombinationer. Antalet möjliga lösenord var beräknat enligt ekvation $|A|^n$, där A representerar lösenordsalfabet och n representerar lösenordslängd. Antalet möjliga lösenord blir snart enormt. Antalet blir så stort att det blir svårt att presentera på ett tydligt sätt. Tabellen visar även hur många siffror som respektive lösenordsrymdens storlek består av, dvs 10-logaritmen av lösenordsrymdens storlek avrundat till närmaste mindre heltal



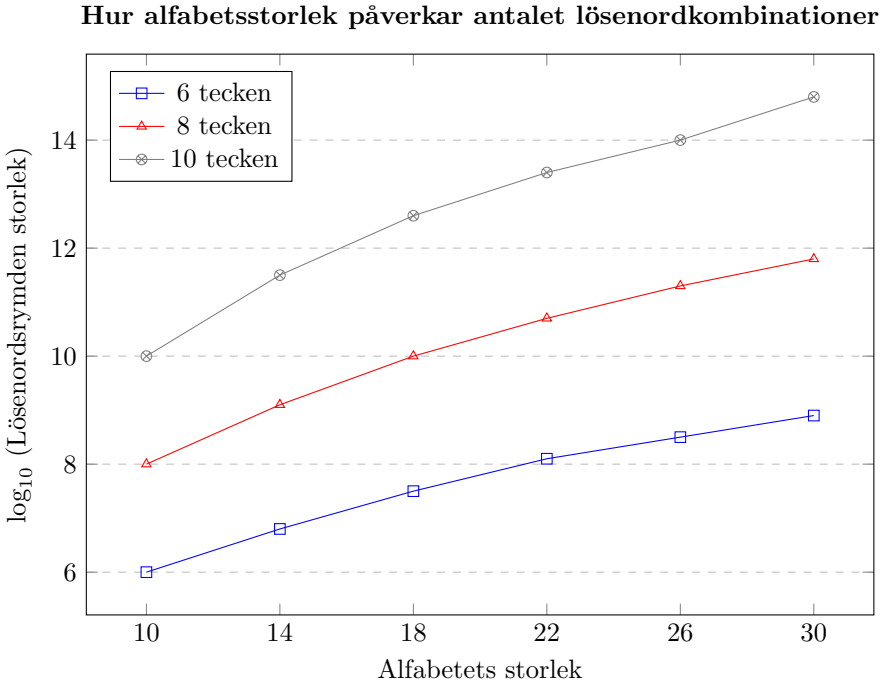
Figur 1: Hur lösenordslängd påverkar antalet lösenordkombinationer. I Figur 1 används antal siffror som respektive lösenordsrymdens storlek består av, det vill säga 10-logaritmen av lösenordsrymdens storlek avrundat till närmaste mindre heltal för att illustrera hur lösenordslängd påverkar lösenordsrymd. Figuren visar hur lösenordslängd påverkar lösenordsrymd/antalet möjliga lösenord för alfabetet, siffror, versaler + gemener + siffror och versaler + gemener + siffror + specialtecken

Tabell 5 visar hur ett lösenords alfabet påverkar lösenordsrymden, det vill säga antalet möjliga lösenord. Tabellen visar även antalet siffror som respektive lösenordsrymdens storlek består av, det vill säga 10-logaritmen av lösenordsrymdens storlek.

Lösenordslängd	Alfabets-storlek	Antalet möjliga lösenord (L)	$\log_{10}$ (L)
6	10	1000000	6
6	12	2985984	6,4
6	14	7529536	6,8
6	16	16777216	7,2
6	18	34012224	7,5
6	20	64000000	7,8
6	22	113379904	8,1
6	24	191102976	8,3
6	26	308915776	8,5
6	28	481890304	8,7
6	30	729000000	8,9
8	10	100000000	8
8	12	429981696	8,6
8	14	1475789056	9,1
8	16	4294967296	9,6
8	18	11019960576	10
8	20	25600000000	10,4
8	22	54875873536	10,7
8	24	110075314176	11
8	26	208827064576	11,3
8	28	377801998336	11,6
8	30	656100000000	11,8
10	10	10000000000	10
10	12	61917364224	10,8
10	14	289254654976	11,5
10	16	1099511627776	12
10	18	3570467226624	12,6
10	20	10240000000000	13
10	22	26559922791424	13,4
10	24	63403380965376	13,8
10	26	141167095653376	14
10	28	296196766695424	14,4
10	30	590490000000000	14,8

Tabell 5: Hur alfabetsstorlek påverkar antalet möjliga lösenordskombinationer. Antalet möjliga lösenord var beräknat enligt ekvation $|A|^n$, där A representerar lösenordsalfabet och n representerar lösenordslängd. Antalet möjliga lösenord blir snart enormt. Antalet blir så stort att det blir svårt att presentera på ett tydligt sätt. Tabellen visar även hur många siffror som respektive lösenordsrymdens storlek består av, dvs 10-logaritmen av lösenordsrymdens storlek

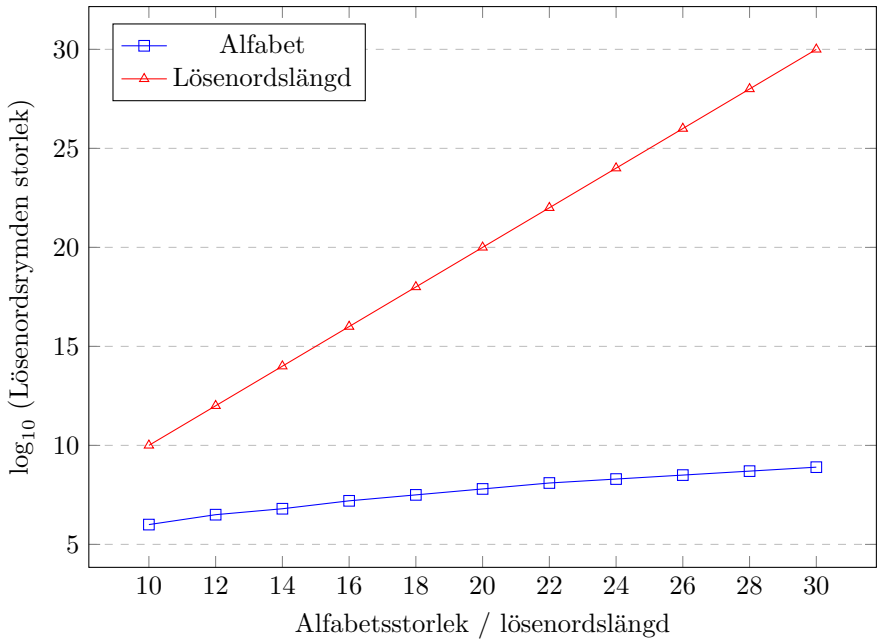
I Figur 2 används antal siffror för att visualisera hur alfabet påverkar lösenordsrymdens storlek.



Figur 2: Hur alfabetsstorlek påverkar antalet möjliga lösenord. I Figur 2 används antal siffror som respektive lösenordsrymdens storlek består av, det vill säga 10-logaritmen av lösenordsrymdens storlek för att illustrera hur lösenordslängd påverkar lösenordsrymd. Eftersom summorna snabbt blir så stora att de är svåra att hantera utgår grafen ifrån antalet siffror i respektive lösenord, istället för själva summan. Figuren visar hur alfabet påverkar lösenordsrymd/antalet möjliga lösenord för ett lösenord med 6 tecken, 8 tecken och 10 tecken.

