

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Práca na tému

Voľne šíriteľné nástroje na rozbitie hesiel

Oliver Leontiev

2021

Predmet: Princípy informačnej bezpečnosti

Obsah

1. Zámer projektu	3
Všeobecný zámer projektu	3
Podmienky experimentu	3
Stručný časový plán	3
2. Úvod do problematiky	4
Sila hesiel	4
3. Hashovanie hesiel	4
Hashovacie algoritmy	4
Solenie hesiel ("salting")	5
4. Analýza metód a algoritmov na zisťovanie hesiel	6
Metódy bez použitia nástrojov na lámanie hesiel	6
Metódy s použitím nástrojov na lámanie hesiel	7
5. Analýza nástrojov na lámanie hesiel	9
John the Ripper	9
Hashcat	12
Subjektívne porovnanie nástrojov hashcat a JtR z pohľadu UX	14
Hydra	15
6. Testovanie výkonu nástrojov	16
Typy testov	16
Metriky testov	17
Test 1: dictionary attack	17
Test 2: krátke a predvídateľné heslá	18
Test 3: krátke a náhodné heslá.....	19
Test 4: dlhé a predvídateľné heslá	20
Test 5: dlhé a náhodné heslá	21
7. Záver	22
Zdroje	23

1. Zámer projektu

Všeobecný zámer projektu

Daná téma bude spracovaná so zameraním na analýzu troch z najrozšírenejších nástrojov na odhaľovanie, dešifrovanie a lámanie hesiel - **Hashcat**, **John the ripper** a **Hydra**. Experiment bude pozostávať z praktického využitia týchto troch nástrojov v kontrolovanom prostredí, pričom budeme testovať ich efektivitu pri rôznych podmienkach (hashovací algoritmus, zložitosť a sila hesiel, dostupné slovníky hesiel) a porovnávať ich medzi sebou pri podobných úkonoch.

Experimentu bude predchádzať analýza algoritmov na hashovanie hesiel a algoritmov na ich lámanie. Rozobratá bude aj problematika silných a slabých hesiel (s výsledkami z experimentu na podloženie tvrdení) a iné zraniteľnosti pri výbere a používaní hesiel a metódy, ktoré ich zneužívajú (social engineering).

Pred experimentom bude opísané a vysvetlené aj ako fungujú samotné nástroje - ako ich používať, aké poskytujú funkcie a v čom sa líšia. Pôjdeme do adekvátnej hĺbky, aby sme pokryli dôležité nastavenia, parametre a typy útokov pre všetky tri nástroje (nebudeme sa venovať úplne všetkým funkciám ani analýze zdrojových kódov).

Podmienky experimentu

Experiment bude prebiehať na **Kali Linux** zariadení spustenom ako virtual machine, z dôvodu, že všetky nástroje sú v základnej výbave Kali Linuxu a pri ich používaní sa budeme snažiť aj nabúrať do druhého zariadenia na sieti cez SSH (najmä v prípade nástroja Hydra).

Pri lámaní hesiel za pomoci slovníkov budeme používať aj vlastné slovníky (krátke, pre rýchlejší a porovnávací charakter experimentu) aj populárne slovníky pre skutočné útoky, napríklad RockYou.txt (pre reprezentatívne experimenty).

Stručný časový plán

Opiera sa o postupnosť týždňov semestra.

3. Týždeň: spustenie Kali Linux VM a zoznámenie sa s nástrojmi.
4. Týždeň: podrobná analýza nástrojov a experimentovanie.
5. Týždeň: prezentácia prvého progress reportu.
- 5-8. Týždeň: experimentovanie.
9. Týždeň: prezentácia druhého progress reportu.

2. Úvod do problematiky

Heslá slúžia na **autentifikáciu** používateľa systému, a teda patria medzi citlivé a cenné informácie. Z toho dôvodu je dôležité ich chrániť pred všetkými druhmi útokov.

Keď používateľ zadá heslo, tak je potrebné ho overiť či je správne, a teda porovnať s heslom uloženým v pamäti systému. Je nutné, aby tieto heslá boli uložené v zahashovanej forme za pomoci **jednosmerného hashovacieho algoritmu** (z hashu sa nedá jednoducho spätne zistiť heslo). Dôvodom je, že záznam o uložených heslách sa môže dostať na verejnosť - napríklad prelomením jedného z hesiel alebo činnosťou autorizovaného aktéra (“**insidera**”).

Iným spôsobom, ako sa môžu heslá dostať mimo systému je tzv. “**sniffing**”. Pokiaľ autentifikácia prebieha cez sieť, tak útočník môže túto komunikáciu zachytiť a prečítať.

Zahashovaním vieme zaručiť, že aj ak útočník úspešne extrahuje heslo zo systému, tak ho stále musí zlomiť porovnávaním s hashom. Takéto lámanie hesiel, kedy sa snažíme zistiť heslá z vopred známych hashov sa nazýva “**offline cracking**” a dá sa pri ňom uplatniť veľa rovnakých metód ako pri priamom nabúravaní sa do systému - generujeme možné heslá a porovnáваме ich hash s hashmi známych hesiel, pričom hľadáme zhodu.

Sila hesiel

Každá automatizovaná metóda lámania hesiel je založená na generovaní možných hesiel a testovaní voči systému (do ktorého sa útočník snaží dostať) alebo hashom známych hesiel. Preto je dôležité aby heslá bolo ťažké uhádnuť - človekom aj nástrojom.

V roku 2020 bolo najpoužívanéjšie heslo “**123456**” [1]. Je potrebné vyhýbať sa heslám, ktoré útočník ľahko uhádne aj bez použitia nástroja, resp. ich dobré nástroje vyskúšajú medzi prvými. Patria sem postupnosti číslíc, heslá obsahujúce celé slová (najmä “password”, “admin” a mená) a postupnosti písmen na klávesnici (“qwerty”, “asdfgh”).

Okrem toho je potreba zabezpečiť heslo aj proti výpočtovej sile softwarových nástrojov. Čím je heslo dlhšie, tým viac možností musí nástroj vyskúšať pri útoku hrubou silou. Zároveň je potreba mať čo najväčší súbor použitých znakov (aj veľké aj malé písmená, čísla a interpunkcia), aby musel nástroj otestovať veľa možností pre každú pozíciu.

3. Hashovanie hesiel

Hashovacie algoritmy

Hashovacie algoritmy sú postupnosti operácií, ktoré zmenia heslo v pôvodnom tvare na inú sekvenciu znakov. Hlavným znakom takéhoto hashovania je, že neexistuje spôsob ako z hashu

získať heslo (spätný algoritmus). Jediné, čo ostáva útočníkovi je zahashovať potenciálne heslá a porovnávať ich s hashom skutočného hesla.

Hashovacie algoritmy neustále čelia dvom hlavným hrozbám:

1. Niektorí nájdu slabinu algoritmu, čo potom dokáže výrazne zefektívniť lámanie hashov. Napríklad algoritmus MD5 je zraniteľný voči **kolíziám** (rôzne heslá produkujú rovnaké hashe).
2. Zahashovanie daným algoritmom musí byť dostatočne zložitá operácia, aby zdržala útočníka pri generovaní veľkého množstva hesiel. Vzhľadom na vznik a dostupnosť neustále výkonnejšieho hardwaru je to veľmi náročné.

Vzhľadom na tieto hrozby vznikajú neustále nové a silnejšie algoritmy, zatiaľ čo staršie upadajú na popularite, lebo prestávajú byť bezpečné. Medzi hashovacie algoritmy patria: **MD5, SHA1, SHA256, PBKDF2, Bcrypt, Scrypt a Argon2**.

