Nástroj pro zpracování výstupů penetračních testů

Bc. Robin Filipčuk

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Robin Filipčuk**
Osobní číslo:	**A17224**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**prezenční**
Téma práce:	**Nástroj pro zpracování výstupů penetračních testů**
Téma anglicky:	**A Tool for Processing Penetration Tests Results**

Zásady pro vypracování:

1. **Seznamte se s problematikou penetračního testování.**
2. **Prostudujte možnosti nástrojů používaných pro penetrační testování a jejich výstupů.**
3. **Vhodným způsobem vyberte nástroje vhodné pro automatizaci penetračního testování a načítání logů.**
4. **Navrhněte strukturu výstupních dat s ohledem na možnost generování reportů na základě normy ISO 27000.**
5. **Navrhněte strukturu aplikace.**
6. **Implementujte aplikaci za použití vhodných technologií.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LAZAR, Guillaume a Robin PENEA. Mastering Qt 5. 2nd ed. Packt Publishing Limited, 2018. ISBN 978-1788995399.**
2. **Qt Documentation [online]. Dostupné také z: https://doc.qt.io/**
3. **SELECKÝ, Matúš. Penetrační testy a exploitace. Brno: Computer Press, 2012. ISBN 978-80-251-3752-9.**
4. **PRATA, Stephen. Mistrovství v C++. 3., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2007. Bestseller (Computer Press). ISBN 978-80-251-1749-1.**
5. **RISCHPATER, Ray. Application Development with Qt Creator. 2nd ed. Packt Publishing Limited, 2014. ISBN 978-1784398675.**
6. **WEIDMAN, Georgia. Penetration testing: a hands-on introduction to hacking. San Francisco: No Starch Press, [2014]. ISBN 978-1593275648.**

Vedoucí diplomové práce:	**Ing. Peter Janků** Ústav informatiky a umělé inteligence
Konzultant:	**Ing. David Malaník, Ph.D.** Ústav informatiky a umělé inteligence
Datum zadání diplomové práce:	**3. prosince 2018**
Termín odevzdání diplomové práce:	**15. května 2019**

Ve Zlíně dne 7. prosince 2018

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15.5.2019

Bc. Robin Filipčuk, v. r.
podpis diplomanta

ABSTRAKT

Hlavním cílem této diplomové práce je prozkoumání problematiky penetračního testování a prostudování norem, které s touto tématikou souvisí. Praktická část práce se zabývá vývojem aplikace určené primárně pro zpracování výstupů z penetračních nástrojů. Zpracované výstupy poslouží k vytvoření závěrečné zprávy z penetračního testování. Aplikace bude dále umět automatizované spouštění vybraných penetračních nástrojů pro zrychlení průběhu penetračního testování.

Klíčová slova: penetrační testování, desktopová aplikace, Qt, ISO/IEC 27000

ABSTRACT

The main aim of this diploma work is the exploration of penetration testing issues and studying of norms related to this topic. Practical part of the work deals with development of an application primarily designed to process outputs from penetration tools. Processed outputs will help to create concluding report from penetration testing. The application will also feature automatized launch of selected penetration tools for penetration testing acceleration.

Keywords: penetration testing, desktop appliaction, Qt, ISO/IEC 27000

Děkuji vedoucímu mé diplomové práce Ing. Peterovi Janků a konzultantovi Ing. Davidu Malaníkovi, Ph.D. za cenné rady, konzultace a pomoc při řešení problémů vyskytlých při vypracování této práce. Dále bych chtěl poděkovat rodině a kamarádům za podporu po celou dobu mého studia.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Bezpečnost informací je v dnešním digitálním světě důležitou a hojně projednávanou problematikou. Proto je tato bezpečnost řízena pravidly, která jsou určena souborem norem řady ISO/IEC 27000. Tyto normy zadávají organizacím, jak bezpečnost informací zavést, provozovat, udržovat a také kontrolovat. Právě pro kontrolu zavedení zabezpečení informací byly vytvořeny společnosti, které provádí pro organizace testování v podobě penetračních testů. Tuto činnost provádí bezpečnostní experti za pomoci různých softwarových nástrojů, které jim jejich práci zjednodušují. Cílem této práce je, vytvořit pomocný nástroj, který bude provádět automatizované spouštění penetračních nástrojů a také zpracovávat jejich výstupy. Zpracování bude prováděno takovým způsobem, že se z výstupů vytřídí pouze informace, které jsou pro bezpečnostního experta důležité. Tudíž si nemusí hledat tyto informace sám v dlouhých a nepřehledných výstupech z penetračních testů. Informace navíc budou zpracovány to takového formátu, kdy bude možné tato data následně strojově přečíst a vložit například do závěrečné zprávy z penetračního testování.

Desktopová aplikace vytvořená za použití vhodných vývojových technologií bude zpracovávat výstupní soubory z vybraných nástrojů používaných pro penetrační testování. Dále bude rozebrána metodika provádění penetračních testů a problematika norem řady ISO/IEC 27000, které s bezpečností informací přímo souvisejí.

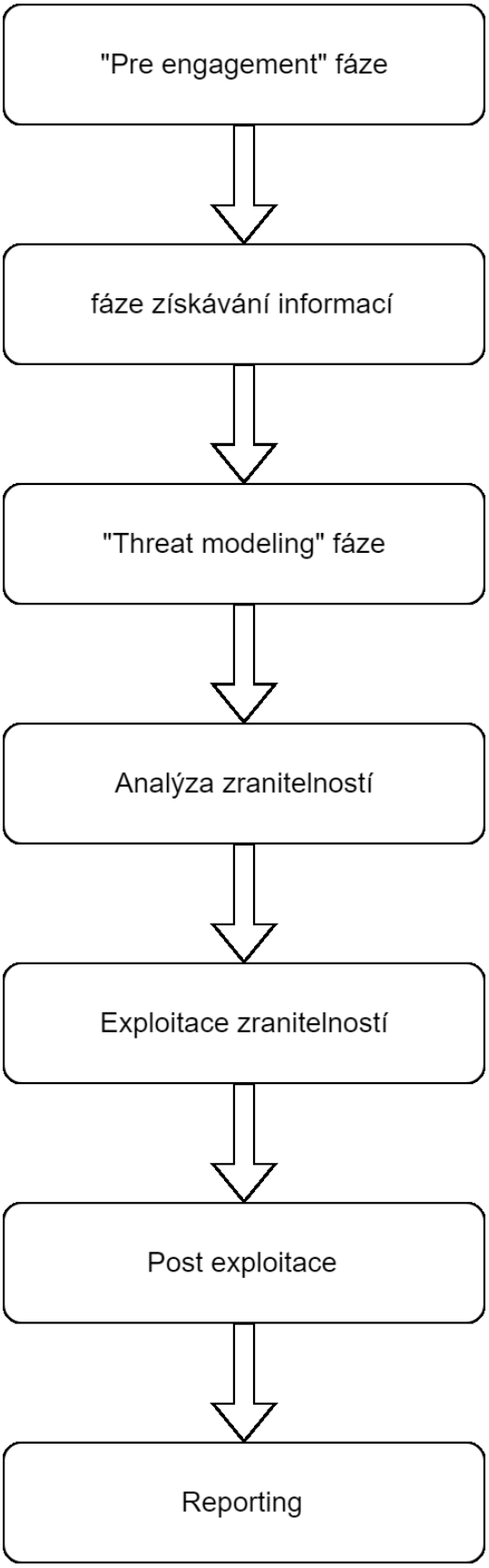
I. TEORETICKÁ ČÁST

1 PENETRAČNÍ TESTOVÁNÍ

Jedná se o simulaci reálného kyberútoku pro zjištění potencionálních bezpečnostních rizik testovaného systému. Během penetračního testu (dále jen „pentest“) zjišťuje bezpečnostní expert (dále jen „pentester“) možné zranitelnosti. Ty se pokouší využít k dalšímu proniknutí do systému, aby zjistil, co všechno může být v případě reálného útoku zneužito, poškozeno případně odcizeno. [1]

