

Password cracking 1

cracking cryptography

Challenge status: Available

Challenge flags: 6

V této úvodní úloze k prolamování hesel se studenti naučí různé druhy útoků pomocí nástroje hashcat a vyzkouší si prolomit široce používané hashovací algoritmy.

Hashování

Hashování je jednosměrná matematická operace, která pro libovolně dlouhý vstupní text vytvoří výstupní text o fixní délce. Nemusí jít pouze o textový řetězec, mohou být použita jakákoli jiná data. Pokud se nesnažíme proniknout přímo do matematiky algoritmů, tak to v zásadě není nic složitého.

Například heslo "ahoj123" zahashované pomocí algoritmu sha256 vytvoří hash:

`f18e8a90bd0f6267a733d0036f3349dcf4f3f68556ce143084b35fe0d70b6044`

Tento hash má 64 znaků a vždy mít bude, nehledě na to, jak dlouhý text mu na vstup dáme. Ale i drobná změna vstupního textu vytvoří kompletně odlišný hash.

Takže když zahashujeme heslo "ahoj1234" pomocí stejného algoritmu, tak vznikne opět hash o délce 64 znaků, ale zcela jiný.

`326851a67389beec6d7ef48bc51b6143e29f8c51c9fb46899f9242b42a032a9f`

Tyto operace jsou jednosměrné a pro stejný vstup vytvoří vždy stejný výstup. Při vytvoření nového účtu zadáte heslo, stránka ho

Theory

Flags

Handbook

Solves

Connect

Ověřte si, že se porovná s hashem uloženým v databázích.

Jak funguje prolamování hesel

Jak jsme si už řekli, hashování je jednosměrná matematická operace, takže z hashe nelze žádným způsobem, pomocí žádných výpočtu získat zpět původní heslo. Jak se tedy hesla crackují?

Využívá se výše zmíněné vlastnosti, že pro konkrétní vstup má hash vždy stejnou podobu, takže pokud zahashujeme heslo "ahoj123", dostaneme:

**f18e8a90bd0f6267a733d0036f3349dcf4f3f68556ce143084b35f
e0d70b6044**

Pokud heslo "ahoj123" zahashujeme znovu, bude mít opět stejný hash:

**f18e8a90bd0f6267a733d0036f3349dcf4f3f68556ce143084b35f
e0d70b6044**. Takže je jasné, že hesla crackujeme tak, že hashujeme hesla, o nichž si myslíme, že je mohl uživatel použít. Pak jen porovnáváme naše hashe s těmi, které chceme prolomit. A už se roztačí spirála obran a útoků mezi uživateli a hackery. Tohle by to totiž znamenalo, že ve většině případů stačí hackerovi znát uživatelovo jméno, přezdívku, mazlíčka, oblíbený sportovní klub... a může prolomit jeho heslo. Proto bezpečnostní experti hackerům museli jejich činnost trochu osolit, opepřít a jinak zavařit.

Salt

Salt je řetězec náhodných znaků, který systém vygeneruje a následně umístí před uživatelovo heslo. Zahashuje pak celý nově vytvořený řetězec. Po zahashování se salt zapíše před hash:

**VsCKEMNGvjDyksH4.59a8aafb1e9e75c25eb2e8a9c936b1d822c82
b6e6c1a842455a657331ffdb950**. Systém u stejného uživatele přidá vždy stejný salt. Uživatel ho samozřejmě nezná. Salt je zcela náhodný, tudíž se jeho přidáním zvýší jedinečnost hesla a zlepšuje se jeho zabezpečení.

V případě, že by dva různí uživatelé použili shodné heslo, systém podle emailové adresy, uživatelského jména nebo jiného identifikátoru pozná, o koho se jedná a při přihlašování použije jemu přiřazený salt. Tudíž výsledkem bude, že dvě identická hesla různých uživatelů budou mít v systému uložený rozdílný hash.



Pepper

Je velmi podobný saltu a funguje na stejném principu, ale s tím rozdílem, že není uložený u hashe, ale zcela jinde. Zatímco je salt spolu se zahashovaným heslem uložený v databázi hesel, pepper je uschován odděleně, aby nebyla zjevná jeho souvislost s heslem. Často je uložen v hardware security modulu. Aby i pepper odolal prolomení hrubou silou (bruteforce), doporučuje se jeho velikost alespoň 112 bitů. Pepper se někdy nazývá secret salt.

Iterace

Je další způsob, jak zvýšit složitost hesel. Jedná se o vkládání hashovacích funkcí do sebe. Výstup funkce vnořené se použije jako vstup jí nadřazené funkce, takže například řetězec:
`sha256(sha256(sha256(sha256(sha256(hash)))))`. Heslo je zahashováno několikrát po sobě.