Napriek tomu, že SHA1 je silnejší algoritmus ako MD5 (vypočítanie hashu s SHA1 je komplexnejšie a vyžaduje viac času), tak oba boli vyhlásené vládou USA a rôznymi technologickými inštitútmi (CMU, NIST) za nevhodné na ochranu hesiel.

Argon2 je k dnešnému dňu jeden z najsilnejších hashovacích algoritmov. V roku 2015 jeho tvorcovia vyhrali **Password Hashing Competition**. Argon2 má tri verzie: **Argon2d, Argon2i a Argon2id** (hybridná verzia). Každá verzia ponúka zvýšenú ochranu proti istým druhom útokov (napr. Argon2d chráni lepšie pred útokmi s použitím GPU ako Argon2i).

Solenie hesiel (“salting”)

Solenie hesiel znamená rozšírenie používateľského hesla o náhodné dáta – **salt**. Tento salt by mal byť unikátny pre každé heslo a môže byť uložený v databáze spolu s heslom. Bežnými spôsobmi solenia sú (+ je operácia zret'azenia) [8]:

saltedhash(password) = hash(password + salt)

saltedhash(password) = hash(hash(password) + salt)

Solenie má dve hlavné výhody:

1. Chráni pred útokmi s predpripravenými hashmi (rainbow table útoky)

Salt spôsobí, že každé heslo má “vlastný hashovací algoritmus” - pôvodný rozšírený o jeden triviálny krok. Predpripravené hashe sú platné iba pre jeden konkrétny hashovací algoritmus. Z toho vyplýva, že útočník by musel predpripraviť hashe pre každý možný salt. Pokiaľ je salt dostatočne veľký, tak by to bolo veľmi náročné.

2. Chráni časté a jednoduché heslá

Vďaka saltu vyzerajú aj rovnaké heslá odlišne, keď sú zahashované. Pokiaľ teda útočník prelomí nejaké heslo, ostatní používatelia s rovnakým heslom sú stále zabezpečení.

4. Analýza metód a algoritmov na zisťovanie hesiel

Metódy bez použitia nástrojov na lámanie hesiel

Existuje niekoľko metód na zistenie cudzieho hesla, ktoré priamo nevyužívajú lámanie pomocou technológie a algoritmov. Sú to metódy, ktoré sa opierajú o nedokonalosti v širšom koncepte používania hesiel (heslá sa musia do systému zadávať, volia si ich používatelia atď.) alebo o slabiny v konkrétnych prípadoch (firma používa všade také isté heslo, ktoré sa dá odvodiť z informácií o firme). Tieto metódy sú často efektívnejšie a menej časovo náročné ako využitie nástrojov na lámanie hesiel. Dá sa proti nim chrániť opatrnosťou a dodržiavaním zásad bezpečnosti.

Odpozorovanie hesiel (“shoulder surfing”)

Heslá sa zvyknú zadávať do systému pomocou klávesnice. Pre útočníka je jednoduché odsledovať postupnosť stláčaných kláves, poprípade odčítať heslo z monitora ak je na ňom zobrazené.

Je veľmi dôležité pred zadaním hesla skontrolovať, či nám niekto nepozera cez plece, v resp. či nás nesleduje kamera, ktorú útočník nastražil. Odporúča sa zakryť klávesy jednou rukou, aby nikto nemohol vidieť, ktoré stláčame. Napr. pri platbe kartou sú niektoré terminály ochránené krytom. Taktiež je dôležité dávať si pozor pri zobrazovaní hesiel na obrazovke a každý systém by mal používateľa patrične upozorniť pred tým, ako heslo zobrazí. Pri zadávaní by mali byť heslá na obrazovke nahradené znakom *.

Táto metóda zisťovania hesiel sa dá využiť aj s podporou softvéru na nahrávanie obrazu alebo ukladanie stlačených kláves a ich poradia.

Phishing

Phishing je spôsob ako od používateľov získať heslá pomocou podvodu, klamlivých e-mailov a falošných linkov. Princípom tejto metódy je presvedčiť držiteľa hesla, že ho využíva na autentifikáciu do určeného systému, ale v skutočnosti ho posiela priamo útočníkovi.

Typickým príkladom je e-mail, ktorý útočník pošle a zamaskuje tak, aby vyzeral že je od administrátora systému alebo je automaticky vygenerovaný systémom, do ktorého má obeť skrz svoje heslo prístup. Zároveň do e-mailu vloží link na stránku, ktorá sa podobá na stránku patriacu k danému systému (aj link zamaskuje, aby vyzeral dôveryhodne) a táto stránka si vyžiada od obete heslo.

Najčastejšie ide o falošnú potrebu obnovy hesla, ale klamlivé dôvody sú často odvodené aj od typu systému (napr. ponúkание zliav v e-shopoch).

Je potrebné vždy overiť, kam skutočne smeruje link v takýchto e-mailoch a nezadávať heslo pokiaľ neexistuje istota, že stránka patrí systému.

Spidering

Veľké firmy často používajú obrovské množstvo hesiel na prístup do rôznych systémov. Je veľmi ťažké zabezpečiť originalitu a potrebnú zložitosť všetkých týchto hesiel. Zároveň sú často používané predvolené (“default”) heslá pre nových používateľov, pri ktorých sa očakáva, že budú hneď zmenené používateľom (nie vždy sa to aj stane).

Spidering je proces získavania a zbierania informácií o prevádzke a používaní systému, do ktorého sa útočník snaží nabúrať. Účelom je nájsť kľúčových slov pre podporu lámania hesla, ale aj nájsť hesla samotných (predvolené heslá, aktívne staré heslá atď.). Tento proces je podporený automatizáciou, napríklad v podobe “spiderbotov” prehľadávajúcich webové stránky firmy.

Sociálne inžinierstvo (“social engineering”)

Táto metóda je jedna z najnebezpečnejších. Nielenže je nezávislá od technológie, ale aj využíva najväčšiu slabinu, ktorú majú všetky systémy spoločné, a to samotných ľudí.

Sociálne inžinierstvo je široký pojem, ktorý zahŕňa metódy opierajúce sa o dôverčivosť a nepozornosť používateľov systému. Typickým príkladom je, keď sa útočník vydáva za technickú podporu systému a zavolá používateľovi s účelom zistiť jeho heslo alebo iné potrebné informácie o systéme. Patria sem aj phishingové e-maily, s obsahom prispôbeným konkrétnej obeti, aby boli dôveryhodnejšie. Útočníci využívajú informácie zo sociálnych sietí, aby ich útoky vyzerali čo najdôveryhodnejšie. Môžu sa vydávať za zamestnancov firmy alebo presvedčiť jedného zo zamestnancov aby im poskytol cenné informácie pod falošnou zámenkou.

Metódy s použitím nástrojov na lámanie hesiel

Útok hrubou silou (“brute force”)

Táto metóda je založená na opakovanom skúšaní všetkých možností znakov na všetkých pozíciách v hesle a je jednou z najčastejších metód na lámanie používateľských hesiel. Zo všetkých metód si vyžaduje najviac výpočtového času a výkonu, ale umožňuje odhaliť aj netradičné heslá, ktoré sa neskladajú zo slov alebo iných logických celkov.

Najmä pri tomto útoku závisí, akým prostriedkom má útočník prístup. Často sa využívajú grafické karty (GPU) - aj niekoľko naraz, vďaka ich schopnosti rýchlo a paralelne vykonávať jednoduché aritmetické operácie.