Figur 3 demonstrerar hur en ökning av lösenordslängd eller alfabetet ökar antalet möjliga lösenord. Start positionen var alfabetets (A) storlek lika med 10 och lösenordslängd (n) lika med 6. Antalet möjliga lösenord var beräknat enligt sambandet $|L| = |A|^n$, där L är lösenordsrymden.

Jämförelse om hur en ökning i lösenordslängd eller alfabet påverkar lösenordsrymd



Figur 3: Jämförelse av hur en ökning i lösenordslängd eller alfabet påverkar antalet möjliga lösenord. Figur 3 demonstrerar hur en ökning av lösenordslängd eller alfabetet ökar antalet möjliga lösenord. Start positionen var alfabetets (A) storlek lika med 10 och lösenordslängd (n) lika med 6. Antalet möjliga lösenord var beräknat enligt sambandet $|L| = |A|^n$, där L är lösenordsrymden.

En modern dator bearbetar 10^9 lösenord per sekund, se Brumen och Cernezel [1]. Tabell 6 visar tiden som krävs att knäcka vissa lösenord. Moores lag anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastigheten inom 10 år kommer att vara 100 gånger högre än idag, se Brumen och Cernezel [1]. Tabell 6 visar också tiden som krävs att knäcka samma lösenord om tio år.

Lösenords-längd	Alfabet storlek	Antalet möjliga lösenord	Nu	Om tio år
6	10	10^6	0,001 sek	0,00001 sek
6	68	98867482624	1 min 38 sek	0,98 sek
6	101	1061520150601	17 min 42 sek	10,6 sek
8	10	10^8	0,1 sek	0,001 sek
8	68	457163239653376	5 dgr 7 tim 12 min	12 tim 43 min
8	101	10828567056280800	125 dgr 7 tim 12 min	1 dag 6 tim
10	10	10^{10}	10 sek	0,1 sek
10	68	211392282015721 * 10^4	67 år	244 dgr 17 tim 5 min
10	101	11046221254112 * 10^7	3503 år	35 år
12	10	10^{12}	16 min 40 sek	10 sek
12	68	977477912040694 * 10^7	309956 år	3099,6 år
12	101	112682503013197 * 10^{10}	35731387 år	357313,9 år
16	10	10^{16}	115 dgr 16 tim 48 min	1 dag 3 tim 46 min
16	68	20899822769037 * 10^{16}	6627290325037 år	66272903250,4 år
16	101	11725786449237 * 10^{20}	3718222491513500 år	37182224915135,0 år

Tabell 6: Tiden som krävs för att beräkna alla hashar för en lösenordsrymd. Den rapporterade hastigheten för att bearbeta ett klartextlösenord till ett hashvärde är 10^9 *kombinationer per sekund, se Brumen och Cernezel [1]. Det innebär att en dator bearbetar 8.64 *10^{13} *lösenord per dag och 2.6784 *10^{15} *efter 31 dagar. Tabell 6 visar tiden som krävs att knäcka vissa lösenord. Moores lag anger att datorkraft fördubblas ungefär var artonde månad. Detta innebär att hastigheten inom 10 år kommer att vara 100 gånger högre än idag, se Brumen och Cernezel [1]. Tabell 6 visar också tiden som krävs att knäcka samma lösenord om tio år.*

Den rapporterade hastigheten för att bearbeta ett klartext lösenord från ett hashvärde är 10^9 kombinationer per sekund, se Brumen och Cernezel [1]. Det innebär att en dator bearbetar 8.64 * 10^{13} lösenord per dag och 2.6784 * 10^{15} efter 31 dagar.

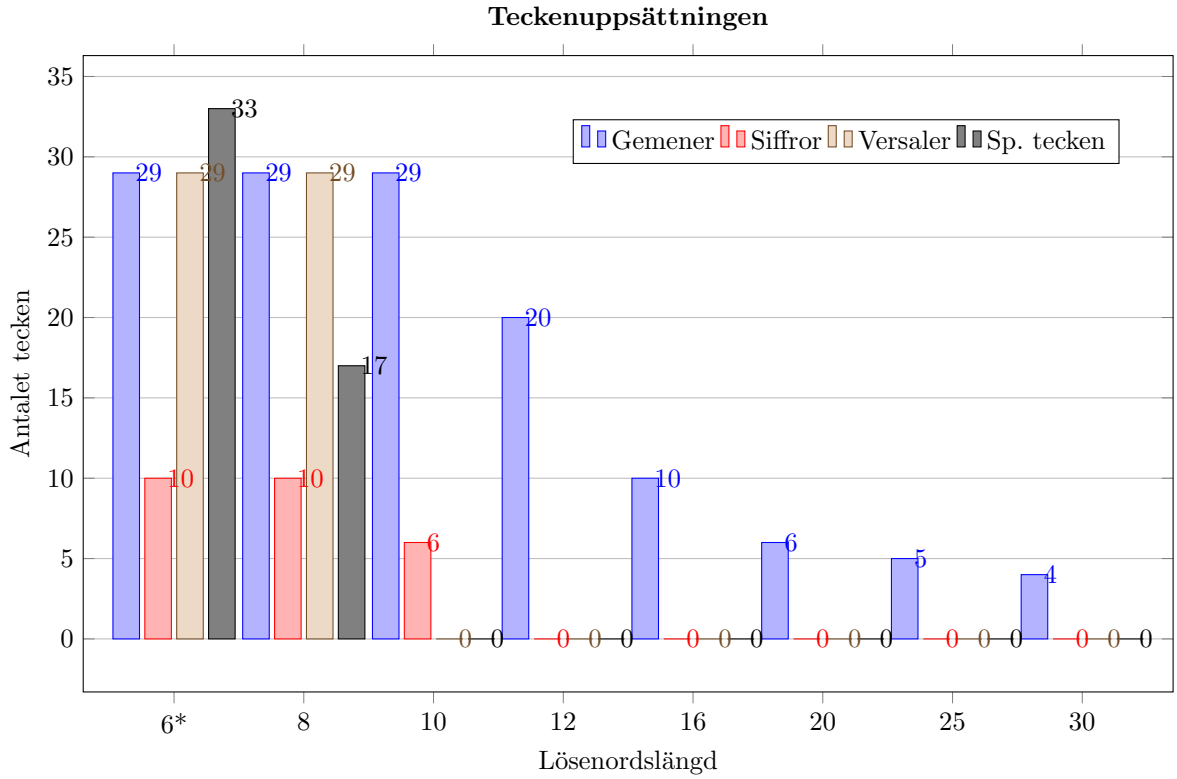
Tabell 7 visar antalet tecken som krävs för att skapa lösenord som i genomsnitt tar 31 dagar att knäcka med brute force knäckning.

För att ett lösenord ska ta minst 31 dagar att knäcka krävs att alfabetet innehåller en viss teckenuppsättning. Till exempel kräver ett lösenord som är åtta tecken långt ett alfabet med 85 olika tecken. Ett sådant alfabet kan skapas med 10 siffror, 29 gemener, 13 versaler och 33 specialtecken eller med 5 siffror, 29 gemener, 29 versaler och 22 specialtecken. Båda dessa teckenuppsättningar resulterar i ett alfabet i samma storlek (85). Forskning har dock visat att användare

Lösenordslängd	Alfabetsstorlek	Antalet möjliga lösenord	Tid att knäcka lösenord
6*	101	1061520150601	0,009 dagar (13 minuter)
8	85	2724905250390620	31 dagar
10	35	2758547353515620	31 dagar
12	20	4096000000000000	47 dagar
16	10	10000000000000000	115 dagar
20	6	3656158440062980	43 dagar
25	5	298023223876953000	3450 dagar
30	4	1152921504606850000	13344 dagar

*Tabell 7: Antalet tecken som krävs för att skapa ett lösenord som tar i genomsnitt 31 dagar att knäcka med brute force. Tabell 7 visar antalet tecken krävs för att skapa ett lösenord som ta minst 31 dagar att knäcka med brute force knäckning. *Observera att det är omöjligt att skapa ett sådant lösenord med bara sex tecken, givet den processorhastighet och alfabetsstorlek som testas.*

har tydliga preferenser när det gäller teckenuppsättningen, se Tupsamudre et al [23]. Användare välja små bokstäver, sen siffror, sen versaler och i sista hand specialtecken. Figur 4 nedan presenterar antalet tecken som krävs utifrån de här preferenserna. Det vill säga gemener är valt först, därefter siffror, sedan versaler och till sist specialtecken.