1.1 Fáze penetračního testu

Samotný pentest se skládá z několika částí (fází), které logickým způsobem na sebe navazují. Vizualizace návaznosti fází penetračního testování je obsažena v obrázku č. 1. První fáze, kterou samotný pentest také začíná, je „pre-engagement“ fáze. V této fázi probíhá diskuze mezi pentesterem a klientem o tom, jakým směrem se bude pentest ubírat, co bude jeho cílem, jaký bude jeho výstup a ostatní formality ohledně testu. Jakmile se klient a pentester dohodnou, může pentest pokračovat do další fáze. Tou je fáze „získávání informací“, při které pentester získává veřejně přístupné informace o klientovi (o cíli) a zjišťuje potenciální cesty, jak se dostat do systému. Následující fází je fáze pojmenovaná jako „threat-modeling“. Při této fázi pentester využívá nálezů z minulé fáze a každému nálezu určí hodnotu a zhodnotí také dopad na klienta. S pomocí tohoto hodnocení může pentester vytvořit plán útoku. Ještě před tím, než začne pentester útočit, provede analýzu zranitelností, při které se pokouší odhalit zranitelnosti, které mohou být zneužity ve fázi exploitace. Po úspěšné exploitaci následuje „post exploitace“, kde je využit výsledek exploitace k získání dalších informací jako jsou citlivé údaje, data, případně přístup k dalším systémům. Nakonec jsou kompletní výsledky celého pentestu vloženy do závěrečné zprávy – reportu. Posloupnost jednotlivých fází je znázorněna na obrázku číslo 1. [1] [2]



Obrázek 1 Vizualizace fází penetračního testu

1.1.1 Pre-engagement fáze

Při této fázi se pentester domlouvá s klientem na průběhu celého penetračního testování. Pentestera převážně zajímá, co klient od penetračního testu očekává, co klienta přivedlo k realizaci penetračního testu, co bude jeho hlavním cílem. Dále pak zjišťuje, zda nejsou v systému citlivá zařízení, se kterými by měl pentester pracovat opatrně nebo se jim zcela vyhnout z jiných bezpečnostních důvodů (např. v nemocničním prostředí) a další podobné informace. Mezi další důležité body, které pentester probírá s klientem během této fáze a stojí za zmínku patří například následující. [1] [2]

1.1.1.1 Rozsah testu

Zde se jedná o dohodnutí rozsahu IP adres zařízení, které budou, případně nebudou součástí testu. Dále se stanoví, jaké druhy akcí budou povoleny na těchto zařízení provádět a případně jestli nebude vadit, když pentester během exploitace vyřadí některá zařízení dočasně z provozu. A v neposlední řadě se rozhodne, jestli je pentesterovi povolen útok ve smyslu sociálního inženýrství. [1]

1.1.1.2 Testovací období

Jednou z důležitých okolností penetračního testu je dohodnout se na období provádění testu. V závislosti na klientovi je možné dohodnout se pouze na určité dny nebo hodiny, kdy je vhodné provádět penetrační test. [1]

1.1.1.3 Kontaktní informace

Pentester by si měl domluvit s klientem, zda chce být informován okamžitě o nalezených chybách. Případně si zjistí kontaktní informace na kompetentní osobou, se kterou může komunikovat v případě nalezení velké bezpečnostní chyby, u které může být nezbytná rychlá oprava pro předejití větších škod a problémů. [1]

1.1.1.4 Beztrestnost

Je velmi důležité, aby pentester měl od klienta povolení pro provádění penetračního testu na zvoleném cíli. Jestli konkrétní cíl, na který se bude útočit, není ve vlastnictví klienta, je důležité, aby klient měl formální svolení od třetích stran pro provedení penetračního testu. Bez ohledu na tyto okolnosti je vhodné, aby smlouva mezi pentesterem a klientem obsahovala

prohlášení, které zbavuje pentestera odpovědnosti v případě, že se stane něco neočekávaného. [1]

1.1.1.5 Dohoda o ceně

V neposlední řadě je vhodné, aby si pentester s klientem dohodl cenu za celkové provedení penetračního testování. Je důležité, aby i toto pentester uvedl ve smlouvě, kterou s klientem dohodne včetně ujištění klienta ohledně důvěrnosti informací, ke kterým se pentester během testu může dostat. [1]

1.1.2 Získávání informací

Během této fáze pentester analyzuje volně dostupné zdroje informací a hledá údaje o klientovi. Je možné využít různých nástrojů jako jsou nástroje pro mapování sítě, skenery portů a další. Tyto nástroje pentester použije pro zjištění, zda nejsou některé služby klientského systému přístupné z internetu, případně jaký software běží na těchto službách. [1] [2]

1.1.3 Threat modeling

Na základě informací získaných v předešlém kroku přemýšlí pentester jako útočník a vytváří si plán útoku, podle kterého bude postupovat. [1] [2]

1.1.4 Analýza zranitelností

Další činností pentestera je analýza zranitelností. Při této fázi pentester zjišťuje, v jaké míře může být strategie exploitace úspěšná. Neúspěšný útok na zranitelnosti může skončit selháním běžících služeb systému nebo může vyvolat poplach v dohlížecích systémech a tím zmenšit pentesterovi šanci na úspěšný útok vůči systému. Často pentester v této fázi využívá specializovaných nástrojů pro vyhledávání zranitelností (skenery zranitelností), které využívají rozsáhlé databáze zranitelností pro co nejlepší zjištění zranitelných chyb obsažených v klientském systému. Ačkoliv jsou tyto nástroje cenným pomocníkem, nemůže se pentester plně spoléhat na jejich výsledky, a proto je často nutné provést ještě manuální rozbor nalezených zranitelností a případně je dále ověřit. [1] [2]

1.1.5 Exploitace

V této fázi pentester aplikuje exploity vůči zranitelnostem, které odhalil v minulých fázích. Koná tak z důvodu přístupu do systému klienta. Některé zranitelnosti můžou být lehce napadnutelné jako například přístupové údaje, které jsou nastaveny na výchozí hodnoty. Tato situace je častou doménou aktivních síťových prvků. [1] [2]

1.1.6 Post exploitace

Během provádění této fáze pentester získává informace o napadeném systému, poohlíží se po důležitých datech a souborech nebo se snaží opatřit si vyšší oprávnění pro způsobení ještě větších škod na klientském systému. Například hledá soubory, kde jsou uloženy přihlašovací údaje administrátorských účtů pro jejich zneužití. Pentester může také zneužít napadený systém pro případné napadnutí jiného dalšího systému. [1] [2]

1.1.7 Reporting

Konečná fáze penetračního testu je vytvoření reportu. Je to prostředek, kde zákazníkovi pentester sdělí své nálezy ve smysluplné podobě. Klient se tedy dozví, jaké body zabezpečení jsou správné a bezpečné, co je případně potřeba zlepšit a jakým způsobem opravit nalezené bezpečnostní chyby celého systému. Pentester musí report sepsat ve srozumitelné podobě tak, aby IT správci systému byli schopni podle reportu chyby opravit a aby zranitelnostem porozuměl i vyšší management klientské společnosti. Report by měl tedy zpravidla obsahovat dvě části, a to shrnutí celého penetračního testu a také jeho technické podrobnosti. Podoba nebo obsah reportu není ničím řízena, pentester by si měl sám vhodně zvolit, co bude výsledný report přesně obsahovat, a jaká bude jeho struktura. [1] [2]

1.1.7.1 Shrnutí

Shrnutí by mělo popisovat cíle penetračního testu a nabízet obecný a srozumitelný přehled o průběhu testu. Předpokládaným čtenářem této části reportu jsou především vedoucí pracovníci odpovědní za bezpečnost systému. Obsahem může být úvodní popis a účel testu, vysvětlení některých pojmů (například zranitelnost, exploit a další), informace o efektivitě testu a nalezené problémy, ohodnocení rizik klientského systému jako celku, okrajové informace o činnostech potřebných k opravě zabezpečení systému a další. [1]

1.1.7.2 Technický report

Tato část závěrečného reportu obsahuje technické podrobnosti ohledně provedeného penetračního testu. Obsahem jsou většinou podrobné technické informace z každé fáze testu. Jmenovitě report obsahuje seznam všech nalezených zranitelností, seznam a úspěšnost útoků prováděných během fáze exploitace, výsledná rizika vycházející z fáze post exploitace a nakonec závěrečné shrnutí všech výsledků včetně míry rizika jednotlivých položek. [1]