Pred útokmi hrubou silou sa dá preventívne chrániť zväčšovaním dĺžky hesla a použitím zložitejšieho hashovacieho algoritmu pri zašifrovaní hesla, aby útočník musel čakať dlhšie pri každom pokuse. Pokiaľ útočník nemá prístup k zašifrovanej reprezentácii hesla ale snaží sa do systému nabúrať priamo (cez terminál alebo iný vstupný bod) je možné aj umelo natiahnuť čas

odozvy, tak aby skúšanie veľkého množstva hesiel za sebou trvalo dlhšie. Vhodnou prevenciou je aj obmedziť počet pokusov o úspešné zadanie hesla (väčšinou na 3 alebo 5) a po ich vyčerpaní prístup cez dané heslo zablokovať na určitý alebo neurčitý čas (zásah administrátora).

Útočník vie tento proces zefektívniť, pokiaľ má nejaké informácie o štruktúre hesla alebo sa spoľahne na nedostatočnú rôznorodosť znakov. Na to slúži **maska**, ktorá umožňuje špecifikovať predpokladanú dĺžku hesla, typy znakov a ich pozície. Takýto útok sa nazýva **útok s maskou** (“mask attack”) a je využívaný najmä ak stačí útočníkovi uhádnuť jedno heslo z viacerých (väčšia pravdepodobnosť, že niektoré vyhovie maske). Príklad masky, ktorú vieme zadať nástroju Hashcat [3]: `?1?1?1?1?1?1?d?d` - takto vieme možnosti obmedziť na 8 miestne heslo, ktoré začína 6 malými písmenami (l – lower case letter) a za nimi nasledujú 2 číslice (d – digit).

Útok so slovníkom (“dictionary attack”)

Tento útok je sofistikovanejšou formou útoku hrubou silou, ktorá sa opiera o fakt, že väčšina používateľov na svete používa heslá s podobnými frázami, slovami a štruktúrou. Zároveň je časté, že používatelia majú jedno heslo s ktorým pristupujú do množstva systémov a služieb.

Útočník použije **slovník** takýchto hesiel (získaných z predošlých útokov) a za pomoci nástroja ich postupne skúša, namiesto toho, aby nástroj generoval iba postupnosti znakov. Zároveň je možné heslá generovať spájaním hesiel zo slovníka alebo ich častí, pričom vieme stále použiť aj masku alebo iné pravidlá, ktoré nástroj pri tvorení hesiel aplikuje. Celý proces útoku sa dokáže výrazne zefektívniť.

Pred takýmto útokom sa dá chrániť najmä používaním rôznych hesiel pre rôzne systémy a služby. Zároveň je potrebné heslá pravidelne meniť (napr. každých 100 dní). Pomáha aj používať netradičné heslá s nepredvídateľnou štruktúrou.

Jeden z najčastejšie používaných slovníkov - vďaka svojej dostupnosti a rozsahu - je **rockyou.txt**, pomenovaný po spoločnosti RockYou, ktorá mala uložené heslá svojich používateľov v nezašifrovanej databáze. V roku 2009 prenikli do tejto databázy útočníci a zoznam miliónov hesiel sa dostal na verejnosť.

Útok s dúhovou tabuľkou (“rainbow table attack”)

Nedostatkom útoku so slovníkom je, že každé slovo zo slovníka musíme zahashovať a porovnať s hashom hesla, ktorý poznáme. Proces hashovania je najdlhšou zložkou skúšania veľkého množstva hesiel. Vieme však tieto hashe vypočítať dopredu pre všetky slová v slovníku, či dokonca pre čo najviac postupností znakov. Vytvorili by sme páry heslo-hash. Potom by nám stačilo porovnávať hashe. Takýto “slovník” by zaberal veľké množstvo miesta (hashe sú oveľa väčšie ako samotné heslá). Takáto metóda predstavuje výmenu času za priestor. Pri naivnej implementácii, by tento priestor bol enormný.

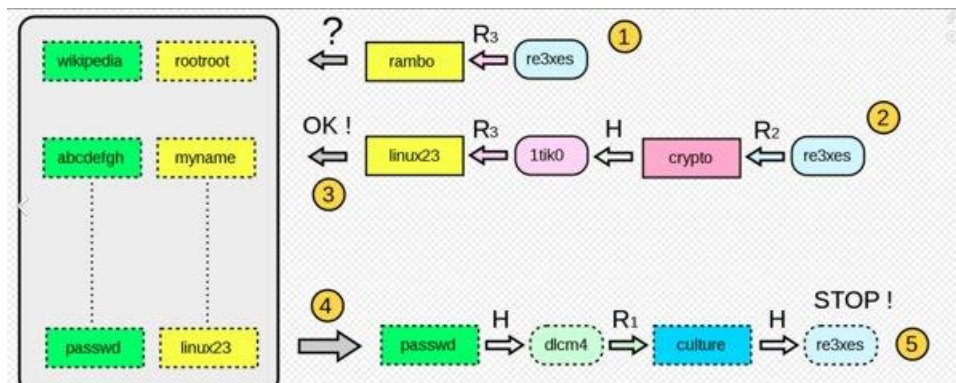
Rainbow tables používajú špeciálny algoritmus na zmenšenie daného priestoru (z TB na GB až stovky MB) na úkor času - stále sú rýchlejšie ako predošlé útoky. Tento algoritmus je

založený na vytváraní **reťazí hashov** (“hash chains”) použitím **redukčných funkcií** R . Redukčná funkcia je funkcia, ktorá z hashu hesla spraví nové a iné potenciálne heslo.

$\text{aaaaaa} \xrightarrow{H} 281DAF40 \xrightarrow{R} \text{sgfnyd} \xrightarrow{H} 920ECF10 \xrightarrow{R} \text{kiebgf}$

Hash chain vzniká striedaním aplikovania hashovacej a redukčnej funkcie po určenú dĺžku k , pričom sa uloží len posledné a prvé slovo (**startpoint a endpoint**). Pokiaľ chceme zistiť heslo patriace nejakému hashu, tak na hash tiež striedavo aplikujeme redukčnú a hashovaciu funkciu, kým nenarazíme na jeden z endpointov. Následne ideme od startpointu prislúchajúcemu danému endpointu a robíme to isté, kým v reťazi nenájdeme hľadaný hash, naše heslo je potom slovo, ktoré bolo v reťazi pred hashom.

Rainbow tables používajú niekoľko redukčných funkcií v rámci reťaze, aby sa predišlo kolíziám. V tomto prípade pri hľadaní endpointu k lámanému hashu začíname vždy poslednou redukčnou funkciou a ideme vždy o krok späť tak, ako na obrázku nižšie (výpočet trvá dlhšie).



Použitie dobrých redukčných funkcií vie zabezpečiť pokrytie čo najviac možných hesiel v rámci jednej rainbow table - redukčná funkcia generuje potenciálne heslá. Veľkou výhodou je, že nemusíme vždy nájsť heslo, ktoré používateľ zadal do systému. Stačí nájsť slovo s rovnakým hashom - využívame to, že aj hashovacie algoritmy majú kolízie.

Najjednoduchšou a najefektívnejšou ochranou proti rainbow table útokom je **solenie**, pretože každé heslo má unikátny salt. Z toho vyplýva, že by sme potrebovali rainbow table pre každý možný salt, v resp. by sme s jednou tabuľkou vedeli prelomiť len jedno konkrétne heslo.

