Konsten att välja ett säkert lösenord: En komparativ analys av lösenordsknäckningsmetoder

1. Introduktion

Lösenords-baserad autentisering är den mest använda autentiseringsmetoden på internet idag [1]. Mängden av lösenord som majoriteten av vanliga datoranvändare måste komma ihåg ökar med antalet användarkonton som de tvingas skapa för olika tjänster, såsom molnlagring, e-post och sociala medier [1]. Eftersom användaren måste hålla reda på flera olika lösenord, återanvänds samma lösenord ofta eller så väljer användarna svaga lösenord som är lätta att komma ihåg [1], [2]. Informationsläckage är väldigt vanliga i dagens läge, år 2022 visade det sig att mera än 300 miljoner användarkonton läckts [3]. För att skydda lösenord i fall av läckage används en hashfunktion för att kryptera lösenorden och bara det krypterade lösenordet sparas i databasen. En hashfunktion är en kryptografisk funktion som tar som indata en godtyckligt lång sträng och producerar en slumpmässig sträng av tecken med fast längd som kallas till en hash.

En hashfunktion producerar alltid samma hash från samma indata, vilket öppnar upp möjligheten att knäcka hashade lösenord genom att hasha olika lösenordsgissningar och jämföra dem med hasharna i databasen. En sådan attack kallas lösenordsknäckning. Det finns flera olika metoder för lösenordshashning och lösenordsknäckning med sina styrkor och svagheter. Denna kandidatavhandling syftar till att analysera lösenord strukturer samt hashfunktioner för lösenordshashning i några av de mest använda verktygen för att skapa webbsidor, nämligen webbramverk och CMS, för att undersöka hur man som både utvecklare och användare kan skydda lösenord mot lösenordsknäckning. För att bedöma säkerheten hos hash funktionerna och lösenords strukturerna kommer tre olika lösenordsknäckningsmetoder att användas.

Lösenordshashning innebär att istället för att spara lösenord i klartext, d.v.s. utan kryptering så körs lösenorden igenom en s.k. hashfunktion som producerar en slumpmässig krypterad output som kallas till en hash [4]. En hashfunktion tar som indata en godtyckligt lång sträng och ger som utdata en sträng av fast längd, bestående av slumpmässiga bitar [4]. Hashfunktioner är s.k. “En-vägs” kryptografiska funktioner [4]. Detta betyder att man inte kan återställa den originella strängen från dess hash, d.v.s. hashfunktioner fungerar bara åt ett håll [4]. En viktig egenskap av hashfunktioner är att samma input alltid producerar samma output [4]. Denna egenskap gör det möjligt att istället för att spara lösenordet i klartext i en databas så sparar man lösenordets hash istället [4]. Då en användare skapar ett lösenord för en webb service, så körs lösenordet igenom en hashfunktion och bara lösenordets hash sparas i servicens databas [4]. Då användaren loggar in med sitt lösenord så körs lösenordet igenom samma hashfunktion och sedan jämförs denna hash med den som finns lagrad i databasen [4]. Om lösenordet är samma som det som användarkontot skapats med så kommer också hasharna att vara samma och på detta sätt autentiseras användaren med sitt lösenord [4]. Lösenordshashning kan dock inte alltid stoppa lösenord från att läckas.

Lösenordsknäckning är en attack där man försöker gissa en användares lösenord [5]. Lösenordsknäckning kan ske både “online” och “offline” [5]. En online attack går ut på att försöka logga in på en användares konto genom att försöka gissa lösenordet [5]. I en sådan attack används oftast osäkra och vanligt förekommande lösenord som t.ex. “Password123” [5]. Online attacker är ofta mycket ineffektiva eftersom kontots användarnamn eller IP-adressen av personen

som utför attacken kan låsas ut från servisen efter för många fel gissningar [5]. I en offline attack har man tillgång till en databas av lösenordshashar och försöker istället att knäcka hasharna [5]. Eftersom en hashfunktion alltid genererar samma hash från samma input sträng så kan man försöka gissa lösenord från en lista av lösenordshashar genom att hasha gissade lösenord med samma hashfunktion som lösenordshasharna genererats med och sedan jämföra de gissade hasharna med lösenordshasharna för att hitta lösenord [5]. I denna avhandling analyseras endast offline attacker, p.g.a. att i en online attack har man oftast en begränsad mängd försök och det är ofta väldigt ineffektivt och producerar sällan resultat. Det finns flera olika metoder för lösenordsknäckning, nämligen uttömmande attack, ordliste attack och regnbågstabell attack.