2 NORMY ISO/IEC 27000

Pro řízení bezpečnosti informací v kontextu organizací byl vytvořen soubor mezinárodních norem ISO/IEC 27000. Tyto normy tedy určují specifikace a požadavky pro zabezpečení informací v organizacích všech typů a velikostí (například pro obchodní podniky, vládní úřady, neziskové organizace a další). Vybudování zabezpečení informací podle norem ISO/IEC 27000 přináší nižší riziko úniku citlivých informací. [3] [4]

2.1 ISO/IEC 27001

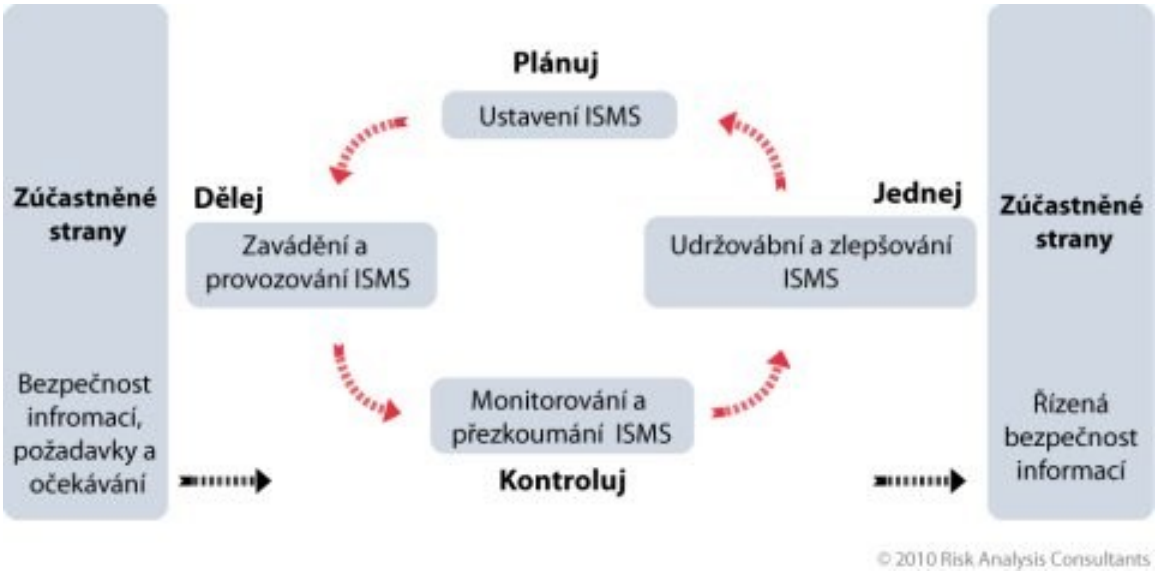
Norma ISO/IEC 27001 je hlavní a zároveň nejdůležitější normou souboru norem ISO/IEC 27000. Tato norma udává požadavky a specifikace na zavedení systému řízení bezpečnosti informací (ISMS) v organizacích a na údržbu a zlepšování tohoto systému. Splní-li organizace zavedení ISMS podle této normy, má nárok na certifikát. [5] [6] [7]

2.1.1 ISMS

Zkratka ISMS (Information Security Management System) označuje systém pro řízení informační bezpečnosti. ISMS je určený normou ISO/IEC 27001 a jedná se o všeobecně uznávaný a standardizovaný systém řízení bezpečnosti každého informačního systému obsaženém v jakékoliv organizaci včetně státních institucí (například soukromé společnosti, ministerstva, nemocnice, banky a další). ISMS se stará o minimalizaci bezpečnostních rizik a o jejich prevenci. Tento systém je nutné průběžně udržovat a inovovat jeho bezpečnostní opatření. [5] [6] [7] [8] [9]

2.1.2 PDCA

Norma ISO/IEC 27001 mimo jiné zavádí model plánování systému ISMS pojmenovaný zkratkou PDCA (Plan-Do-Check-Act), v češtině přeloženo jako Plánuj-Dělej-Kontroluj-Jednej. Jak je z názvu patrné, je tento model rozdělen na 4 etapy. Schéma tohoto postupu popisuje následující obrázek. [5] [9] [10]



Obrázek 2 Schéma PDCA, postupu pro zavedení ISMS [5]

2.1.2.1 Plánuj

Úkolem je stanovit cíle, požadavky a rozsah systému ISMS a definovat jeho politiku. Definují se systematické přístupy a procedury pro ohodnocování, identifikaci a řešení bezpečnostních rizik. [5] [10]

2.1.2.2 Dělej

Zde se jedná o zavedení a používání systému sestaveného podle předchozího kroku. Dochází k implementaci zvládání rizik a bezpečnostních opatření a v neposlední řadě i k řízení provozu a zdrojů ISMS. [5] [10]

2.1.2.3 Kontroluj

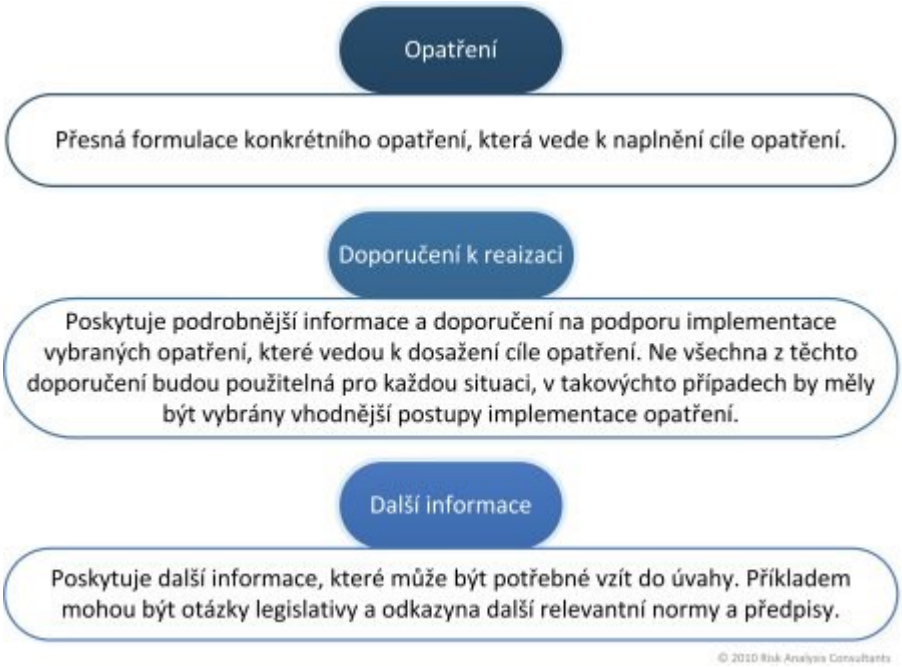
Zabývá se monitorováním a přezkoumáním řízení a účinnosti systému ISMS a ostatních bezpečnostních opatření. Provádí se audity a aktualizace bezpečnostních plánů ISMS. [5] [10]

2.1.2.4 Jednej

V tomto kroku se provádí aplikace preventivních a nápravných akcí pro vylepšení ISMS založených na výsledcích monitorování získaných v předchozí etapě. [5] [10]

2.2 ISO/IEC 27002

Další normou je norma s označením ISO/IEC 27002. Tato norma obsahuje soubor nejlepších praktik z oboru bezpečnosti informací a může také posloužit jako seznam pro kontrolu všeho správného, co je nezbytné udělat pro bezpečnost informací v organizaci. Zmíněné praktiky jsou rozděleny do 14 hlavních oddílů, které definují cíle kontrolních opatření. Jak vypadá struktura opatření je znázorněno na obrázku číslo 3. [11] [12] [13]



Obrázek 3 Popis struktury opatření obsažených v normě ISO/IEC 27002 [11]

Norma organizacím nepřikazuje, která opatření musí být nutně aplikována, ale ponechává jim volnost v rozhodnutí. Organizace pak vybírají opatření podle hodnocení rizik. Díky tomuto přístupu nabízí norma organizacím velkou flexibilitu při implementaci. V případě certifikace to však může být problém, jelikož může být složité posoudit, zda aplikovaná bezpečnostní opatření jsou plně v souladu s normou. [11] [12] [13]

2.3 Další normy

Součástí souboru norem ISO/IEC 27000 jsou další desítky různých norem z oblasti bezpečnosti informací. Pro všechny tyto normy rezervovala organizace ISO celou sérii 27000. [3] [7]

Pro příklad jsou některé z dalších norem uvedeny v následující tabulce:

Označení normy	Stručný popis
ISO/IEC 27009	Obsahem jsou definice požadavků pro používání ISO/IEC 27001 ve specifických odvětvích.
ISO/IEC 27021	Stanovuje požadavky na odbornou způsobilost bezpečnostních expertů z oblasti ISMS.
ISO/IEC 27035	Pojednává o řízení incidentů bezpečnosti informací.
ISO/IEC 27038	Obsahem jsou doporučení pro publikování digitálních dokumentů.
ISO/IEC 27040	Obsahem jsou doporučení pro bezpečné ukládání dat.
ISO/IEC 27042	Obsahem jsou doporučení pro analýzu a vyhodnocení digitálních důkazů.