5. Analýza nástrojov na lámanie hesiel

Pri nasledujúcej demonštrácii funkcií a používania nástrojov sme využívali hashovací algoritmus MD5 na hashovanie hesiel (pomocou Linuxového príkazu `md5sum`), kvôli jeho rýchlosti. Nástroje boli použité na systéme s procesorom Intel Core i7 3.6GHz a grafickou kartou NVIDIA GeForce GTX 1080, pričom pre porovnávanie nástrojov sme využívali iba procesor.

John the Ripper

John the Ripper (JtR) je nástroj na lámanie zahashovaných hesiel zo súboru za pomoci slovníkov, pravidiel a masiek. Pri základnom volaní: `$ john hashes.txt`, kde *hashes.txt* je súbor so zahashovanými heslami sa JtR pokúsi identifikovať o aký druh hashu sa jedná a potom za pomoci predvoleného slovníka odhaliť heslá. Pokiaľ mu ostanú heslá, ktoré neboli v slovníku, tak sa prepne do módu incremental:ASCII. To predstavuje brute force útok s ASCII character setom.

Vytvorili sme súbor desiatich jednoduchých hesiel: **jana1211, bumbum, fittka, admin1, root, 123456, qwerty, asdfg, kali, user, guest**. Tieto heslá sme zahashovali MD5 algoritmom a následne sme sa pokúsili ich prelomiť pomocou JtR.

Hneď na začiatku JtR hlási:

Warning: detected hash type "LM", but the string is also recognized as "Raw-MD5"

Use the "--format=Raw-MD5" option to force loading these as that type instead

A následne predpokladaný čas na dokončenie lámania hesiel je 2 dni. Ak ale použijeme odporúčaný príkaz a prezradíme JtR o aký hash sa jedná, tak všetky hashe prelomí za 51 sekúnd.

```
$ john weak_pwds_md5 --format=Raw-MD5
```



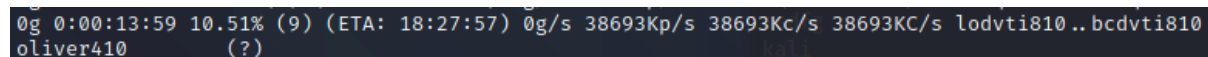
The screenshot shows the terminal output of the command `john weak_pwds_md5 --format=Raw-MD5`. It starts with "Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist" and lists several passwords with question marks: 123456, qwerty, asdfg, admin1, kali, bumbum, root, jana1211, user, and fittka. Then it says "Proceeding with incremental:ASCII". After a progress bar, it shows "10g 0:00:00:51 DONE 3/3 (2021-04-04 15:32) 0.1950g/s 30105Kp/s 30105Kc/s 59230KC/s fig4?J..fito46". Below this, it says "Use the '--show --format=Raw-MD5' options to display all of the cracked passwords reliably" and "Session completed". On the right side of the terminal, there is a sidebar with "Places" and "Desktop" icons.

Taktiež môžeme vidieť, ktoré heslá sa nachádzajú v predvolenom slovníku a JtR ich pozná. Ostatné sú prelomené hrubou silou.

Pridáme teraz heslá **oliver410** a **bratislava**. Prvá vec, ktorú si všimneme je, že predošlých 10 hesiel sa nám načíta namiesto toho, aby boli lámané odznova. JtR si ukladá prelomené heslá do súboru *john.pot* a pomocou príkazu `--show` ich vieme zobrazit' pri načítaní. Dve pridané heslá nie sú v predvolenom slovníku a zároveň sú celkom dlhé (9 a 10 znakov) a ich lámanie by trvalo veľmi dlho.

Na zlomenie prvého hesla **oliver410** skúsime **útok s maskou**. Tvárime sa, že vieme aké dlhé je heslo a z akých typov znakov sa skladá. Predpokladaný čas prelomenia hesla sú 2 hodiny. Heslo je však prelomené po 14 minútach a preskúmaní 10,51% možných hesiel.

```
$ john weak_pwds_md5 --format=Raw-MD5 --mask=?l?l?l?l?l?l?d?d?d -min-len=9
```



The screenshot shows the terminal output of the command `john weak_pwds_md5 --format=Raw-MD5 --mask=?l?l?l?l?l?l?d?d?d -min-len=9`. It shows "0g 0:00:13:59 10.51% (9) (ETA: 18:27:57) 0g/s 38693Kp/s 38693Kc/s 38693KC/s lodvti810..bcdvti810" and "oliver410 (?)".

Na zefektívnenie prelomenia tohto hesla a umožnenie prelomenia hesla **bratislava** vytvoríme vlastný slovník, ktorý sa bude skladať z prelomených hesiel a informácií o používateľovi, napr. meno, mesto a dátum narodenia. Slovník bude teda obsahovať slová: **jana1211, bumbum, fittka, admin1, root, 123456, qwerty, asdfg, kali, user, guest, Oliver, Leontiev, 4, 10, 1999, Bratislava.**

Útok so slovníkom vykonáme pomocou príkazu:

```
$ john medium pwds md5 --format=Raw-MD5 --wordlist=my_wordlist
```

JtR rýchlo prácu skončí, ale ani jedno heslo neprelomí, lebo sa v slovníku nenachádzajú. Potrebujeme použiť pravidlá, ktoré umožnia operácie nad slovami v slovníku a vytvoria z nich nové možnosti a kombinácie (“mangling rules”). Predvolený súbor pravidiel pre útok so slovníkom je “Wordlist”.

```
$ john medium pwds md5 --format=Raw-MD5 --wordlist=my_wordlist --rules:Wordlist
```

```
bratislava      (?)
1g 0:00:00:00 DONE (2021-04-04 16:46) 1.639g/s 314.7p/s 314.7c/s 314.7C/s jana1211..root0
```

Vďaka mangling rules sme okamžite zlomili heslo **bratislava**, ktoré bolo v slovníku s veľkým písmenom na začiatku. Pravidlá nezahŕňajú kombinovanie slov v slovníku do jedného hesla, čo je presne to čo potrebujeme na zistenie posledného hesla **oliver410**. Túto funkcionality umožňuje mód **prince**, ktorý skúša lepiť slová zo slovníka dokopy pre nájdenie hesla. Kombináciou pravidiel a prince módu vieme zlomiť aj posledné heslo.

```
$ john medium pwds md5 --format=Raw-MD5 --prince=my_wordlist --rules
```

```
oliver410      (?)
2g 0:00:00:00 DONE (2021-04-04 17:11) 3.571g/s 10628p/s 10628c/s 13371C/s 410USER44..10bumbum4
```

Pridáme heslo **LeontieV62**, ktoré vieme prelomiť pomocou hybridného útoku so slovníkom a maskou. Musíme použiť explicitne všetky pravidlá (--rules:All), lebo predvolené nezahŕňajú zmenu malých písmen na veľké na konci slova.

```
$ john harder pwd md5 --format=Raw-MD5 -w=my_wordlist --rules:All --mask='?w?d?d'
```

```
LeontieV62     (?)
1g 0:00:00:01 DONE (2021-04-04 17:44) 0.5649g/s 295159p/s 295159c/s 295159C/s LeontiEv19..LeontieV
```

JtR poskytuje aj príkaz --loopback , ktorý použije prelomené heslá zo spomínaného *john.pot* súboru ako slovník.