Figur 4: Teckenuppsättningen för ett lösenord som tar i genomsnitt 31 dagar att knäcka med brute force. Figur 4 presenterar teckenuppsättningen efter användare preferenser. Det vill säga gemener är valt först, därefter siffror, sedan versaler och till sist specialtecken

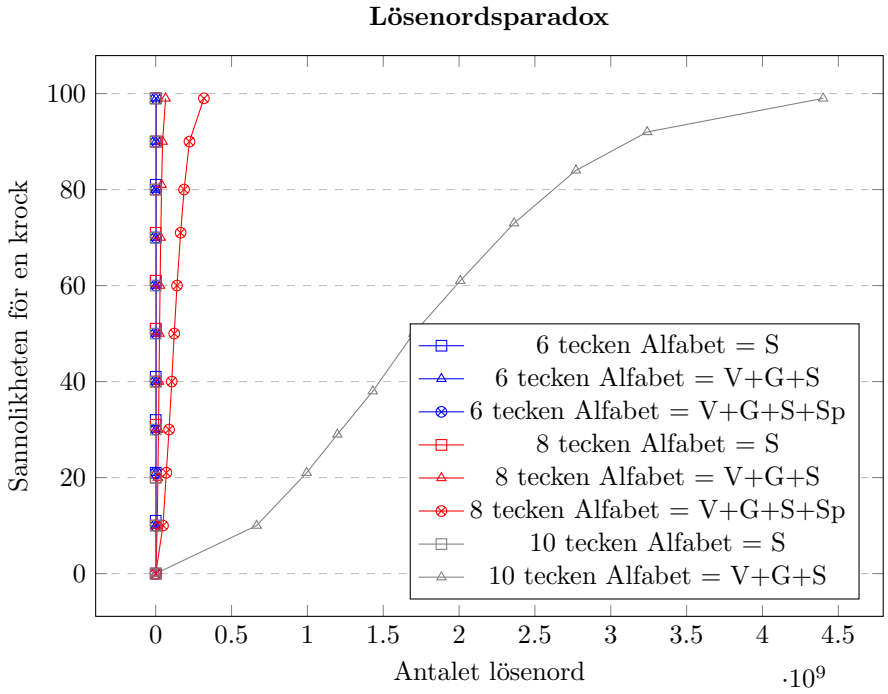
5.2 Lösenordsparadoxen

Tabell 8 visar sannolikheten för en krock för vissa lösenord, det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet. Det finns tre olika alfabet kombinationer i Tabellen. Där S representerar siffror, V representerar versaler, G står för gemener och Sp står för specialtecken.

Lösenords-längd	Sannolikheten för minst en krock (procent)	Användare S	Användare V,G,S	Användare V,G,S,Sp
6	20	692	214149	698259
6	50	1196	369654	1203339
6	80	1811	563183	1850554
6	99	3087	961697	3131610
8	20	6617	14266038	69879562
8	50	11969	25012267	122084775
8	80	17872	38797212	186990447
8	99	30414	65082764	317488076
10	20	67176	994903794	-
10	50	117321	1699320122	-
10	80	178090	2770649306	-
10	99	305277	4400007077	-

Tabell 8: Sannolikheten för en krock. Det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet. Det finns tre olika alfabet kombinationer i Tabellen. S representerar siffror, V, G, S representerar versaler, gemener och siffror. V,G,S,Sp står för versaler, gemener, siffror och specialtecken. Bindesstreck antyder att beräkningen ej hunnit slutföras vid rapportens inlämning.

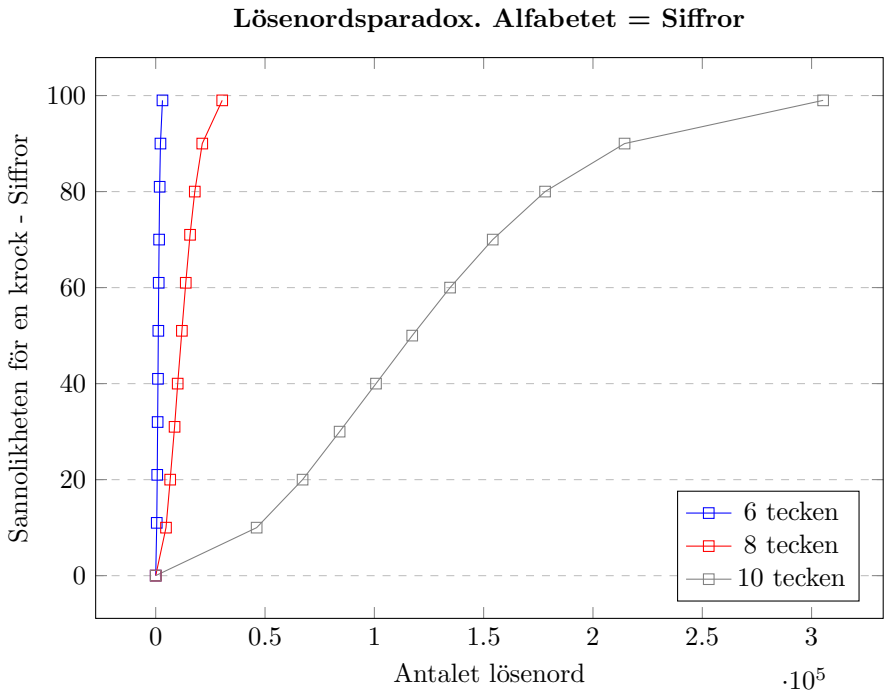
Risken för en krock är visualiserad i graf 5. Grafen presenterar informationen från Tabell 8.



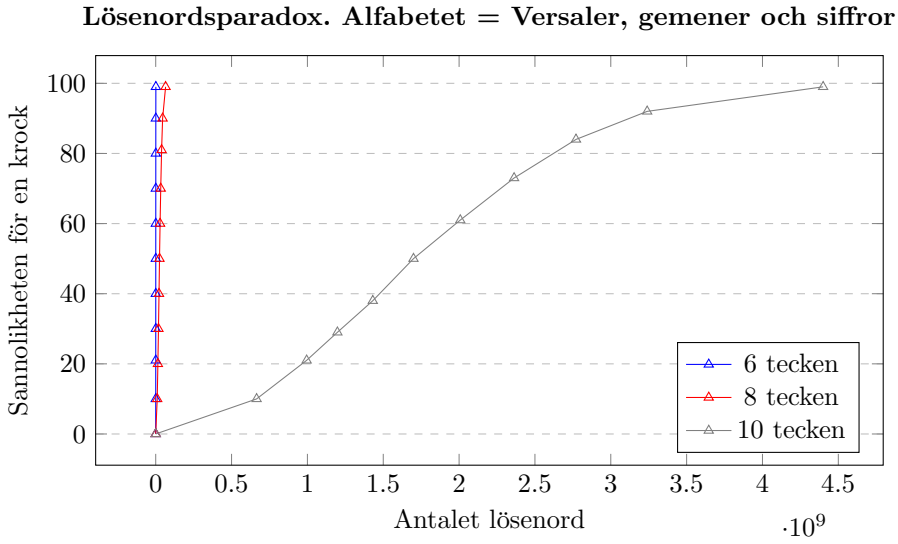
Figur 5: Sannolikheten för en krock. Det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet. Grafen presenterar informationen från Tabell 8.

figuren nedan presenterar risk för en krock med de tre olika alfabetet. De demonstrerar informationen i Tabell 8.