Tabulka 1 Výčet některých dalších norem ze souboru norem ISO/IEC 27000 [3]

2.4 Vztah norem k penetračním testům

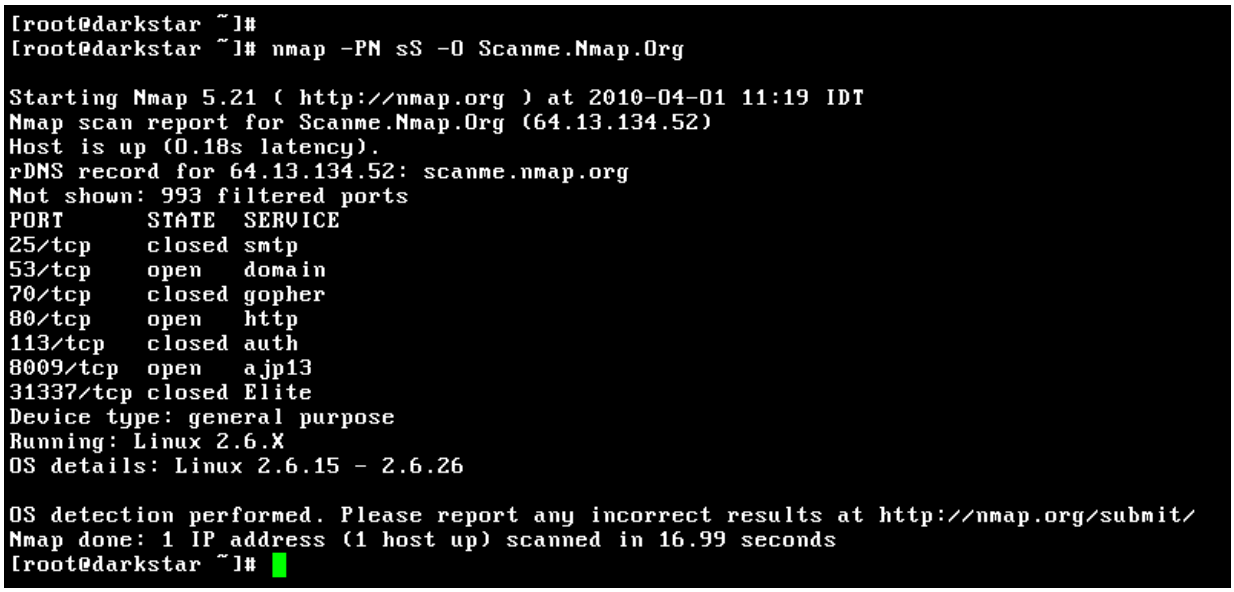
V rámci ISMS a s tím související normy ISO/IEC 27001 je jedním z kroků zavedení ISMS kontrola nastavených bezpečnostních opatření. Způsoby provedení této kontroly nejsou normou přímo dány, a proto v kontextu penetračního testování tyto normy přímo neudávají konkrétní požadavky na provádění penetračních testů. Neurčují ani obsah či strukturu výsledné zprávy (reportu) z průběhu penetračního testování. [5]

3 NÁSTROJE PRO PENETRAČNÍ TESTOVÁNÍ

V případě penetračního testování využívají bezpečnostní experti všemožných nástrojů, které jim v některých ohledech zjednoduší jejich práci. Tyto nástroje jsou většinou z části nebo plně automatizované a není třeba se jim v průběhu běhu aktivně věnovat. Výstupy z těchto nástrojů, mimo konzolový výstup, mají formát XML nebo JSON, a proto je jednoduché s nimi pracovat. V následující části jsou uvedeny některé nejpoužívanější penetrační nástroje a jejich přednosti.

3.1 Nmap

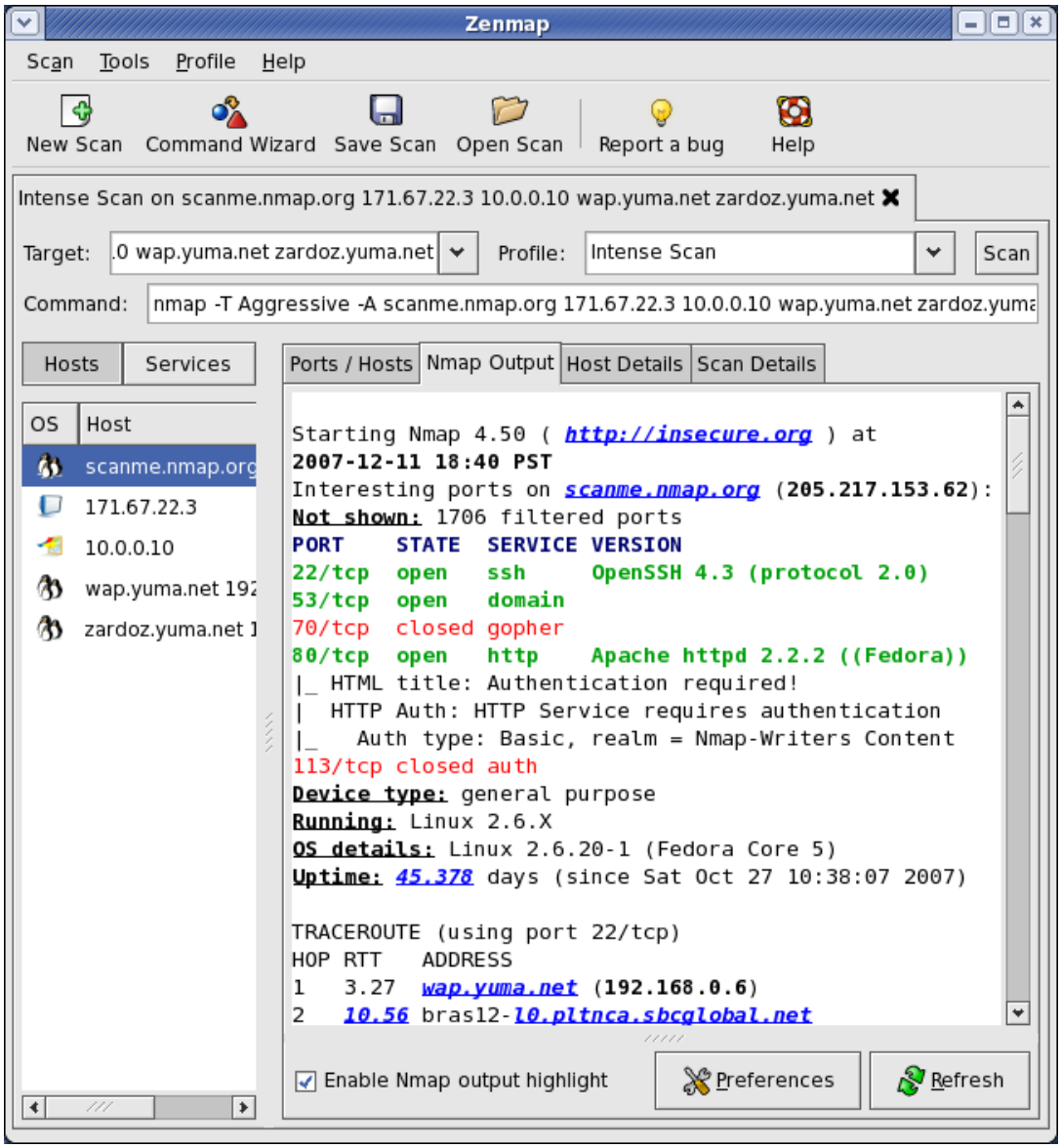
Nmap je open source nástroj pro penetrační testování a bezpečnostní audit. Jeho název vznikl jako zkratka anglického spojení „Network Mapper“ (doslova přeloženo jako mapovač sítě), z čehož plyne i hlavní funkce tohoto nástroje, což je skenování síťového prostoru. Tento nástroj je schopný vyhledávat zařízení v síti a skenovat stav jejich TCP i UDP portů v plném rozsahu (0 až 65535), na kterých zjistí podrobnosti o běžících službách. Dále je možné získat informace o operačním systému běžícím na těchto zařízeních včetně doby běhu. Nmap je distribuován zdarma, ale pouze jako konzolová aplikace běžící na systémech Windows, Linux a Mac OS. Jeho nastavení a spouštění se provádí přes příkazové přepínače. Obrázek číslo 4 ukazuje příklad konzolového výstupu. Nmap nabízí kromě konzolového výstupu také výstup v podobě XML nebo JSON formátu. [14]