Ďalej sme vytvorili súbor zahashovaných hesiel podľa 25 najpoužívanejších hesiel v roku 2020 (podľa stránky nordpass.com). Sú to heslá: **123456, 123456789, picture1, password, 12345678, 111111, 123123, 12345, 1234567890, senha, 1234567, qwerty, abc123, Milion2, 000000, 1234, iloveyou, aaron431, password1, qqww1122, 123, omgpop, 12321, 654321.**

Pri pokuse prelomiť tieto heslá pomocou slovníka *rockyou.txt* bolo výsledkom, že bez použitia mangling rules sme okamžite prelomili 22 z 25 hesiel a ostatné sa v slovníku nenachádzali. Pri použití všetkých mangling rules (-rules:All) sme prelomili všetkých 25 hesiel za 2 minúty. To je ukážka toho, aké je nebezpečné používať časté heslá a zároveň ako efektívne vedia byť útoky so známymi slovníkmi.

JtR disponuje príkazom **unshadow**. Tento príkaz je veľmi užitočný, ak sa nám podarí získať súbory *etc/passwd* a *etc/shadow* z Linux systému. V týchto súboroch sú (okrem iného) uložené prihlasovacie údaje a zahashované heslá používateľov. Ak ich chceme prelomiť pomocou JtR, nemusíme ich manuálne extrahovať z daných súborov. Stačí nám použiť príkaz unshadow:

```
$ unshadow passwd shadow > unshadowed
```

Tým spojíme súbory *passwd* a *shadow* do jedného súboru *unshadowed*, ktorý JtR vie prečítať a snažiť sa prelomiť.

Hashcat

Hashcat je nástroj na lámanie hesiel, ktorý je alternatívou k JtR a ponúka niekoľko odlišných funkcií a mechanizmov. Ovládanie nástroja hashcat je sprostredkované pomocou číselných označení pre rôzne typy útokov a hashov.

Typy útokov sa v hashcate nazývajú **módy útoku** a sú označené číslami:

0 – Straight

Predstavuje útok so slovníkom, ekvivalentný `--wordlist` v JtR.

1 – Combination

Útok, ktorý vyžaduje špecifikovať dva slovníky (alebo 2-krát ten istý slovník) a zreťazí všetky slová z druhého ku slovám z prvého (nie naopak). Jedná sa o menej výkonnú alternatívu *prince* útoku z JtR, pretože zreťazenie nie je viacnásobné ani obojsmerné a nevyskúšajú sa samotné slová zo slovníkov bez zreťazenia. Pomocou prepínačov `-j` a `-k` môžeme špecifikovať rôzne mangling rules pre slová z prvého (`-j`) a druhého (`-k`) slovníka.

3 – Brute force

Klasický útok hrubou silou, ekvivalentný `--incremental` v JtR. Zároveň aj útok s maskou, keďže hashcat umožňuje špecifikovať masku v tomto móde.

6 – Hybrid Wordlist + Mask

Hybridný útok s použitím masky na doplnenie slovníka (pridá masku na koniec každého slova zo slovníka). Podobný `-w=wordlist.txt` a použitím `w` v maske v JtR.

7 – Hybrid Mask + Wordlist

Opačná verzia módu 6. Hashcat na rozdiel od JtR neumožňuje jednoducho umiestniť slovník doprostred masky (iba na začiatok alebo koniec). Dá sa to ale implementovať použitím pravidiel.

Dôležitým rozdielom medzi hashcatom a JtR je, že hashcat nám umožňuje jednoducho využiť rôzne hardwarové zariadenia, prepínať medzi nimi, či dokonca ich paralelne využiť spolu v rámci jedného útoku. Využíva sa na to prepínač `-d` alebo `-D`. To nám neskôr umožní využiť grafickú kartu na urýchlenie lámania hesiel. Zatiaľ ale obmedzíme hashcat na procesor, aby sme videli porovnanie s JtR (ten mal prístup iba k procesoru).

Hashcat nedokáže detegovať hash tak, ako JtR a musíme mu ho explicitne zadať pri každom útoku pomocou prepínača `-m` a číselného kódu hashu (MD5 má kód 0).

Pokiaľ sa pokúsime hashcat použiť tak, ako na začiatku JtR a zadáme mu iba súbor s hashmi na prelomenie (`$ hashcat hashes.txt`), zistíme, že hashcat nemá predvolený (“default”) mód. Má ale predvolené hodnoty a to sú: Straight mód útoku a MD5 hash. To ale znamená, že hashcat očakáva slovník a ten sme mu nedodali a náš útok zlyhá (hashcat nemá predvolený slovník, na rozdiel od JtR).

Zopakujeme lámanie hesiel **jana1211, bumbum, fittka, admin1, root, 123456, qwerty, asdfg, kali, user, guest** pomocou brute force módu v hashcate. Hashcat nielen vypíše prelomené heslá, ale aj podáva detailný výpis informácií o stave prebiehajúceho útoku. Konečný vstup a výstup vyzerajú takto:

```
>hashcat -a 3 -m 0 weak_pwds.txt
```

```
Session.....: hashcat
Status.....: Cracked
Hash.Name.....: MD5
Hash.Target....: weak_pwds.txt
Time.Started....: Sat May 08 17:17:59 2021 (8 mins, 37 secs)
Time.Estimated...: Sat May 08 17:26:36 2021 (0 secs)
Guess.Mask.....: ?1?2?2?2?2?2?2?3 [8]
Guess.Charset....: -1 ?1?d?u, -2 ?1?d, -3 ?1?d*!$@_, -4 Undefined
Guess.Queue.....: 8/15 (53.33%)
Speed.#2.....: 286.0 MH/s (7.13ms) @ Accel:1024 Loops:256 Thr:1 Vec:8
Recovered.....: 11/11 (100.00%) Digests
Progress.....: 142199488512/5533380698112 (2.57%)
Rejected.....: 0/142199488512 (0.00%)
Restore.Point....: 1769472/68864256 (2.57%)
Restore.Sub.#2...: Salt:0 Amplifier:2048-2304 Iteration:0-256
Candidates.#2....: 1zae2via -> jorgcoul
Hardware.Mon.#2..: N/A

Started: Sat May 08 17:10:06 2021
Stopped: Sat May 08 17:26:37 2021
```

Prelomené heslá nájdeme v konzole, lebo sa vypísali počas útoku. Tak isto ich vieme vypísať pomocou `--show`, keďže aj hashcat ukladá prelomené heslá (do súboru *hashcat.potfile*).

Vidíme, že prelomenie týchto hesiel hashcatu trvalo **16 minút**. To je v porovnaní s JtR - ktorému to za pomoci predvoleného slovníka trvalo 51 sekúnd - oveľa horší výsledok. Zaujímavé je, že hashcat skoro celý čas lámal heslo **jana1211**. To je logické, keďže je najdlhšie s dĺžkou 8 znakov. Lenže toto heslo sa nenachádzalo v slovníku JtR. Z toho vyplýva, že brute force útok v JtR má inteligentný vnútorný algoritmus (predvolenú postupnosť masiek), ktorý umožňuje efektívnejšie lámanie hrubou silou (minimálne v niektorých prípadoch). V hashcate by sme museli túto masku vybrať alebo vytvoriť samy. Hashcat používa “naivný” brute force – postupne zväčšuje súbor znakov a dĺžku hesla a začína od najmenších hodnôt.