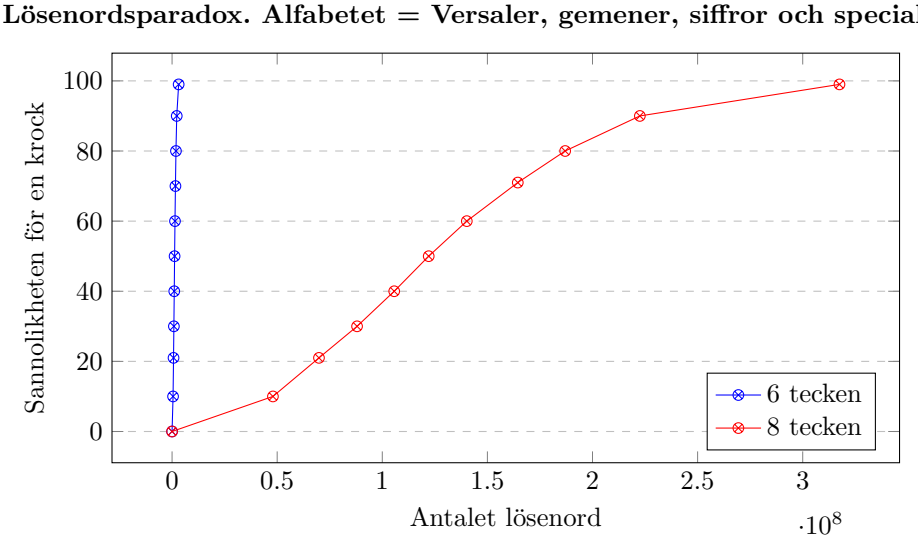
Figur 6 presenterar risk för en krock med en alfabetet som innehålla siffror. Figur 7 visar risk för en krock med alfabetet versaler, gemener och siffror. Figur 8 visar risk för en krock med alfabetet versaler, gemener, siffror och specialtecken.



Figur 6: Sannolikheten för en krock med en alfabetet som innehålla siffror. Det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet.



Figur 7: Sannolikheten för en krock med alfabetet versaler, gemener och siffror. Det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet.



Figur 8: Sannolikheten för en krock med alfabetet versaler, gemener, siffror och specialtecken. Det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet.

5.3 Semantik

De olika databaserna innehåller i stort underhållningsrelaterade tjänster, och inte säkerhetskritiska användningsområden som banktjänster.

Tabell 9 visar de vanligast förekommande lösenorden i den använda testdatan.

Plats	Spel	Hälsa	Bokaffär
1	212122	healers	123456
2	lol123	sunshine	nato
3	1996323	hypnosis	qwerty
4	lol	massage	123456789
5	lopas123	charlie	12345
6	123456	osteopath	password
7	pure3gun4	premium	666666
8	lol1	holistic	12345678
9	abc123	spirit	111111
10	123321	elephant	000000
11	hehe	therapy	123321
12	killer	angels	1234
13	tjrocks95	namaste	123123
14	pokemon	password	otan
15	owned	george	1234567
16	ismail	healing	natootan
17	lobster	abundance	123
18	powerman	orange	qazwsx
19	poopypoopy	matthew	tridentd5
20	runescape	monkey	1234567890

Tabell 9: De 20 mest frekvent förekommande lösenorden i den testdata arbetet omfattar. Detta är beräknat genom frekvensen av identiskt återkommande lösenord från skilda användare i en och samma databas.

Tabell 10 visar de mest förekommande segmenten i testdatan.

Plats	Spel	Hälsa	Bokaffär
1	eater	xxx	man
2	man	cat	tan
3	cool	therapy	jan
4	poop	rose	love
5	bo	healing	pass
6	killer	healers	i
7	life	angel	blue
8	owned	tigg	art
9	connor	harry	abc
10	king	man	mine
11	don	bear	pit
12	bin	dog	password
13	kid	jessica	nicole
14	boy	bowen	master
15	josh	premium	lynx
16	love	angels	sam
17	ball	massage	crean
18	nigger	tan	security
19	law	master	rut
20	hit	elizabeth	mar

Tabell 10: De 20 mest frekvent förekommande segmenten i den testdata arbetet omfattar. Detta är beräknat genom att analysera samtliga lösenord i testdata och dokumentera vilka segment lösenorden är uppbyggda av. Segment listas i tabellen rangordnat efter mest förekommande frekvens.

Tabell 11 visar de vanligaste semantiska kategorierna för segment i testdatan. Semantik i det här fallet avser ett samlingsnamn för ett antal synonymer, matchat från ett ordnät. n.01 antyder ett samlingsset av substantiv och v.01 ett av verb.

Plats	Spel	Hälsa	Bokaffär
1	male_names	female_names	male_names
2	last_names	male_names	female_names
3	cities	cities	lastnames
4	female_names	lastnames	cities
5	own.v.01	therapist.n.01	man.n.01
6	eater.n.01	countries	countries
7	man.n.01	therapy.n.01	tan.n.01
8	killer.n.01	thirty.n.01	mine.n.01
9	cool.n.01	mend.n.01	rudiment.n.01
10	crap.n.01	cat.v.01	password.n.01
11	life.n.01	man.n.01	art.n.01
12	ball.n.01	peace.n.01	baseonballs.n.01
13	ma.n.01	dog.n.01	iodine.n.01
14	pullthelegof.v.01	reflexology.n.01	pit.n.01
15	nigger.n.01	premium.n.01	blue.n.01
16	chicken.n.01	massage.n.01	maestro.n.01
17	rock.n.01	dancer.n.01	professioinal.n.01
18	bin.n.01	Lashkar-e-Taiba.n.01	wolf.n.01
19	malechild.n.01	sunlight.n.01	NorthAtlanticTreatyOrganization.n.01
20	hit.v.01	dragon.n.01	lynx.n.01

Tabell 11: De 20 mest frekvent förekommande semantiska generaliseringarna i den testdata arbetet omfattar. Detta är beräknat genom att analysera samtliga segment i testdata och dokumentera vilka semantiska kategorier som är mest frekvent förekommande. Två segment kan tillhöra samma semantiska generalisering. Semantiska kategorier listas i Tabellen rangordnat efter mest förekommande frekvens. Semantik i det här fallet avser ett samlingsnamn för ett antal synonymer, matchat från ett ordnät. n.01 antyder ett samlingsset av substantiv och v.01 ett av verb.

5.4 Procedur för att öka lösenordsstyrkan

Arbetet har undersökt lösenordsval och lösenords sårbarhet för brute force knäckning och knäckning med hjälp av ordlistor. Det finns några enkla åtgärder som en användare eller ett företag kan implementera för att stärka ett lösenord.

- Undvik vanliga lösenord.