Obrázek 4 Příklad konzolového výstupu nástroje Nmap [14]

3.1.1 Zenmap

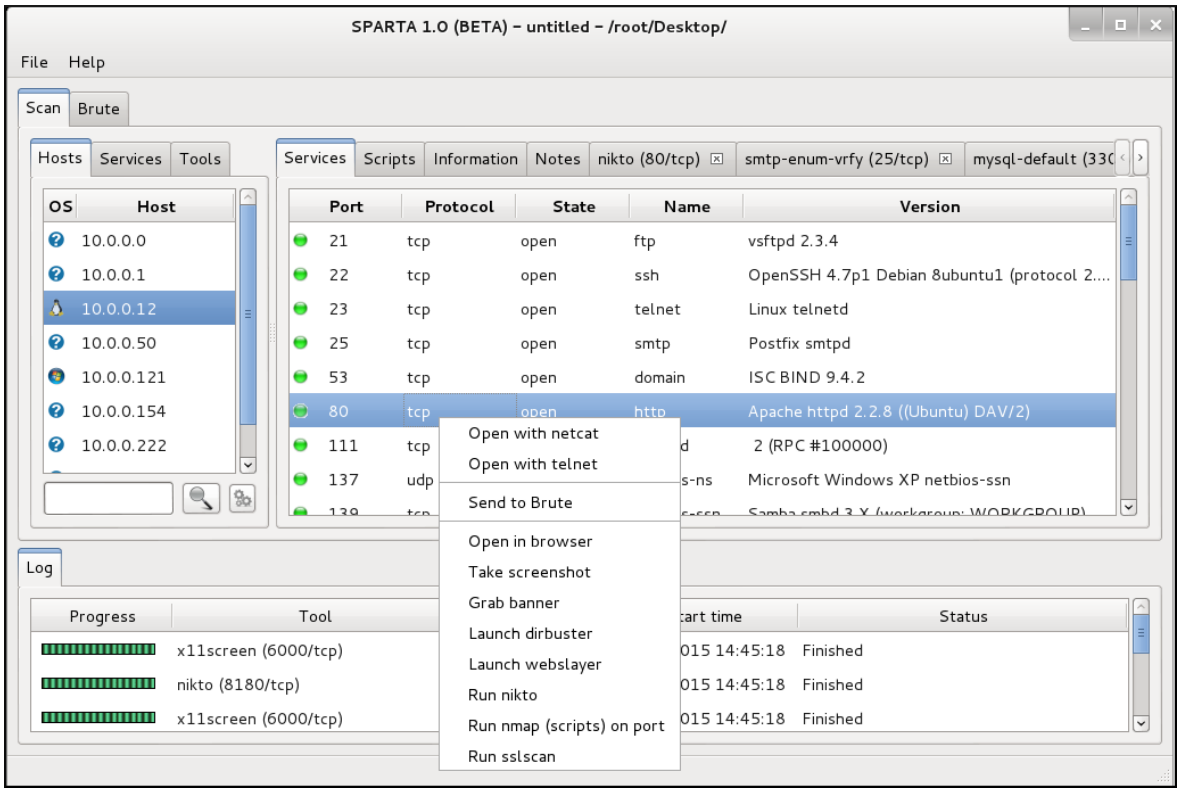
V případě nutnosti grafického uživatelského rozhraní je možné sáhnout po nástroji „Zenmap“, což je pouze grafická nadstavba pro konzolový Nmap. Stejně jako Nmap je multiplatformní. Jak můžeme vidět na následujícím obrázku, Zenmap vyniká zejména jednoduchým ovládáním a strukturovaným zobrazením výsledků. [15]



Obrázek 5 Příklad aplikace Zenmap (Nmap s grafickým rozhraním) [15]

3.2 SPARTA

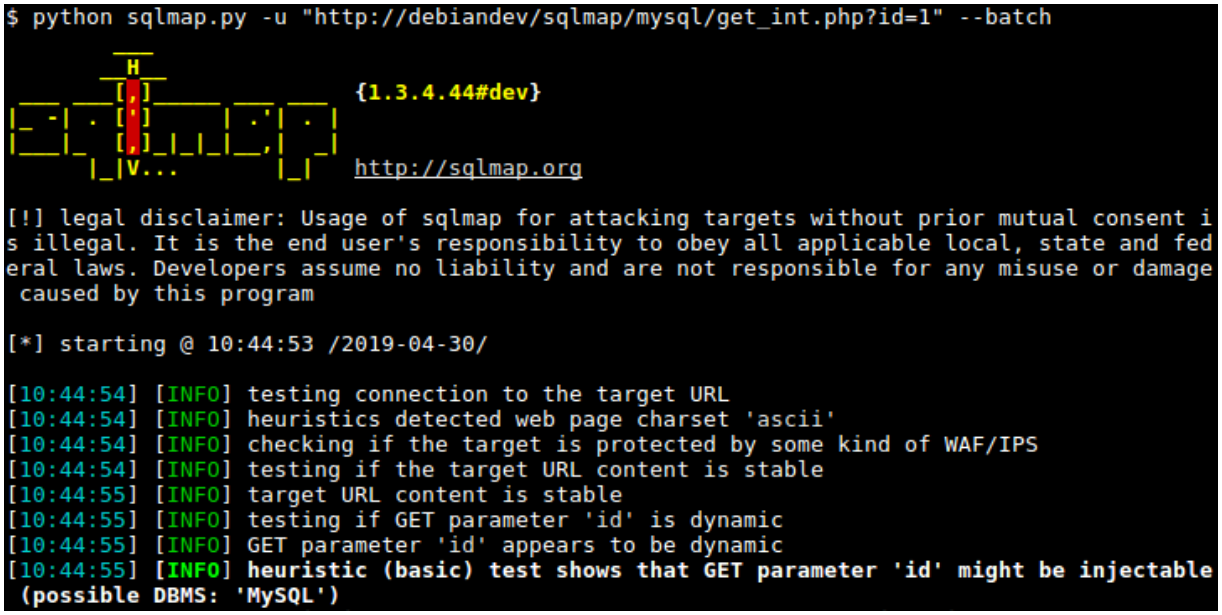
Nástroj SPARTA je aplikace s grafickým uživatelským rozhraním, která má pentesterovi zjednodušit práci s penetračním testováním síťové infrastruktury a to tím, že zjednodušuje skenování a následné procházení výsledků. Dovoluje pentesterovi ušetřit čas jednoduchým přístupem k jeho sadě nástrojů a zobrazením jejich výstupů v pohodlnější formě, než je konzolový výstup. Grafické rozhraní nástroje je jednoduché a přehledné (viz obrázek číslo 6). V aplikaci je možné nastavit si spouštěcí konfigurace pro jednotlivé nástroje pentestera a není tedy nutnost tyto nástroje nastavovat při každém vykonávání penetračního testu. Mezi nástroje, které se běžně používají ve spojení s nástrojem SPARTA, patří např. výše zmíněný Nmap, nikto nebo hydra. Těchto nástrojů můžou být desítky až stovky a je pouze jen na pentesterovi, které nástroje bude pomocí nástroje SPARTA spouštět. [16]



Obrázek 6 Přiklad nástroje SPARTA pro penetrační testování [16]