Použime heslo **bratislava** na ukážku slovníkového útoku s použitím pravidiel. Opäť použijeme slovník so slovami **jana1211, bumbum, fittka, admin1, root, 123456, qwerty, asdfg, kali, user, guest, Oliver, Leontiev, 4, 10, 1999, Bratislava**. V hashcate sú pravidlá uložené v *.rule* súboroch a musíme špecifikovať cestu ku konkrétnemu súboru pravidiel pomocou prepínača `-r` - *best64.rule* obsahuje najpoužívanejšie pravidlá.

```
>hashcat -a 0 -m 0 medium_pwds_md5.txt my_wordlist.txt -r rules/best64.rule
```



```

948930d408d148b0a8767e66138b460e:bratislava

Session.....: hashcat
Status.....: Exhausted
Hash.Name.....: MD5
Hash.Target.....: medium_pwds_md5.txt
Time.Started.....: Sat May 08 19:54:46 2021 (0 secs)
Time.Estimated...: Sat May 08 19:54:46 2021 (0 secs)
Guess.Base.....: File (my_wordlist.txt)
Guess.Mod.....: Rules (rules/best64.rule)
Guess.Queue.....: 1/1 (100.00%)
Speed.#2.....: 3570.6 kH/s (0.07ms) @ Accel:512 Loops:77 Thr:1 Vec:8
Recovered.....: 1/2 (50.00%) Digests
Progress.....: 1309/1309 (100.00%)
Rejected.....: 0/1309 (0.00%)
Restore.Point....: 17/17 (100.00%)
Restore.Sub.#2...: Salt:0 Amplifier:0-77 Iteration:0-77
Candidates.#2...: jana1211 -> Bslava
Hardware.Mon.#2..: N/A

Started: Sat May 08 19:54:43 2021
Stopped: Sat May 08 19:54:48 2021

```

Heslo sme úspešne prelomili. Ostatné typy útokov sú veľmi obdobné ako v JtR a nebudeme ich demonštrovať.

Dôležitou vlastnosťou hashcat je výber zariadení. Na začiatku každého útoku nám hashcat zobrazí zariadenia, ku ktorým má prístup a ktoré používa na daný útok.

```

OpenCL API (OpenCL 1.2 CUDA 11.2.162) - Platform #1 [NVIDIA Corporation]
=====
* Device #1: GeForce GTX 1080, skipped

OpenCL API (OpenCL 2.1 WINDOWS) - Platform #2 [Intel(R) Corporation]
=====
* Device #2: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 16261/16325 MB (4081 MB allocatable), 8MCU

```

Vidíme, že v danom prípade hashcat preskočil grafickú kartu a použil iba procesor. To sme zabezpečili prepínačom `-D`, za ktorým nasleduje číslo pre druh zariadenia (1 – CPU, 2 – GPU). Vieme zadať aj obe čísla (`-D 1, 2`), a tak povedať hashcatu aby použil obe paralelne. Aké to má následky ukážeme pri testovaní.

Subjektívne porovnanie nástrojov hashcat a JtR z pohľadu UX

Funkcionalita nástrojov **John the Ripper** a **hashcat** sa výrazne prekrýva, a preto je vhodné ich porovnať z pohľadu používateľskej skúsenosti a jednoduchosti používania.

John the Ripper poskytuje funkčný a intuitívny “default” mód a zároveň disponuje schopnosťou detegovať typ hashu bez toho aby bol explicitne uvedený. V predošlej ukážke sme mohli vidieť, že táto funkcia nie vždy funguje úspešne a najlepšie výsledky dosiahneme pokiaľ hash uvedieme. Napriek tomu tieto dve skutočnosti považujeme za výraznú výhodu JtR nad hashcatom. Taktiež sú názvy funkcií a prepínačov v JtR intuitívnejšie na používanie. JtR disponuje niekoľkými funkciami, ktoré hashcat neposkytuje a sú veľmi užitočné, napr. **prince**

mód a unshadow. Nevýhodou JtR je chýbajúca možnosť výberu zariadení a minimálne výpisy informácií o stave útoku.

Hashcat poskytuje jednoduchý spôsob ako využiť vybrané zariadenia na vykonanie útoku. Počas útoku poskytuje príliš veľa detailných informácií o priebehu, a to podstatné (kedy bolo nejaké heslo prelomené) sa ľahko stratí. V hashcate sú všetky módy útoku a typy hashov ukryté za číselnými označeniami. To považujeme za neprehľadné a zbytočne to zvyšuje náročnosť používania. Taktiež uprednostňujeme výber pravidiel v JtR (stačí názov typu pravidiel, napr. Wordlist alebo All), pred tým v hashcat (musíme zadať cestu k súboru s pravidlami). Hashcat poskytuje veľa funkcií nad rámec JtR (napríklad Workload Profiles, generovanie náhodných pravidiel atď.), ale žiadne z nich nám neprišli kritické alebo všeobecne aplikovateľné (majú využitie v konkrétnych prípadoch).

Podporované typy hashov oboch nástrojov nie sú úplne rovnaké. Každý ponúka niekoľko extra typov, ktoré druhý nástroj nepodporuje.

Výsledkom porovnania nástrojov z pohľadu používateľskej skúsenosti je, že definitívne uprednostňujeme jednoduchosť a intuitívne prvky nástroja **John the Ripper**.

Hydra

Hydra je nástroj určený na nabúranie sa do systému, ktorý vyžaduje prihlásenie cez sieť. Používa na to útok so slovníkom (Hydra neposkytuje funkcionality útoku s hrubou silou). Hydra vytvorí niekoľko pripojení cez určený protokol (SSH, FTP atď.) a vyskúša sa prihlásiť do systému všetkými heslami zo slovníka. Tento útok môže zahŕňať aj prihlasovacie mená.

Proti takémuto útoku sa dá jednoducho ochrániť obmedzením počtu nesprávnych prihlásení alebo paralelných pripojení. Preto sme museli ako obeť použiť zraniteľný systém, ktorý tieto opatrenia nepoužíva. Zvolili sme **Metasploitable 2**.

Metasploitable 2 je virtuálny stroj s Ubuntu Linux OS, ktorý je určený ako obeť pre testovanie a má množstvo úmyselných zraniteľností, ktoré nám umožnia demonštrovať funkcie nástroja Hydra. Hydru spustíme na Kali Linux VM a budeme útočiť na VM s bežiacim Metasploitable 2.

Hydra sa používa zadaním prihlasovacieho mena, hesla, IP adresy obete a protokolu. Prihlasovacie meno sa zadá pomocou prepínača `-l` alebo `-L` pre zadanie súboru so slovníkom prihlasovacích mien na vyskúšanie. Heslá sa zadávajú analogicky s prepínačmi `-p` a `-P`. Môžeme taktiež vytvoriť súbor s párami login:heslo a ten zadať pomocou prepínača `-C`. Zároveň vieme zadať aj súbor so zoznamom adries systémov, na ktoré chceme zaútočiť. Na to slúži prepínač `-M`.

Prihlasovacie meno aj heslo do Metasploitable 2 je **msfadmin** a IP adresa je 192.168.100.4. Vytvorili sme slovník zo 780 hesiel z *rockyou.txt* a na koniec sme pridali heslo msfadmin. Nazvali sme ho *meta_wordlist*. Takto dopadol útok cez protokol FTP:

```

$ hydra -l msfadmin -P meta_wordlist 192.168.100.4 ftp
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and et
hics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-14 11:19:19
[DATA] max 16 tasks per 1 server, overall 16 tasks, 781 login tries (l:1/p:781), ~49 tries per ta
sk
[DATA] attacking ftp://192.168.100.4:21/
[STATUS] 304.00 tries/min, 304 tries in 00:01h, 477 to do in 00:02h, 16 active
[STATUS] 296.00 tries/min, 592 tries in 00:02h, 189 to do in 00:01h, 16 active
[21][ftp] host: 192.168.100.4 login: msfadmin password: msfadmin
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-04-14 11:22:03

```

Môžeme vidieť, že Hydra vytvorila 16 pripojení (**tasks**) a našla správne heslo za **3 minúty**. To znamená, že máme FTP prístup k systému Metasploitable 2 a vieme z neho stiahnuť akýkoľvek súbor.