I en uttömmande attack går man igenom alla möjliga kombinationer av karaktärer för att knäcka lösenordshashar [5]. En uttömmande attack kan teoretiskt sett knäcka alla möjliga lösenordshashar men i praktiken är det mycket långsamt eftersom det skulle kräva så stora mängder beräkningskraft och tid. I en ordliste attack däremot, använder man sig av en lista bestående ofta förekommande lösenord som ofta baserar sig på läckta lösenord [5]. Ordliste attacker kan delas upp i fyra olika kategorier, nämligen ren, Probabilistic Context Free Grammar (PCFG) baserad, Markov model baserad och mangling regler. En ren uttömmande attack använder sig endast av en färdig lista av vanliga lösenord. PCFG-baserade attacken uttnytjar PCFG-teorier för att generera en lista på modifierade lösenord, rangordnade på basen av sannolikheten av deras förekomst. I den Markov model baserade attacken skapas en lista av lösenord baserat på sannolikhetsfördelningen av karaktär sekvenser genom att använda Markov baserade modeller. I en attack som använder sig av mangling regler skapar man en lista av lösenord genom att plocka ord från en ordlista och modifiera dem med olika regler som t.ex. “sätt karaktären + i slutet av lösenordet”. Regnbågstabell attacker använder sig av en lista på färdigt hashade lösenord som kallas till en regnbågstabell. Regnbågstabell attacken snabbar upp lösenordsknäckningen eftersom lösenordshasharna är färdigt kalkylerade före attacken vilket minskar på den behövda beräkningskraften. Det finns dock en skyddsmekanism som gör regnbågstabeller oanvändbara, nämligen salt. Salt är en slumpmässig karaktär sträng som läggs till lösenordet vid hashning. Eftersom det är omöjligt för en angripare att veta vad salten är så kan en regnbågstabell skapas.

2. Hashfunktioner i de vanligaste webbramverk och CMS

2.1 MD5

3. Lösenordsknäckning

3.1 Uttömmande attack

En uttömmande lösenordsknäcknings attack utförs genom att pröva alla möjliga kombinationer av tecken för att producera lösenordsgissningar. I en uttömmande utförs med beaktande av två parameter värden, nämligen teckenkod och lösenordslängd. Teoretiskt sätt kan en uttömmande attack knäcka alla möjliga lösenord, den enda begränsande faktorn är tid. För att knäcka ett lösenord som består av 5 tecken som kan vara stora eller små bokstäver [a-z, A-Z], siffror och specialtecken som kan befinna sig i vilket position som helst i lösenordet, krävs det att $94^5 = 7\ 339\ 040\ 224$ lösenordsgissningar beräknas. För ett lösenord av längden 9 blir antalet lösenordsgissningar som måste beräknas $94^9 = 572\ 994\ 802\ 228\ 616\ 704$. Varje tecken som läggs till lösenordet gör sökutrymmet 94 gånger större. Därför är uttömmande attacker endast effektiva mot korta och enkla lösenord.

3.2 Ordliste attack

I en ordliste attack (Dictionary attack), används en ordlista för att producera lösenordsgissningar [6]. En ordlista består ofta av vanliga lösenord och lösenord från informations läckage, vanliga för- och efternamn, namn på idrottslag, bilmärken, o.s.v. En populär ordlista är rockyou.txt som består av miljontals lösenord som läcktes från företaget RockYou. I en ren ordliste attack kollar man upp lösenordsgissningar från en ordlista, hashar dem och jämför dem med lösenordshashar för att hitta lösenord. Fast än rena ordliste attacker är mycket simpla, är de mycket effektiva eftersom användare har en tendens att välja enkla och lätt

förutsägbara lösenord. För att knäcka mer unika lösenord kan olika metoder användas för att skapa en modifierad ordlista. Mangling regler, PCFG teorier och Markov modeller kan användas för att skapa sådana modifierade ordlistor.