  Det snabbaste sätt att knäcka ett lösenord är med hjälp av ordlistor. Ordlistan innehåller allmänt ord, namn, stavelser, siffror och kombinationer av dessa, samt erfarenhetsmässigt vanliga lösenord som “qwerty”. En

modern dator bearbetar enorma antal kombinationer varje sekund och datorer kommer att bli ännu snabbare i framtiden. Det är därför bra att inte använda ett ord från ett lexikon för att förhindra sårbarhet mot en så kallad “dictionary attack”.

- “Mangling” av ett lösenord är inte tillräckligt

  “Mangling” innebär att förändra en del av lösenord med något annat. “iloveyou” skulle exempelvis förändras till “1loveYou” där i är ersättas med 1 och y med Y. Leet är en annan version av mangling där varje bokstäv kan ersättas med någon annat, till exempel “a” kan ersättas med en av “4”, “/\”, “/–\”, “^” eller “aye”, se Flamand [7]. Problemet är att lösenordsknäckningsmjukvara som Hashcat och John the Ripper erbjuder funktioner som generar en ”manglad” version av en ordlista. Mangling hjälper därför inte om ursprungslösenordet finns i en ordlista.

- Öka lösenordsrymden

  En större lösenordsrymd innebär att det finns flera möjliga lösenordskombinationer. En användare kan öka lösenordsrymden genom ökning av lösenordslängden eller tillägg av tecken i alfabetet, se Tabell 8. En ökning i lösenordslängd resulterar i betydligt fler möjliga lösenord än en motsvarande ökning i ett lösenords alfabet, se Figur 3.

  – Öka lösenordslängden

    Ett bra sätt att öka lösenordsrymd är att kombinera tre eller fler ord till ett lösenord. Det är dock viktigt att orden inte är relaterade till varandra. Till exempel är “halmstadbollklubb” ett långt lösenord (17 tecken), med en ganska lång lösenordsrymd 29^{17}. Men orden “halmstad”, “boll” och “klubb” används ofta ihop, vilket medför en risk att lösenordet kan finnas med i en ordlista. Det är viktigt att kombinera minst tre ord eftersom datorer idag bearbetar så många lösenordskombinationer att det finns risk att datorer snart kommer att klara av att även kombinera alla ord med varandra.

  – Öka lösenordsalfabetet

    Ett annat sätt att öka lösenordsrymd är att öka lösenordsalfabetet. Det vill säga lösenord består av en viss kombination av versaler, gemener, siffror och specialtecken. På det här sättet kan en användare

skapa ett starkt lösenord som är kortare än ett lösenord som består av bara ett alfabet. Ett lösenord med mindre än 10 tecken är dock sårbart för brute force knäckning oavsett innehåll, se Tabell 6.

- Använd utländska bokstäver

  Brute force knäckning och knäckning med hjälp av ordlistor fungera ofta på samma antagande. Nämligen att lösenord har skapats med tecken från det lokala språket. I Sverige innebär det svenska, som innehåller latinska bokstäver. Ett lösenord som innehåller en utländsk bokstav (som exempelvis ryska eller thai) kommer att bli mycket svårare att knäcka.

- Använd en annan eller kompletterande typ av autentisering

  Arbetet visar att det finns tydlig sårbarhet i användandet av lösenord. Ett sätt att öka lösenordsäkerhet är att kombinera lösenord med en annan typ av autentisering. Banker över hela världen använder nu tvåstegsverifiering innan en användare få tillgångar till tjänster. Tvåstegsverifiering är ett bra sätt att öka säkerhet om konsekvenser av ett IT-brott är allvarliga, till exempel stor ekonomisk förlust. Dock kan det finnas ett värde i att använda rätt verktyg för jobbet. En skärmsläckare behöver nödvändigtvis inte en tvåstegsverifiering.

Framförallt handlar det om att användare väljer ett lösenord som är otippat och att lösenordsrymden är så stor att det blir svårt att knäcka med brute force knäckning.

6 Diskussion

Denna sektion avser reflektera över resultaten, och sätta dem i sitt sammanhang.

6.1 Lösenordsrymd

En ökning i lösenordslängd resulterar i betydligt fler möjliga lösenord än en motsvarande ökning i ett lösenords alfabet. Tabell 4 visar att ett lösenord med 6 tecken vars alfabet innehåller enbart siffror, det vill säga tio olika tecken, resulterar i 1 000 000 (10^6) möjliga lösenord. En ökning i lösenordslängd till åtta tecken resulterar i 100 000 000 (10^8) möjliga lösenord. Ökningen av längd med två tecken har därför resulterat i en ökning av 99 000 000 möjliga lösenord. En ökning av två tecken i lösenordsalfabet istället för lösenordslängd resulterar i 2 985 984 möjliga lösenord, en ökning med bara 1 985 984. Det är 97 014 016 färre möjliga lösenord än en motsvarande ökning av lösenordslängd. Figur 1 och 2 visularserar hur en ökning av lösenordslängd eller alfabetet ökar antalet möjliga lösenord. Figur 3 visar hur en ökning i lösenordslängd eller alfabet påverkar antalet möjliga lösenord. Den linje som representerar lösenordslängd stiger betydligt mer än den linje som representerar lösenordsalfabet.

Detta stöder Tupsamudres et al [23] arbete som påstår att lösenordslängd är den viktigaste faktorn när det gäller lösenordssäkert. Det stöder också logiken bakom att många webbsidor idag kräver en viss lösenordslängd.

I det här arbetet var ett delmoment att uppskatta hur snabbt en modern dator kunde knäcka ett lösenord med brute force-knäckning, samt hur lösenordsrymdens storlek påverkar den mängd tid som krävs. En modern dator var definierad som en dator som uppfyllde specifikationen i Brumen och Cernezels arbete [1], det vill säga en desktop som hade ett AMD Radeon HD 7970 Graphical Processing Unit (GPU) installerad. Systemet använde mjukvaran Hashcat för att matcha hashar och lösenord. Det är värt att notera att specifikationen inte är avancerad. Ett AMD Radeon HD 7970 GPU kostar cirka $500 och Hashcat är open source mjukvara, så den är gratis. Moores lag [16] anger att datorkraft fördubblas ungefär var 18:e månad. Detta innebär att hastigheter inom 10 år kommer att vara 100 gånger snabbare än idag, vilket i sin tur innebär att datorer om 10 år kommer att knäcka lösenord 100 gånger snabbare än idag.

Den tid som krävs för att knäcka vissa lösenord finns i Tabell 6. Den visar att ju längre ett lösenord är desto längre tid krävs för att knäcka ett lösenord.

Det är omöjligt att skapa lösenord som aldrig kan knäckas, se Egelman et al [6]. Ett viktigt koncept inom lösenordssäkerhet är att skapa ett lösenord som är tillräckligt starka. Vad som bedöms som tillräckligt starka beror mycket på vad det är för information som ska skyddas. Microsoft generarar till exempel ett starkt lösenord som är 25 tecken långt för att bekräfta köp vid e-handel. Många webbsidor kräver dock ett lösenord som bara är åtta tecken långt. Tabell 6 visar också tiden som krävs för att knäcka lösenord om tio år. Tabellen visar att lösenord som är starka idag förmodligen inte kommer att vara lika starka i framtiden. Till exempel, ett lösenord som är åtta tecken långt med ett möjligt alfabet av 101 olika tecken kräver upp till 125,3 dagar att knäcka idag. Det borde vara tillräckligt mycket tid för en IT-avdelning att märka att någon obehörig har fått tillgång till informationen och be användarna byta lösenord. Men om tio år kan ett sådant lösenord knäckas inom 1,25 dagar, vilket kanske inte är tillräckligt med tid för att märka ett intrång och byta alla lösenord. Minskningen i tid som krävs för att bearbeta lösenord har ännu större betydelse för information som är sparad off-line. En hårddisk som innehåller känslig information och skyddas av ett lösenord som betraktas som starkt idag kan bli en säkerhetsrisk längre fram. En obehörig som har tillgång till hårddisken några år senare när lösenord blivit lättare att knäcka kan då relativt enkelt få tillgång till känsliga uppgifter.