Následne sme útok zopakovali pre protokol SSH. Hydra nás upozorní, že SSH zvykne obmedzovať počet paralelných tasks a v skutočnosti nám dovolí spustiť iba 14. Počet tasks vieme obmedziť prepínačom **-t**.

```

$ hydra -l msfadmin -P meta_wordlist 192.168.100.4 ssh -t 14
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and et
hics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-14 11:45:27
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce
the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a pr
evious session found, to prevent overwriting, ./hydra.restore
[DATA] max 14 tasks per 1 server, overall 14 tasks, 781 login tries (l:1/p:781), ~56 tries per ta
sk
[DATA] attacking ssh://192.168.100.4:22/
[STATUS] 443.00 tries/min, 443 tries in 00:01h, 367 to do in 00:01h, 14 active
[STATUS] 275.50 tries/min, 551 tries in 00:02h, 259 to do in 00:01h, 14 active
[STATUS] 211.67 tries/min, 635 tries in 00:03h, 175 to do in 00:01h, 14 active
[STATUS] 193.50 tries/min, 774 tries in 00:04h, 36 to do in 00:01h, 14 active
[22][ssh] host: 192.168.100.4 login: msfadmin password: msfadmin
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 11 final worker threads did not complete until end.
[ERROR] 11 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-04-14 11:49:40

```

Opäť Hydra heslo prelomila, ale trvalo to o niečo dlhšie (**4 minúty**), kvôli menšiemu počtu tasks.

Skúsili sme aj použiť rovnaký slovník a prelomiť prihlasovacie meno a heslo naraz. Takýto útok bol úspešný po **2 hodinách**.

6. Testovanie výkonu nástrojov

Typy testov

Vykonáme 5 typov testov:

1. **Porovnanie všetkých troch nástrojov** (John the Ripper, hashcat a Hydra).

Vzhľadom na to, že Hydra poskytuje iba útok so slovníkom, tak sa bude jednať práve o **dictionary attack**. Porovnáme rýchlosť prelomenia hesla do systému Metasploitable 2 cez

FTP protokol s rýchlosťou prelomenia hashu hesla nástrojmi JtR a hashcat. Aby bol útok nástrojmi autentický, využijeme súbor *shadow* zo systému Metasploitable 2 na získanie hashov na prelomenie. V JtR využijeme funkciu unshadow a v hashcate hashe extrahujeme manuálne.

2. Krátke a predvídateľné heslá (JtR vs. hashcat)

Pri používaní nástrojov sme si všimli, že JtR je úspešnejší pri lámaní hesiel s predvídateľnou štruktúrou (slová, slová s číslicami na konci atď.). Vytvoríme zoznam krátkych predvídateľných hesiel a porovnáme rýchlosť JtR a hashcat pri lámaní **hrubou silou**.

3. Krátke a náhodné heslá (JtR vs. hashcat)

Vykonáme útok hrubou silou nad zoznamov krátkych hesiel vygenerovaných generátorom hesiel. Porovnáme výsledky tohto testu s testom 2.

4. Dlhé a predvídateľné heslá (JtR vs. hashcat)

Otestujeme lámanie dlhších hesiel s predvídateľnou štruktúrou útokom hrubou silou.

5. Dlhé a náhodné heslá (JtR vs. hashcat)

Otestujeme lámanie hrubou silou dlhších hesiel vygenerovaných generátorom hesiel. Výsledky porovnáme s testom 4.

Metriky testov

V nasledujúcich testoch budeme porovnávať **rýchlosť úspešného prelomenia** niekoľkých hashov nástrojmi. Všetky tri nástroje porovnáme iba v prvom teste. Ďalej budeme testovať iba JtR a hashcat. Zároveň testy 2-5 otestujeme aj na GPU a kombinácií CPU + GPU pomocou nástroja hashcat. Uvidíme, o koľko je použitie GPU rýchlejšie, a či použitie CPU a GPU naraz je ešte rýchlejšie.

Pri teste budeme aj sledovať, či bol úspešný v rozumnom časovom rozmedzí. Pokiaľ nie tak ho prehlásime za neúspešný a zapíšeme čas a stav vykonávania pred zastavením.

Testy 2-5 vykonáme pre hashe MD5 aj SHA1, aby sme porovnali a demonštrovali ich silu. Očakávame, že SHA1 bude v každom prípade trvať prelomiť dlhšie, a teda je silnejší.

Test 1: dictionary attack

Detaily testu

Vytvorili sme slovník so 780 slovami z *rockyou.txt* a na koniec sme pridali správne heslo. Pre hashcat a JtR sme lámali hashe z *shadow* súboru. Pre Hydru sme hľadali heslo, ktoré je vstupom do systému cez FTP protokol pri maximálnom počte tasks (16).

Očakávaný výsledok

Vzhľadom na to, že Hydra musí všetky heslá otestovať po sieti očakávame lepší výsledok pri offline lámaní pomocou nástrojov hashcat a JtR. Medzi týmito dvoma nástrojmi neočakávame veľký rozdiel.

Výsledok

Typ testu	hashcat	JtR	Hydra
1: dictionary attack	1 sek	1 sek	3:16 min

Zhodnotenie výsledku

Náš predpoklad bol síce správny, ale taký veľký rozdiel sme neočakávali. Test preukazuje, že offline lámanie hashov je oveľa efektívnejšie ako útok po sieti s nástrojom Hydra.

Test 2: krátke a predvídateľné heslá

Detaily testu

Pokúsime sa prelomiť hashe nasledovných 10 hesiel s dĺžkou 6 znakov: **admin1**, **user00**, **oliver**, **passwd**, **qwerty**, **asdfgh**, **123456**, **abcdef**, **987654**, **rocker**.

Očakávaný výsledok

Očakávame, že JtR bude výrazne lepší v tomto teste ako v teste 3. Napriek tomu predpokladáme, že hashcat bude rýchlejší kvôli jeho algoritmu, ktorý je viac naklonený kratším heslám. Rozdiel očakávame tesný.

Výsledok

Typ testu	hashcat (CPU)	JtR (CPU)	hashcat (GPU)	hashcat (CPU+GPU)
2: MD5	14 sek	2:08 min	4 sek	4 sek
2: SHA1	18 sek	3:09 min	6 sek	5 sek

Zhodnotenie výsledku

Ako vidíme, hashcat prekonal JtR s väčším rozdielom ako sme čakali. Aj počas pokusu sme sledovali, že zatiaľ čo hashcat láme heslá od kratších dĺžok po dlhšie, JtR skáče medzi dĺžkami a hľadá heslá najmä v oblasti 7 a 8 znakov.

Test 3: krátke a náhodné heslá

Detaily testu

Pokúsime sa prelomiť hashe nasledovných 10 hesiel s dĺžkou 6 znakov: **nmdsbf, bqvlif, ysuopf, bagcuf, gsgfrf, noffif, surfnf, tyvksf, vasedf, mbvvvf**. Tieto heslá sme vygenerovali nástrojom <https://www.random.org/strings/>.

Najprv sme tento test skúsili s heslami vygenerovanými online nástrojom <https://www.dashlane.com/features/password-generator>. Tieto heslá boli všetky dĺžky 5 znakov: **uNH*5, D^BqX, S\od5, %X4nT, 7C!i:, t8yC., q+3@P, p6}^C, e9~AS, Rr6#***. Ani jednému nástroju sa nepodarilo prelomiť ani jedno heslo pod 1 hodinu. Takýto test nám ukázal, že aj krátke heslá vedia byť silné pri správnom striedaní typov znakov a využití netradičných symbolov.