3.3 Sqlmap

Sqlmap je další z penetračních nástrojů, který je také open source. Navíc je také multiplatformní, neboť je celý psán v interpretovaném jazyce Python. Tento nástroj se zaměřuje na detekci a využití zranitelností typu „SQL injection“. To provádí pomocí tzv. http dotazů (např. GET nebo POST). Mimo to je nástroj schopen přistupovat do zranitelných databází a umožňuje editaci databázových tabulek včetně možnosti smazání záznamů. Sqlmap podporuje všechny běžně používané databázové systémy včetně MySQL, Oracle, MS SQL a další. Ovládání tohoto nástroje je prováděno přes konzoli, jak můžeme vidět na následujícím obrázku. Výstup tohoto nástroje je pouze konzolový výpis, který je možné uložit do textového souboru. [17]

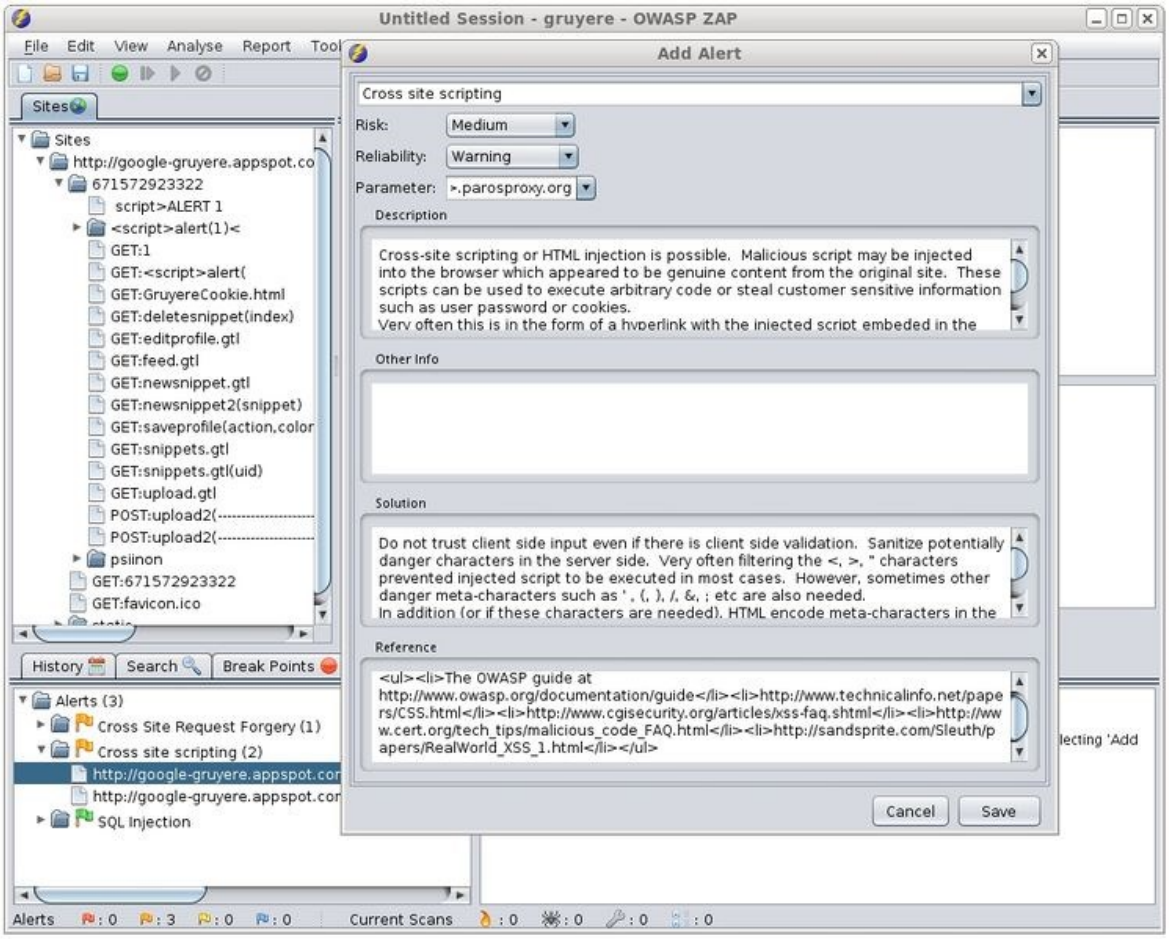


Obrázek 7 Příklad běhu penetračního nástroje Sqlmap [17]

3.4 OWASP ZAP

OWASP Zed Attack Proxy (ZAP) je jeden z dalších světově populárních bezpečnostních nástrojů. I tento je šiřitelný zdarma a je aktivně udržován početnou skupinou vývojářů z oblasti informační bezpečnosti. Nástroj disponuje komplexním grafickým rozhraním (viz obrázek číslo 8). S pomocí tohoto nástroje je možné automaticky odhalovat zranitelné chyby

v zabezpečení webových aplikací, a to i během vývoje dané aplikace. Pentesteři jej také používají pro ruční testování během provádění penetračního testu. [18]



Obrázek 8 Příklad grafického rozhraní nástroje OWASP Zed Attack Proxy [18]

3.5 Automatizované spouštění nástrojů

Jednou z dalších možností, jak si může pentester ulehčit práci, je automatizované spouštění nástrojů pro penetrační testování. Pro tyto potřeby je nutné, aby penetrační nástroje byly na toto spouštění již připravené a měly minimálně konzolové rozhraní pro spouštění. Co se týče výstupů, některé penetrační nástroje nabízí pouze konzolový výstup, který je pro strojové zpracování složitý. Jiné nástroje zas nabízí uložení výsledků do standartních souborových formátů jako jsou XML a JSON, kde je strojové zpracování jednoduché a práci s těmito typy výstupů podporuje většina programovacích jazyků. [14] [16] [17] [18]

Pro shrnutí možností penetračních nástrojů poslouží následující tabulka:

Název	**Podpora OS**	**Možnost automatizace**	**Formát výstupu**
Nmap	Windows, Linux, MacOS	ANO	XML, JSON
Sparta	Linux	NE	SQLite databáze
Sqlmap	Linux	NE	Konzolový výstup
OWASP ZAP	Windows, Linux, MacOS	NE	XML, JSON

Tabulka 2 Srovnání penetračních nástrojů [14] [16] [17] [18]

II. PRAKTICKÁ ČÁST

4 POŽADAVKY APLIKACE

Jak už bylo řečeno v úvodu, cílem této práce je vytvoření desktopové aplikace pro zjednodušení práce pentestera. Aplikace bude umožňovat automatizované spouštění a zpracování výstupů z vybraných penetračních nástrojů. Výsledné zpracované výstupy budou strukturovány takovým způsobem, aby je pentester mohl jednoduše použít a nemusel ručně procházet dlouhé logovací soubory z penetračních nástrojů. Tím se pentesterovi zkrátí doba, po kterou bude vytvářet penetrační report. Pro případné strojové načítání do penetračního reportu budou data uložena ve formátech určených pro předávání dat, a to konkrétně ve formátu XML a JSON.

Od funkce aplikace se odvíjí i její požadavky. Základním požadavkem je zpracování běžných výstupů penetračních nástrojů do zjednodušené podoby ve formátu vhodném pro další strojové zpracování. Dalším požadavkem je, aby aplikace byla multiplatformní, a tedy aby běžela na různých operačních systémech. Je to hlavně z toho důvodu, že pentester často střídá operační systém, na kterém pracuje. Mezi další požadavky lze zařadit intuitivní a jednoduché ovládání.

4.1 Požadavky pro jednotlivé penetrační nástroje

Vzhledem k možnostem automatizace a formátu výstupu penetračních nástrojů (viz tabulka č. 2 v kapitole 3.5) byly pro tuto aplikaci vybrány pouze některé nástroje. Pro automatizované spouštění byl vybrán pouze nástroj Nmap, jelikož umožňuje jednoduché strojové ovládání přes konzolové rozhraní. V případě zpracování výstupů byly vybrány nástroje Nmap, OWASP ZAP a Sparta, a to hlavně kvůli formátu jejich výstupů. Tím, že se jedná o různé nástroje jsou níže sepsány požadavky pro každý nástroj zvlášť.