Den tid som krävs för att knäcka ett lösenord anses som den viktigaste faktorn när det gäller lösenordssäkerhet, se Cazier och Medlin [3], Creese et al [5] och Egelman et al [6].

Tabell 7 visar antalet tecken som krävs för att skapa ett lösenord som i genomsnitt tar 31 dagar att knäcka med brute force knäckning. Tidsramen 31 dagar valdes eftersom det bedöms som tillräckligt med tid för att ett företag ska upptäcka ett intrång och implementera lämpliga åtgärder som exempel att byta alla lösenord. Det kan finnas situationer när 31 dagar inte ger tillräckligt skydd och där konsekvensen av ett dataintrång resulterar i en stor förlust. Just detta antal dagar har dock bedömts tillräckligt i det här arbetet för att illustrera hur lösenordslängd och alfabet påverkar hur mycket tid som krävs för att knäcka ett lösenord. Tabell 7 visar att det är omöjligt att skapa ett sådant lösenord med enbart sex tecken och den maximala alfabetsstorleken av 101 tecken som beaktas här. Detta innebär att det inte spelar någon roll vilket lösenord en användare har om lösenordet bara är sex tecken långt. Lösenordet kommer att knäckas inom 13 minuter. Det är fortfarande svårt att skapa ett säkert lösenord med åtta tecken.

För ett lösenord som inte går att knäcka på 31 dagar och är åtta tecken långt krävs ett alfabet med 85 olika tecken. Det betyder att lösenorden måste innehålla ett tecken från varje alfabet (gemener, versaler, siffror och specialtecken). Tabell 7 visar att ju längre ett lösenord är desto mindre storlek på alfabetet krävs för att skapa ett tillräckligt starkt lösenord. Tabellen visar att ett lösenord som är 30 tecken långt kräver ett alfabet av bara fyra tecken för att skapa en betryggande lösenordsstyrka. Ett sådant lösenord kräver dessutom betydligt längre tid än 31 dagar att knäcka, nämligen upp till 36,5 år.

Ju större en lösenordsrymd är, desto svårare är det att knäcka lösenord med brute force knäckning, se Tabell 6. Det finns två sätt att öka lösenordsrymdens storlek, nämligen genom att antingen öka lösenordslängd eller genom att göra tillägg av tecken i alfabetet. Ett lösenord som är tio tecken långt med ett alfabet som består av enbart siffror (10 tecken) kan producera 10^{10} möjliga lösenordskombinationer som lätt knäcks (det skulle ta 10 sekunder att testa alla möjliga kombinationer, se Tabell 6). Ett lika långt lösenord med ett alfabet som består av versaler, gemener och siffror producerar en betydligt större lösenordsrymd som vars lösenord är betydligt svårare att knäcka (67 år skulle det att testa alla möjliga kombinationer, givet att Moore's lag inte tas med i beräkningen). Ett lika långt lösenord med ett alfabet som består av versaler, gemener, siffror och specialtecken har ännu större lösenordsrymd och kräver ännu längre tid att testa alla möjliga kombinationer – 3503 år. Detta innebär att en användare kan skapa ett starkt lösenord med bara tio tecken med det måste innehålla ett tillräckligt stort alfabet, det vill säga helst en kombination av versaler, gemener, siffror och specialtecken. Ett sådant lösenord kan bli komplicerat och därför svårt att komma ihåg. En användare kan istället välja att skapa ett starkt lösenord med ett mindre alfabet, men då krävs att lösenordet är tillräckligt långt. Ett lösenord som är tolv tecken långt med ett alfabet som består av enbart gemener producerar 353814783205469000 (29^{12}) möjliga lösenordskombinationer. Det skulle ta mer än elva år att testa alla möjliga kombinationer. Ett sådant lösenord är längre men kanske lättare att komma ihåg eftersom det kan bestå av bara ord. Det är dock viktigt att orden inte är relaterade till varandra för att undvika risk att lösenordet kan finnas med i en ordlista, (se avsnitt 5.4 Procedur för att öka lösenordsstyrkan).

För att ett lösenord ska ta minst 31 dagar att knäcka krävs att alfabetet innehåller en viss teckenuppsättning. Till exempel kräver ett lösenord som är åtta

tecken långt ett alfabet med 85 olika tecken. Ett sådant alfabet kan skapas med 10 siffror, 29 gemener, 13 versaler och 33 specialtecken eller med 5 siffror, 29 gemener, 29 versaler och 22 specialtecken. Båda dessa teckenuppsättningar resulterar i ett alfabet i samma storlek (85). Forskning har dock visat att användare har tydliga preferenser när det gäller teckenuppsättningen, se Tupsamudre et al [23]. Användare väljer helst gemena bokstäver, sedan siffror, sedan versaler och i sista hand specialtecken. Figur 4 nedan presenterar antalet tecken som krävs utifrån de här preferenserna. Det vill säga gemener är valt i första hand, siffror i andra hand, i tredje hand versaler och slutligen i sista hand specialtecken.

figuren visar att ett säkert lösenord i dagsläget inte kräver specialtecken om lösenordet är längre än 8 tecken. Versaler behövs inte med lösenord längre än 10 tecken medan siffror inte behövs för lösenord längre än 12 tecken. För lösenord längre än 12 tecken krävs enbart gemener för att skapa ett starkt lösenord. Tabellen illustrerar att ju längre ett lösenord är desto mindre storlek krävs på alfabetet för att skapa ett tillräckligt starkt lösenord.

6.2 Lösenordsparadoxen

Arbetet har hittills utgått från beräkningar av hur lång tid som krävs för att knäcka varje möjlig lösenordskombination. Det är dock osannolikt att en angripare skulle behöva testa varje kombination innan ett lösenord kan knäckas. Ofta används en ordlista inledningsvis, innan varje möjlig kombination testas, eftersom en ordlisteattack är mycket snabbare än brute force knäckning. Angriparen behöver inte använda brute force knäckning om lösenord knäcks med en ordlista. Även om en ordlisteattack misslyckats och angriparen måste använda brute force knäckning är det fortfarande osannolikt att en angripare behöver testa varje kombination.

Tabell 8 visar sannolikheten för en krock för vissa lösenord, det vill säga sannolikheten att vid minst två användare har samma lösenord inom en viss kombination av lösenordslängd och lösenordsalfabet. Detta är viktigt, då det är ett första steg i att påvisa att en lösenordsrymd kan vara missvisande. Exempelvis, i ett sex tecken långt lösenord bestående av enbart siffror, finns det 1 000 000 möjliga kombinationer. Men redan vid 1196 totalt använda lösenord är det en 50% sannolikhet att minst två användare har valt samma lösenord. Alltså mer sannolikt att de har än att de inte har. Och detta är givet något av en utopi, där samtliga användare väljer lösenord med likartad spridning, helt

utan preferenser för enkla mönster som 123456. Vid 3087 användare är det 99% sannolikhet att minst två användare har valt samma lösenord.