Očakávaný výsledok

Očakávame, že JtR bude výrazne horší ako v teste 2. Zároveň očakávame, že výsledky hashcat budú rovnaké ako v teste 2.

Výsledok

Typ testu	hashcat (CPU)	JtR (CPU)	hashcat (GPU)	hashcat (CPU+GPU)
3: MD5	17 sek	5:58 min	4 sek	4 sek
3: SHA1	22 sek	12:21 min	6 sek	5 sek

Zhodnotenie výsledku

Naše očakávania sa naplnili. Vidíme, že JtR naozaj skôr prelomí predvídateľné heslá. Hashcat bol nepatrne pomalší oproti testu 2. To je pravdepodobne spôsobené iba tým, že znaky v heslách boli tentokrát ďalej v prehľadávajúcom priestore hashcat.

Test 4: dlhé a predvídateľné heslá

Detaily testu

Pokúsime sa prelomiť hashe nasledovných 5 hesiel s dĺžkou 8 znakov: **jana1211**, **rootroot**, **12345678**, **asdfghjk**, **admin123**.

Očakávaný výsledok

Očakávame výrazný náskok JtR oproti hashcat (CPU), keďže vieme, že JtR sa dostane k dlhším heslám skôr ako hashcat, a zároveň rýchlejšie láme predvídateľné heslá. Pri hashcate na CPU počítame s časom nad 15 minút. Zároveň očakávame, že uvidíme rozdiel medzi GPU a GPU + CPU utilizáciou. Je pravdepodobné, že hashcat s využitím GPU bude rýchlejší ako JtR.

Výsledok

Typ testu	hashcat (CPU)	JtR (CPU)	hashcat (GPU)	hashcat (CPU+GPU)
4: MD5	> 30 min (neúspech)	2:17 min	2:03 min	2:06 min
4: SHA1	> 30 min (neúspech)	5:01 min	2:46 min	2:44 min

Zhodnotenie výsledku

Pozorovali sme, že hashcat (CPU) nezvládol prelomiť heslá do 30 minút (prelomil 3/5) a vzhľadom na to, že JtR ich prelomil za 2 a 5 minút sme test zastavili. Očakávali sme, že JtR prekoná hashcat, ale prekonal naše očakávania a priblížil sa dokonca času hashcat (GPU) pre MD5 hash. Z toho definitívne vyplýva, že brute force mód v JtR je viac uspôsobený na heslá so štruktúrou podobnou často používaným heslám, zatiaľ čo hashcat má s dlhými heslami problém.

Predpokladali sme, že uvidíme rozdiel medzi použitím iba GPU a GPU spolu s CPU. To sa však nestalo a výsledky sú rovnaké. Dá sa z toho odvodiť, že GPU je o toľko výkonnejšie, že prídanie CPU nezvyší rýchlosť o pozorovateľné množstvo.

Test 5: dlhé a náhodné heslá

Detaily testu

Pokúsime sa prelomiť hashe nasledovných 5 hesiel s dĺžkou 7 znakov: **epgtzzn**, **rmslslpfvirhhz**, **fxdrasl**, **hqiujzn**.

Tieto heslá sme vygenerovali nástrojom <https://www.random.org/strings/>.

Očakávaný výsledok

Očakávame, že pre MD5 hash JtR tieto heslá neprelomí do 30 minút a hashcat (CPU) ich prelomí do 10 minút. Pre GPU a GPU + CPU prípady neočakávame veľkú zmenu oproti testu 4.

Výsledok

Typ testu	hashcat (CPU)	JtR (CPU)	hashcat (GPU)	hashcat (CPU+GPU)
5: MD5	10:24 min	> 30 min (neúspech)	19 sek	20 sek
5: SHA1	11:55 min	> 30 min (neúspech)	27 sek	27 sek

Zhodnotenie výsledku

Pri JtR sa naše očakávania naplnili a prelomil iba 2/5 hesiel pod 30 minút. Hashcat (CPU) tesne prekročilo predpokladanú hranicu. Z tohto testu vyplýva, že pri dlhších heslách s náhodnými znakmi je hashcat rýchlejší. JtR síce začne prehľadávať dlhšie heslá skôr, ale skáče medzi rôznymi dĺžkami a skúša najprv známe typy hesiel. Kvôli tomu ho hashcat predbehne.

Prekvapivé je o koľko rýchlejšie bolo prelomenie hesiel s pomocou GPU. Môžeme vidieť, prečo väčšina služieb pokladá minimálnu dĺžku hesla na hranicu 8 znakov. Tieto heslá boli iba 7-miestne a s GPU ich hashcat prelomil za pár desiatok sekúnd, zatiaľ čo 8-miestne v teste 4 mu zabrali aspoň pár minút.

7. Záver

Na záver uvedieme vyhodnotenie testovania a práce s nástrojmi.

Pri používaní nástrojov sme dospeli k názoru, že nástroj John the Ripper je príjemnejší a jednoduchší na používanie ako nástroj hashcat. Zároveň ponúka niekoľko extra funkcií, ktoré sme využili a boli užitočné pre našu prácu. Nástroj Hydra sa tiež používa veľmi intuitívne, ale chýba nám pri ňom možnosť útoku hrubou silou. Je ale veľmi užitočný, pokiaľ nemáme prístup k hashom hesiel a chceme sa skúsiť nabúrať priamo do systému.

Z testovania sme zistili, že útok po sieti s nástrojom Hydra je neporovnateľne menej efektívny ako offline cracking pomocou JtR alebo hashcat. Napriek tomu, že vo väčšine testov útoku hrubou silou (v rámci CPU) bol hashcat výkonnejší ako JtR, JtR bol výrazne efektívnejší pri teste 4, ktorý najviac pripomínal skutočné používateľské heslá. Kvôli tomu pokladáme JtR za vhodnejší nástroj v situáciách, kedy lámeme skutočné používateľské heslá a nemáme prístup ku GPU.

Použitie GPU vie rýchlosť prelomenia hesla niekoľkonásobne zvýšiť a hashcat je nástroj, ktorý by sme použili vždy pri prístupe ku GPU alebo ak by sme chceli využiť viac zariadení paralelne. V našich testoch použitie CPU a GPU naraz výsledok neovplyvnilo, kvôli veľkému rozdielu vo výkone týchto zariadení.

Zdroje

- 1) *Top 200 most common passwords of the year 2020.* <https://nordpass.com/most-common-passwords-list/>
- 2) *How to Crack a Password.* <https://www.guru99.com/how-to-crack-password-of-an-application.html>
- 3) *How to Perform a Mask Attack Using hashcat.* <https://www.4armed.com/blog/perform-mask-attack-hashcat/>
- 4) *Password cracking.* In Wikipedia. https://en.wikipedia.org/wiki/Password_cracking
- 5) Walker, D. (2020, November 17). *The top 12 password-cracking techniques used by hackers.* <https://www.itpro.co.uk/security/34616/the-top-password-cracking-techniques-used-by-hackers>
- 6) *Password strength.* In Wikipedia. https://en.wikipedia.org/wiki/Password_strength
- 7) Fruhlinger, J. (2019, September 25). *Social engineering explained: How criminals exploit human behavior.* <https://www.csoonline.com/article/2124681/what-is-social-engineering.html>
- 8) *Rainbow table.* In Wikipedia. https://en.wikipedia.org/wiki/Rainbow_table