3.2.1 Mangling regler

I en regelbaserad ordliste attack modifieras en ren ordlista med hjälp av olika regler för att producera nya lösenordsgissningar [7]. Några populära regler är t.ex. att byta ut vissa bokstäver som ‘s’ och ‘a’ mot specialtecken som ‘$’ och ‘@’, byta ut första bokstaven till en stor bokstav eller att skriva om lösenordet baklänges. Om lösenordet “password” finns i en ordlista så kan man genom att modifiera ordlistan med regler också hitta lösenordet “p@$$word” och på detta sätt knäcka en större mängd lösenord. Eftersom flera tjänster förhindrar användaren från att skapa lösenord utan en kombination av specialtecken, siffror och stora och små bokstäver, modifierar användaren ofta sina lösenord så att de ändå är lätta att komma ihåg, t.e.x. “password” => “P@$$w0rd”. Då regler används för att modifiera en ordlista ökar också ordlistans storlek och samtidigt blir attacken långsammare. Om en ordlista med 100 lösenordsgissnar modifieras genom att lägga till två heltal [0-9] i slutet av lösenorden, produceras 100 unika variationer av varje lösenord i ordlistan och ordlistans storlek ökar från 100 till 10 000. Om denna regel kombineras med en anna regel som som byter ut stora bokstäver mot små och små mot stora så blir ordlistans storlek dubbelt större. För att utföra en effektiv regelbaserad ordliste attack är det viktigt att välja regler som producerar största graden av lyckade lösenordsgissningar för den minsta mängden lösenordsgissningar per lösenord i ordlistan.

3.2.2 PCFG

Ett sätt att effektivt producera regler är användingen av Probabilistic Context Free Grammars (PCFG). PCFG används inom språkteknik för att analysera och generera teckensträngar med en bestämd struktur och kan därför också användas för att producera lösenordsgissningar. Context free grammars består av icke-terminaler (variabler), terminaler

(tecken), start variabeln **S** och produktioner av formen **a -> b**, där **a** är en icke-terminal och **b** är en sträng av icke-terminaler och terminaler. PCFG rangordnar produktionerna enligt sannolikheten av deras förekomst. För att generera produktioner används en lista på lösenord som träningsdata. Från träningsdatat härleds produktioner som producerar basstrukturer, bestående av variabler och produktioner som bildar tecken. För att representera variabler används L_n för bokstäver, D_n för siffror, S_n för specialtecken och S för startsymbol, där n anger mängden av tecken i variabeln, t.ex. lösenordet "pass123!" har strukturen $L_4D_3S_1$. För varje variabel rangordnas dess produktioner enligt sannolikheten av deras förekomst. Ett simpelt exempel synns i tabell 1. Produktioner sker enligt följande $S \rightarrow L_4D_3S_1 \rightarrow L_4123S_1 \rightarrow L_4123!$, där L_4 fylls i från en ordlista. PCFG används alltså för att skapa regler för att modifiera en ordlista. Om vi har strukturen $L_4123!$ och en ordlista som består av ["pass", "john", "bart"] så produceras lösenordsgissningarna "pass123!", "john123!" och "bart123!".

Variabel	**Produktion**	**Sannolikhet**
$S \rightarrow$	$L_4D_3S_1$	0.60
$S \rightarrow$	$D_1L_3S_1$	0.40
$D_3 \rightarrow$	123	0.80
$D_3 \rightarrow$	456	0.20
$D_1 \rightarrow$	1	0.82
$D_1 \rightarrow$	4	0.18
$S_1 \rightarrow$	!	0.55
$S_1 \rightarrow$	+	0.45

3.2.3 Markov modell

Källförteckning

[1] A. Kanta, I. Coisel, and M. Scanlon, “A Novel Dictionary Generation Methodology for Contextual-Based Password Cracking,” *IEEE Access*, vol. 10, pp. 59178–59188, 2022, doi: 10.1109/ACCESS.2022.3179701.

[2] S. Ji, S. Yang, X. Hu, W. Han, Z. Li, and R. Beyah, “Zero-Sum Password Cracking Game: A Large-Scale Empirical Study on the Crackability, Correlation, and Security of Passwords,” *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 5, pp. 550–564, Sep. 2017, doi: 10.1109/TDSC.2015.2481884.

[3] Surfshark, “Data breach statistics 2021 vs. 2022,” Jan. 18, 2023. https://surfshark.com/blog/data-breach-recap-2022

[4] D. Wong, *Real-world cryptography*. Shelter Island, NY: Manning Publications Co, 2021.

[5] C. Ntantogian, S. Malliaros, and C. Xenakis, “Evaluation of password hashing schemes in open source web platforms,” *Comput. Secur.*, vol. 84, pp. 206–224, Jul. 2019, doi: 10.1016/j.cose.2019.03.011.

[6] L. Bosnjak, J. Sres, and B. Brumen, “Brute-force and dictionary attack on hashed real-world passwords,” in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, May 2018, pp. 1161–1166. doi: 10.23919/MIPRO.2018.8400211.

[7] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, “Password Cracking Using Probabilistic Context-Free Grammars,” in *2009 30th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2009, pp. 391–405. doi: 10.1109/SP.2009.8.