Jak identifikovat hash

Představte si, že jste třeba na soutěži CTF nebo při pentestování nějaké firmy dostali třeba:

`7e2677c4c788b2ca0c378c4824dd4928e5fcdd43b504050e2eaec22984762c78`. Co s tím? Prvním krokem je určit, o jaký hash se jedná. Přesněji, jaký algoritmus byl použit pro jeho vytvoření.

K tomu můžete použít například software hashid nebo hash-identifier. Oba programy jsou předinstalovány v Kali-linuxu. Pokud zrovna nemáte přístup k těmto programům můžete použít nějaký online nástroj. Stačí do vašeho vyhledávače zadat "hash identifier" a určitě narazíte na nějakou užitečnou stránku.

Nejpopulárnější hashe jsou:

- md5 (zastaralý)
- sha256
- NTLM (windows hesla)

Způsoby crackování

OK, identifikovali jste algoritmus, už víte, o jaký druh hashe se jedná. Teď zbývá jen ho rychle prolomit. Pro nejrychlejší prolomení je potřeba použít správný typ útoku. Každý způsob má své výhody i své nevýhody. Vše závisí na situaci.

Mask attack

Také se občas označuje jako bruteforce attack. Jedná se o útok, kde se zkouší všechny možné kombinace definované zadanou maskou. Zadáme masku a program si již generuje možné kandidáty na heslo. Pointa je, že znáte masku. Pokud víte, že heslo je pouze z písmen, můžete definovat masku, která generuje náhodné kombinace zvolených znaků.

Nebo víte, že heslo bude z malých písmen plus bude obsahovat čísla nebo víte, že to bude komplexnější maska, že první 4 znaky budou velká nebo malá písmena, další 2 znaky budou čísla a poslední znak bude nějaký speciální znak. Tj. definujete masku, která pak za vás zkouší všechny odpovídající kombinace.

- ?l = abcdefghijklmnopqrstuvwxyz
- ?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
- ?d = 0123456789
- ?h = 0123456789abcdef
- ?H = 0123456789ABCDEF
- ?s = «space»!"#\$%&'()*+,.-/:<=>?@[]^_`{}~
- ?a = ?l?u?d?s
- ?b = 0x00 - 0xff

Výhodou tohoto útoku je, že systematicky vyzkouší všechny možnosti, takže pokud máte dostatečně rychlý hashovací algoritmus a dostatečně krátké heslo, tak je bruteforcnutí hesla relativně dobrá cesta.

Nejčastěji by se tento druh útoku použil na náhodně generovaná hesla, která nemají žádnou podobu se slovy, nebo na hesla, která se skládají pouze z čísel. Většinou však hesla nejsou

náhodné řetězce znaků, ale tvoří nějaké slovo, blízké tomu, kdo ho vytvořil. Typický uživatel pro zvýšení neprolomitelnosti hesla, nebo kvůli požadavku IT technika přidává pář čísel na konec. Zde pak přichází ke slovu Dictionary attack a Rule attack.

Dictionary attack

Je to nejfektivnější útok při crackování běžných hesel. Používá se slovník (dictionary) již prolomených hesel v daném jazyku. (Použít slovník s anglickými hesly nebude moc efektivní na prolomení českých hesel.)

Rule attack

Rule neboli pravidlo je nějak definované upravení původního hesla, takže z hesla "ahoj" můžeme pomocí pravidel vytvořit například "Ahoj", "ahoj123", "ahojahoj", "AHOJ007" a další variace. Více pravidel pohromadě se nazývá ruleset.

Kombinace dictionary attacku a rulesetu je pravděpodobně nejfektivnější útok na velké databáze. Nejrychleji prolomíte nejvíce hesel. Avšak to, že s tímto útokem budete mít 100% úspěch je velice nepravděpodobné.

Pokud jde o prolomení jednoho specifického hesla, můžete zvolit taktiku, kde seženete co nejvíce údajů o oběti a vytvoříte vlastní slovník se slovy, která by mohla být potenciální hesla a aplikujete na tento malý slovník hodně velký ruleset. Některé rulesety obsahují až statisíce pravidel, což znamená, že každé heslo v slovníku se ještě sto tisíckrát upraví a vyzkouší se všechny jeho varianty.

Více informací se dá najít zde: www.hashcat.net/wiki/doku.php?id=rule_based_attack

Jak vznikají slovníky a kde je sehnat

Na internetu se stále objevují ukradené nebo leaknuté databáze zahashovaných hesel. Někdo tato hesla prolomí a z velkých objemů dat se dá vytvářet slovník nejčastějších hesel. Je však nutné brát v potaz, že slovníky budou nejčastěji obsahovat anglická hesla. S trochou hledání se však dají sehnat i slovníky nejčastějších hesel pro daný jazyk i pro češtinu. Za nejznámější wordlist by se dal považovat "rockyou".