4.1.1 Nmap

- zpracovávat XML výstup
- získat verzi Nmapu, jeho nastavení, čas, datum a délku trvání testu
- získat IP adresu cíle a otevřené porty včetně běžících služeb
- získat MAC adresu cíle a výrobce zařízení
- identifikovat operační systém a jeho detaily
- získat síťovou vzdálenost cíle a síťovou cestu k cíli
- získat shrnutí z testu

4.1.2 OWASP ZAP

- získat čas, datum a délku trvání testu
- získat název, popis a URL adresu každé hrozby
- získat parametry a řešení pro každou hrozbu
- získat souhrn všech hrozeb (počet hrozeb podle míry rizika)

4.1.3 Sparta

- zpracovat výstup modulu Nmap
- z modulu nikto získat IP adresu, port cíle a OSVDB kódy

5 VÝVOJOVÉ NÁSTROJE DESKTOPOVÝCH APLIKACÍ

Pro vývoj desktopových aplikací existuje v dnešní době mnoho jazyků a knihoven. Každý jazyk má své kladné a záporné stránky, a proto je výběr vhodného jazyka na začátku projektu velmi důležitý a může nám v různých směrech zjednodušit samotný vývoj aplikace. Některé jazyky jsou určeny pro vývoj pouze na jedné platformě, jiné se zas vydaly směrem multiplatformního vývoje.

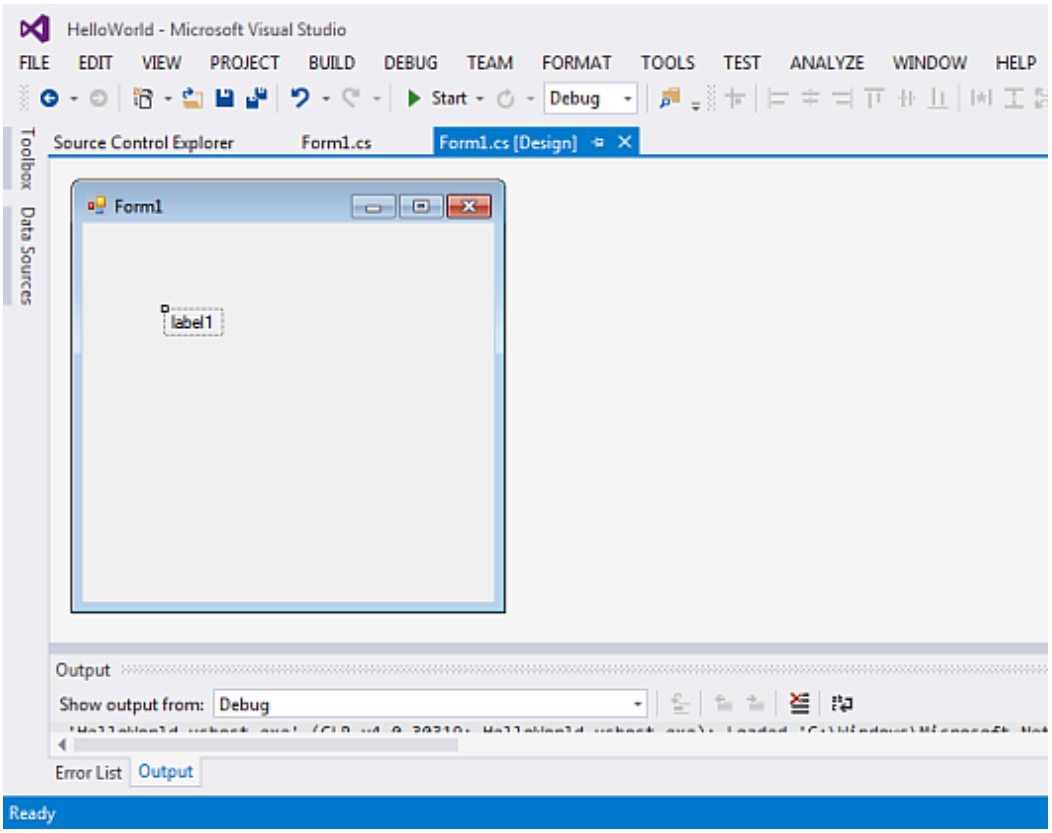
5.1 C# a .Net

C# je programovací jazyk vyvíjen společností Microsoft. Jedná se o vysokoúrovňový čistě objektově orientovaný jazyk. Spolu s tímto jazykem úzce souvisí i .Net framework, ze kterého vychází většina vlastností jazyka. Jako jediný z vybraných programovacích jazyků neumožňuje vývoj multiplatformních aplikací. Jednou z velkých výhod jazyka je jeho jednoduchost na naučení a relativně rychlý vývoj desktopových aplikací. To je částečně způsobeno i jednoduchou syntaxí jazyka (viz obrázek číslo 9). Co se týká vzhledu grafických prvků pro vývoj aplikace s grafickým uživatelským rozhraním (dále jen GUI), tak se jedná o starší knihovnu WindowsForms, a proto i vzhled celé aplikace je na dnešní dobu již zastaralý a koresponduje s motivy operačních systémů firmy Microsoft, a to konkrétně verze Windows Vista a Windows 7. [19] [20]

```
// A Hello World! program in C#.
using System;
namespace HelloWorld
{
    class Hello
    {
        static void Main()
        {
            Console.WriteLine("Hello World!");

            // Keep the console window open in debug mode.
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
    }
}
```

Obrázek 9 Syntaxe jazyka C# na příkladu „Hello world“ [21]

Budeme-li se zabývat vývojovým prostředím pro jazyk C#, tak se pro programování používá převážně vývojové prostředí Microsoft Visual Studio, které obsahuje nepřeberné množství nástrojů pro ulehčení práce programátora. Ukázka tohoto vývojového prostředí je na následujícím obrázku. [22]



Obrázek 10 Vzhled platformy Windows forms ve vývojovém prostředí Microsoft Visual Studio [23]

5.2 Java

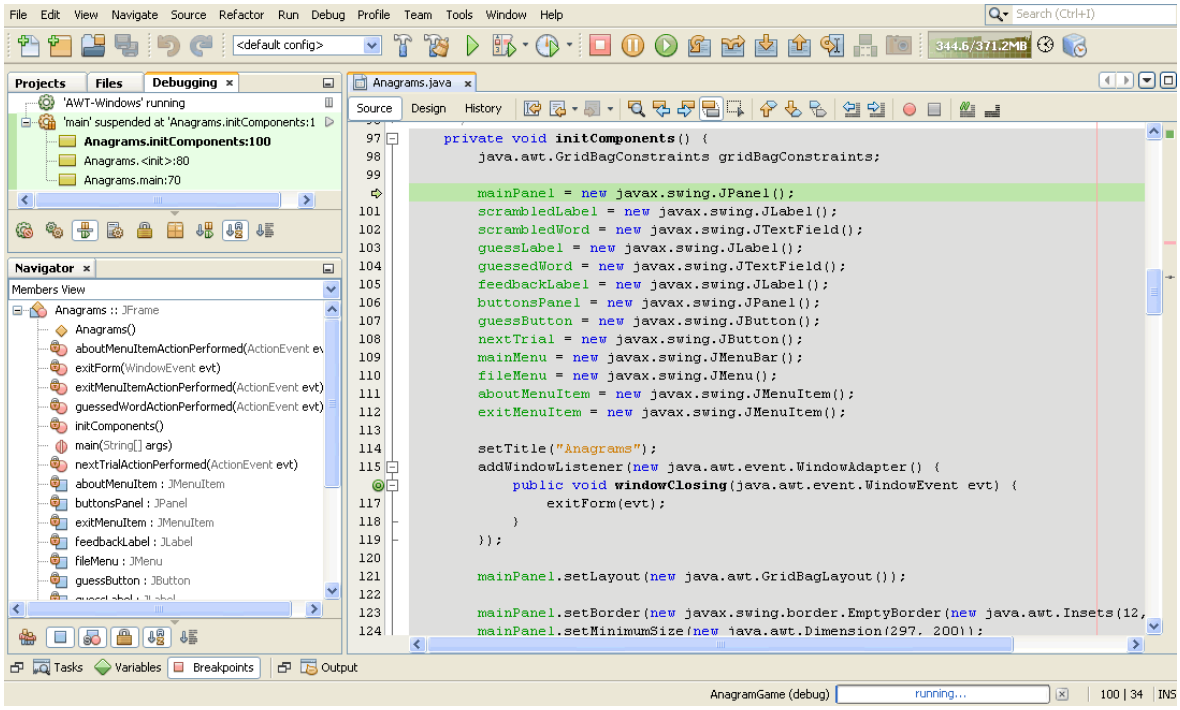
Java je další z objektově orientovaných jazyků. Program vytvořený v tomto jazyce je, díky vlastnímu interpreteru Java Virtual Machine (JVM), schopen běžet na většině zařízení, které podporují tento interpreter. Mezi tyto zařízení patří stroje s operačním systémem Windows, Linux ale i MacOS a je tedy možné označit Javu za multiplatformní. Za zmínku stojí i fakt, že je Java od roku 2007 vyvíjena jako open source. Co se týká GUI, je možné, díky otevřenosti kódu, používat větší množství grafických knihoven. Jednou z těch známějších je kni-