6.3 Semantiska mönster i intresseindelade databasdumpar

De mest frekvent förekommande segmenten bär liknelser med de mest frekvent förekommande lösenorden, men det finns också intressanta observationer att göra. Vad det gäller testdata för ett spelcommunity, så är egennamn tillhörande tv- och dataspel inte längre frekvent förekommande vid överblick av segment. Det går alltså att fundera över ifall det är mindre frekvent förekommande att göra uppbyggnadsvariationer baserat på spelnamn än på ord från vardagligt språk.

Egennamn är mest vanligt förekommande. Det överensstämmer med resultat från tidigare arbeten, så som Veras et al [25] påvisat. Resterande semantiska kategorier bär indikationer på en relation till intresseområdet tillhörande databasdumpen, så som uppmaningar om att vinna i datorspel för ett spelsällskap, eller hela och terapi för ett sällskap med inriktning mot ett hälsoområde.

7 Slutsats

En modern dator bearbetar enorma antal kombinationer varje sekund och datorerna kommer att bli ännu snabbare i framtiden. Det är därför viktigt att skapa ett lösenord som inte är lätt knäcks. Framförallt handlar det om att användare väljer ett lösenord som är oförväntat och därför inte med på en ordlista. Lösenordsrymden ska åtminstone vara så stor att lösenordet blir svårt att knäcka med brute force knäckning.

En större lösenordsrymd innebär att det finns flera möjliga lösenordskombinationer. En ökning i lösenordslängd resulterar i betydligt fler möjliga lösenord än en motsvarande ökning i ett lösenords alfabet, se Figur 3. Den tid som krävs för att knäcka vissa lösenord finns i Tabell 6. Den visar att ju större en lösenordsrymd är desto längre tid krävs för att knäcka ett lösenord.

Lösenordsparadoxen visar att för korta lösenord där lösenorden är jämnt fördelade med avseende på tecknen i alfabetet som de använder, så är risken för att användare använder samma lösenord hög redan vid små mängder användare. I och med att exempelvis ett lösenord med åtta tecken bestående av versaler, gemener och siffror inte närmar en krock förrän vid 25 miljoner användare, så skulle eventuellt detta problem kunna undvikas, genom att lösenorden är jämnt fördelade med avseende på tecknen i alfabetet som de använder.

Det finns dock mönster som indikerar att data skiljs åt mellan intresseindelningar i arbetets testdata. Om det beror på ålder hos användare, eller undermedveten påverkan av tjänsten de registrerat ett lösenord för, är i detta stadie egalt. Det för detta arbetes målformulering viktiga är huruvida de är likartade eller ej. Författarna av detta arbete vill mena att resultatet ger en antydan om att det finns ett intresse i att vidare göra bredare undersökningar på ifall detta mönster upprepas. Detta eftersom en förståelse för de influenser som påverkar ett lösenordsval ytterligare kan stärka säkerheten runtomkring autentiseringsmetoden.

Resultatet visar även samma indikationer som Veras et al [25] har upplevt, med att pojk och flicknamn är den överlägset vanligaste kategorien. Det styrker tidigare arbeten som visar effektiviteten av ordlistor över brute force knäckning.

En slutsats är att den kanske allra farligaste svagheten vid val av lösenord är den preferensbildade spridningen som uppstår, där användare väljer mönsterbaserade lösenord. Detta begränsar lösenordsrymden till strax över antalet ord i

en ordbok, och enkla sekvenser att knappa in på tangentbordet. Dock är antydan om att det finns ytterligare möjlighet till anpassade brute focre verktyg baserat på användarintressen eller applikationsinriktning en intressant vidareutveckling.

7.1 Framtida arbete

7.1.1 Lösenordsparadoxen

Att beräkna sannolikheten för en krock vid större teckenlängder skulle kunna expanderas med en studie av hur de matematiska funktionerna anropar processorinstruktioner. Detta i syfte att optimera kod med eventuella algoritmer för att klara av att beräkna betydligt större tal. Det är också intressant att se över ifall födelsedagsparadoxen skulle kunna tänkas ha en betydelse vid hashkollisioner.

7.1.2 Semantiska mönster

Det finns intressanta indikationer i resultaten av semantikdelen av detta arbete. För att vikta hur värdefulla dessa resultat är skulle en studie på allt större data mängder kunna utföras. Det finns ett värde i att undersöka ifall intresserelaterade lösenord blir mindre tydligt förekommande i större databaser (miljontals användare).

8 Referenser

Referenser

[1] B Brumen och A Cernezel. "Brute force analysis of PsychoPass-generated Passwords". I: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE. 2014, s. 1366–1371.

[2] X de Carné de Carnavalet och Mohammad Mannan. "From very weak to very strong: Analyzing password-strength meters". I: *Proceedings of the Network and Distributed System Security Symposium*. 2014.

[3] Joseph A Cazier och B Dawn Medlin. "Password security: An empirical investigation into e-commerce passwords and their crack times". I: *Information Systems Security* 15.6 (2006), s. 45–55.

[4] Hsien-Cheng Chou, Hung-Chang Lee, Hwan-Jeu Yu, Fei-Pei Lai, Kuo-Hsuan Huang och Chih-Wen Hsueh. "Password cracking based on learned patterns from disclosed passwords". I: *Int. Journal of Innovative Computing, Information and Control* 9.2 (2013).

[5] Sadie Creese, Duncan Hodges, Sue Jamison-Powell och Monica Whitty. "Relationships between password choices, perceptions of risk and security expertise". I: *Human Aspects of Information Security, Privacy, and Trust*. Springer, 2013, s. 80–89.

[6] Serge Egelman, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov och Cormac Herley. "Does my password go up to eleven?: the impact of password meters on password selection". I: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, s. 2379–2388.

[7] Eveline Flamand. "Deciphering L33t5p34k Internet Slang on Message Boards". Diss. Ghent University, 2008.

[8] Dinei Florencio och Cormac Herley. "A large-scale study of web password habits". I: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, s. 657–666.

[9] Dinei Florêncio och Cormac Herley. “Where do security policies come from?” I: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM. 2010, s. 10.

[10] T Granlund. *GMP: GNU Triangulated Surface library*. `/https://gmplib.org/manual/`. 1993–2014.

[11] Markus Jakobsson och Mayank Dhiman. “The benefits of understanding passwords”. I: *Mobile Authentication*. Springer, 2013, s. 5–24.

[12] Cheng-Chi Lee, Chia-Hsin Liu och Min-Shiang Hwang. “Guessing Attacks on Strong-Password Authentication Protocol.” I: *IJ Network Security* 15.1 (2013), s. 64–67.

[13] Karen D Loch, Houston H Carr och Merrill E Warkentin. “Threats to information systems: today’s reality, yesterday’s understanding”. I: *Mis Quarterly* (1992), s. 173–186.

[14] BK Marshall. “Password research study.” I: *Infosecurity Europe 2003 information security survey* (2003).

[15] Andrew McAfee och Erik Brynjolfsson. “Big data: the management revolution.” I: *Harvard business review* 90 (2012), s. 60–6.

[16] Gordon E Moore. “Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff.” I: *Solid-State Circuits Society Newsletter, IEEE* 11.5 (2006), s. 33–35.

[17] Robert Morris och Ken Thompson. “Password security: A case history”. I: *Communications of the ACM* 22.11 (1979), s. 594–597.

[18] Gilbert Notoatmodjo. “Exploring the ‘Weakest Link’: A Study of Personal Password Security”. Diss. Citeseer, 2007.

[19] Marcus K Rogers och Kate Seigfried. “The future of computer forensics: a needs analysis survey”. I: *Computers & Security* 23.1 (2004), s. 12–16.

[20] Martina Angela Sasse, Sacha Brostoff och Dirk Weirich. “Transforming the ‘weakest link’—a human/computer interaction approach to usable and effective security”. I: *BT technology journal* 19.3 (2001), s. 122–131.