Rainbow tables

Jedná se o soupis hesel a jejich hashů pro daný algoritmus. To je velké zrychlení při prolamování hesla, protože každé potenciální heslo nemusíte hashovat. Stačí pouze v rainbowtables najít jeho hash a podívat se, jaké je heslo v plaintextu. To všechno sice zní úžasně, vypadá to, že stačí, aby jednou někdo s velkým výpočetním výkonem vytvořil velký rainbowtable a žádné heslo nebude nikdy v bezpečí. Ale opak je pravdou.

Salt a pepper eliminují rainbow table attack z toho důvodu, že stejná hesla mají vždy jiný hash právě kvůli saltu nebo pepperu.

Další nevýhodou je objem dat. Rainbowtables mohou dosahovat i několik desítek terabytů, ale klidně i mnohem více.

Online rainbowtables

Jedná se o rainbowtables, ale bez nevýhody uložení obrovských objemů dat na svém počítači. Samotné rainbowtables co obsahují obrovská množství hesel v nejpopulárnějších algoritmech jako například sha256, md5, lm, NTLM... se nacházejí v online databázi a jsou přístupné pomocí webové stránky. Třeba www.crackstation.net. Pokud například v soutěži ctf dostanete jako úkol nebo součást úkolu prolomit sha256 nebo md5 hash, tak nejjednodušší a nejrychlejší varianta je zkoumat právě crackstation nebo její alternativy.

Hashcat

Jedná se o nejpopulárnější i nejrychlejší software na crackování hesel, který je ještě k tomu open source. Hlavní výhodou hashcatu je jeho podpora akcelerace pomocí grafické karty nebo více grafických karet. Grafické karty mají oproti procesoru tisíce jader, a proto jsou násobně krát rychlejší pro výpočty jako je například hashování.

Hashcat má za sebou mnoho let vývoje a skrývá v sobě mnoho vlastností, které ho umisťují před ostatní crackovací softwary. Hashcat je multi platform software, takže může běžet na Windows, Linuxu i macOS. Umí crackovat více hashů najednou, klidně i celé databáze. Dovoluje pozastavit crackovací session a následně ho umí znova obnovit na místě kde skončil. Má zabudovanou kontrolu teploty grafických karet, takže pokud teplota karty překročí například 95 °C, tak se hashcat přeruší a progress se uloží, aby se mohl session opět obnovit.

Jak ovládat hashcat

Hashcat nemá grafické rozhraní, je to čistě terminálová aplikace. Proto se ovládá pomocí přepínačů a ty nejpoužívanější si nyní popíšeme. Hashcat má i tu výhodu, že pokud byste si zrovna na nějaký zde popsány pokyn nemohli vzpomenout, tak "hashcat -h" vám vypíše help a všechny přepínače.

- -a [číslo] - určuje typ útoku. Například -a 0 je dictionary attack, -a 3 je mask attack

- -m [číslo] - udává hash mode, místo [číslo] napište číselný kód hashe. Například NTLM je kód 1000.
- -r [soubor s pravidly] - aplikuje pravidla z souboru na každé potenciální heslo

Obecný zápis:

```
hashcat [parametry] [hash|hashfile] [slovník|maska]
```

Příklady konkrétního zápisu:

```
hashcat -m 0 -a 3 md5_hash.txt ?d?d?d?d
```

```
hashcat -m 1000 -a 0 ntlm_hash.txt rockyou.txt
```

Co ovlivňuje rychlosť crackovania hesiel

Rychlosť crackovania hesiel v domácich podmínkach na vašom počítači ovlivňuje hlavně rychlosť grafické karty. Nemusí to být vždy pravidlom, ale čím modernejší a dražší karta, tím bude pravdepodobnejšia výkonnosť pre výpočty. Popričom znásobíte rychlosť použitím viac grafických karet. Rychlosť se zvedá rovnomenne s každou pridanou kartou, takže 4krát viac karet bude mít 4krát vyššiu crackovaciu rychlosť.

Kolize v hashovacích algoritmech

Nepričíma vám, že tu něco malinko nehraje? Jak už víte hash má vždy fixnú dĺžku, takže i keďže je neomezený počet vstupných reťazcov (hesiel), je omezený počet hashov. Z toho plyne, že pre dva rozdielne vstupy bude musieť byť stejný hash. A presne to je kolize. Kolize je nebezpečná, protože zoslabuje silu hashovacieho algoritmu.