[21] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa och Koji Toyota. “Birthday paradox for multi-collisions”. I: *Information Security and Cryptology–ICISC 2006*. Springer, 2006, s. 29–40.

[22] Leona Tam, Myron Glassman och Mark Vandenwauver. “The psychology of password management: a tradeoff between security and convenience”. I: *Behaviour & Information Technology* 29.3 (2010), s. 233–244.

[23] Harshal Tupsamudre, Vijayanand Banahatti och Sachin Lodha. “Passwords - Divided they Stand, United they Fall”. I: *Int. conference on PASSWORDS, Trondheim, Norway*. 2014.

[24] Wu Tzong-Chen och Sung Hung-Sung. “Authenticating passwords over an insecure channel”. I: *Computers & Security* 15.5 (1996), s. 431–439.

[25] Rafael Veras, Christopher Collins och Julie Thorpe. “On the semantic patterns of passwords and their security impact”. I: *Network and Distributed System Security Symposium (NDSS'14)*. 2014.

[26] Leah Zhang-Kennedy, Sonia Chiasson och Robert Biddle. “Password advice shouldn't be boring: Visualizing password guessing attacks”. I: *eCrime Researchers Summit (eCRS), 2013*. IEEE. 2013, s. 1–11.

[27] Moshe Zviran och William J Haga. “Cognitive passwords: The key to easy access control”. I: *Computers & Security* 9.8 (1990), s. 723–736.

Appendix

Appendix A - Lösenordsparadox programmeringskod

```
/* sample.c - Password Paradox Experiment 1 */
/* Detta script utnyttjar gmp-biblioteket för att beräkna sannolikheten för att
/* givet att sannolikheten för val av ett tillgängligt lösenord är identisk. */
#include<stdio.h>
#include<math.h>
#include<gmp.h>

int main(void) {

  // The values for the password space being calculated
  int bitsArr[8] = {6, 8, 10, 12, 16, 20, 24};
  int symbolsArr[4] = {10, 68, 101, 120};
  int noOfBits = 5;
  int noOfSymbols = 3;
  int bitPos = 0;
  int symbolPos = 0;

  //unsigned long long int nLoop = 0;
  unsigned long long int passwordspaceLoop = 0;
  int areWeDone = 0;

  unsigned long long int nAdder = 0;
  int bigIntTest = 0;
  int bigIntLoop = 0;
  int searchingThreshold = 1;



  int uglyStepSolution = 0;

  // Register GMP variables
  mpf_t x, y, passwordspace, passwordspaceMinusOne, passwordspaceSqrt, result, n
  mpf_init2 (y, 256);
```

```
mpf_init2 (x, 256);
mpf_init(result);
mpf_init(passwordspace);
mpf_init(passwordspaceSqrt);
mpf_init(passwordspaceMinusOne);
mpf_init(final);
mpf_init(toThePower);
mpf_init(n);
mpf_init(nMinusOne);
mpf_init(nMultiplier);
mpf_init(nLoop);
mpf_init(nStep);
mpf_init(almost);
mpf_init(bigIntPlaceholder);
mpf_init(toThePowerDiv);
mpf_init(uglyStepTest);


for(bitPos = 0; bitPos < noOfBits; bitPos++)
{
  for(symbolPos = 0; symbolPos < noOfSymbols; symbolPos++)
  {
    mpf_set_si(x, bitsArr[bitPos]); //bit length
    mpf_set_si(y, symbolsArr[symbolPos]); //symboler i alfabetet

    // passwordspace = symboler ^ bit length
    mpf_pow_ui(passwordspace, y, bitsArr[bitPos]);
    gmp_printf("==========================================\n"
               "Testar %.0Ff ^ %.0Ff    \n"
               "==========================================\n"
               "Lösenordsrymd        %.0Ff                  \n"
               //"------------------------------------------\n"
               , y, x, passwordspace);

    // passwordspace minus 1
```

```
mpf_sub_ui(passwordspaceMinusOne, passwordspace, 1);

// result = chansen att misslyckas
mpf_div(result, passwordspaceMinusOne, passwordspace);

//mpf_set_str(nLoop, "10027200006", 10); //loopens startvärde 30%~ 10^101
// 17613606006 = 75%
//mpf_set_str(nLoop, "17613606006 ", 10); //loopens startvärde 75% 10^101
mpf_set_str(nLoop, "17613606006 ", 10); //loopens startvärde
mpf_set_str(nStep, "102519000", 10); //stega med
gmp_printf("-------------------------------------------------\n"
           "Börja vid           %.0Ff \n"
           "Stega med           %.0Ff \n"
           "-------------------------------------------------\n", nLoop, nStep);
searchingThreshold = 1;
while(searchingThreshold == 1)
{
  mpf_set(n, nLoop); //hämta det n-värde vi använder i denna iteration
  mpf_sub_ui(nMinusOne, nLoop, 1); //nvärde minus 1

  mpf_mul(toThePowerDiv, n, nMinusOne); //detta är n*(n-1)

  mpf_div_ui(toThePower, toThePowerDiv, 2); //detta är "delat på två-delen
  gmp_printf("Vid antal n %.0Ff", nLoop); //debug output antal n

  bigIntTest = mpf_cmp_d(toThePower, 4294967294);
  //4,294,967,295
  if(bigIntTest == 1) {
    //printf("Större än unsigned long int\n"); //debug
    bigIntLoop = 1;
    mpf_set_ui(almost, 1);
    while(bigIntLoop == 1) {
      if(mpf_cmp_d(toThePower, 4294967294) == 1) {
        mpf_pow_ui(bigIntPlaceholder, result, 4294967294);
        mpf_mul(almost, bigIntPlaceholder, almost);
```

```
            mpf_sub_ui(toThePower, toThePower, 4294967294);
          } else {
            mpf_pow_ui(bigIntPlaceholder, result, mpf_get_ui(toThePower));
            mpf_mul(almost, bigIntPlaceholder, almost);
            //printf("reached end\n"); //debug
            bigIntLoop = 0;
            break;
          }
        }

      } else {
        mpf_pow_ui(almost, result, mpf_get_ui(toThePower));
//detta ska eventuellt exponentieras flera gånger
      }

      mpf_ui_sub(final, 1, almost);
      areWeDone = mpf_cmp_d(final, 0.99);
      gmp_printf("␣med␣sannolikheten␣%Ff\n", final);
      if(areWeDone == 1)
      {
        gmp_printf(
                   "Antal␣lösenord␣␣␣␣␣␣%.0Ff\n"
                   "Exponent␣␣␣␣␣␣␣␣␣␣␣␣%.0Ff\n"
                   "Sannolikhet␣krock␣␣␣%Ff\n",
                   n, toThePower, final);
        if(bigIntTest == 1) {
          printf("OBS:␣Exponenten␣större␣än␣unsigned␣long␣int\n\n");
        } else {
          printf("\n");
        }
        searchingThreshold = 0;
        break;
      }
      else
      {
```

```
            mpf_add(nLoop, nLoop, nStep); //öka loopen gmp-funktion
          }

        }
      }
    }
  mpf_clear(x);
  mpf_clear(y);
  mpf_clear(passwordspace);
  mpf_clear(passwordspaceMinusOne);
  mpf_clear(result);
  return 0;
}
```

Joakim Lindell

Mark Crossley

Besöksadress: Kristian IV:s väg 3
Postadress: Box 823, 301 18 Halmstad
Telefon: 035-16 71 00
E-mail: registrator@hh.se
www.hh.se