hovna JavaFX, která pochází přímo od vývojáře samotné Javy. V rámci přehlednosti zdrojového kódu je na tom Java hůře než C#, její syntaxe je u některých prvků složitá a zmatená (viz následující obrázek), ale i přesto je velmi populární v komerčním prostředí. [24] [25] [26]

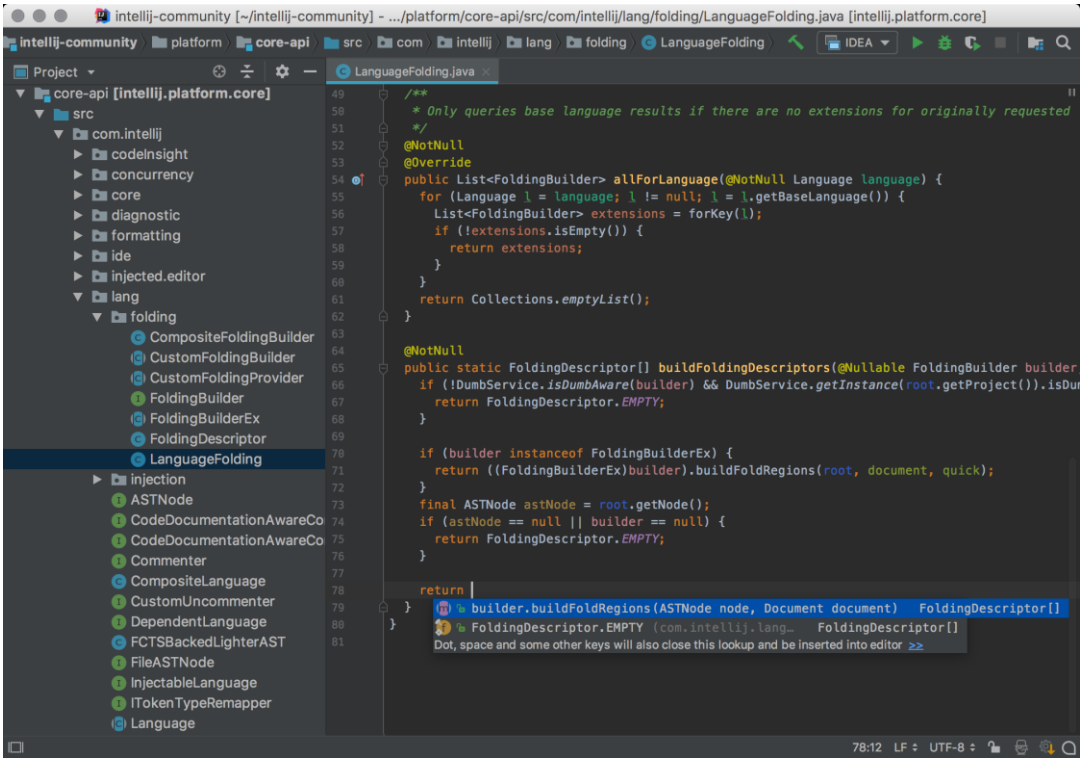
```
public class HelloWorld {
  public static void main (String args[]) {
    System.out.println ("Hello world!");
  }
}
```

Obrázek 11 Syntaxe jazyka Java na příkladu „Hello world“ [27]

V čem má Java oproti C# navrch, jsou vývojové nástroje pro tvorbu aplikací. Díky své oblíbenosti je Java podporována vícero nástroji, mezi které spadá například Eclipse, NetBeans nebo stále oblíbenější IntelliJ IDEA. Rozdíly mezi těmito prostředími jsou z hlediska funkcí minimální, liší se však vzhledem. Například vývojové prostředí NetBeans disponuje postarším vzhledem zatímco prostředí IntelliJ IDEA využívá moderního vzhledu, a toto lze vidět na obrázcích číslo 12 a 13. [28] [29] [30]



Obrázek 12 Příklad vývojového prostředí NetBeans [31]



Obrázek 13 Příklad vývojového prostředí IntelliJ IDEA [30]

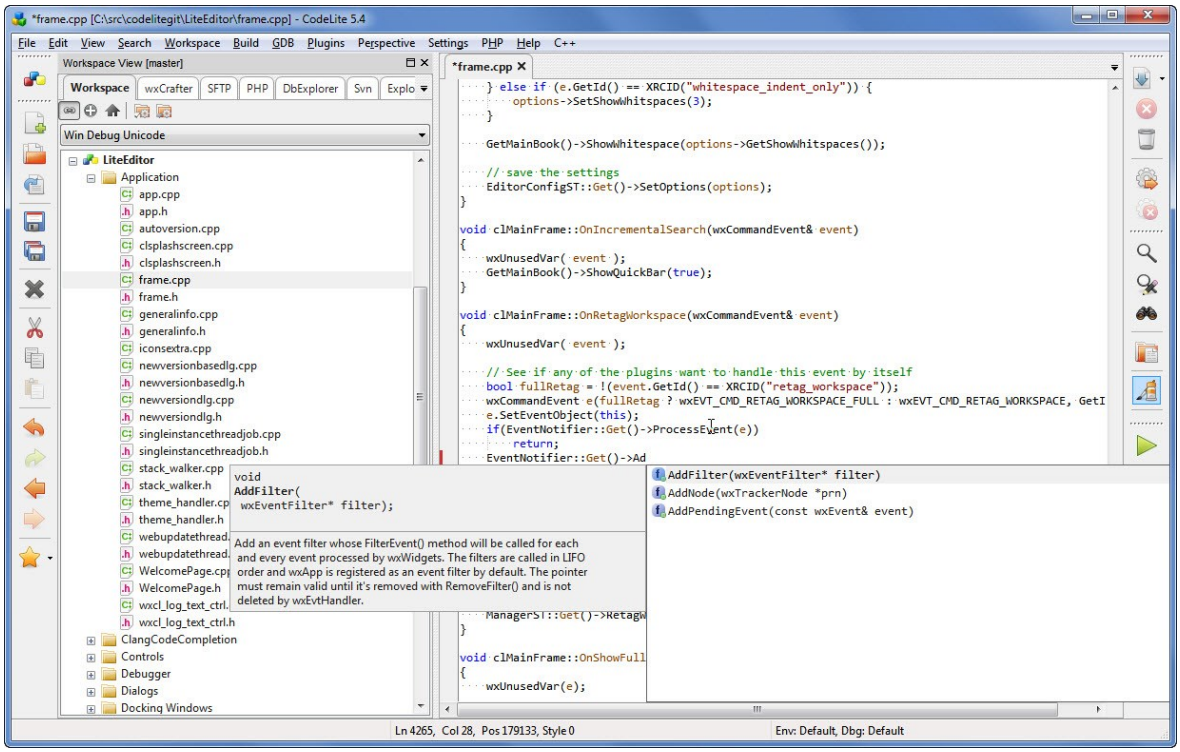
5.3 C++

Jazyk C++ je jeden z nejstarších programovacích jazyků, které umožňují objektově orientovaný vývoj aplikací. I přes svoje stáří je tento jazyk stále hojně používán a jeho vývoj nadále probíhá. Jak již název napovídá, vychází jazyk C++ z jazyka C, ale nejsou mezi sebou zcela kompatibilní. Jak si lze povšimnout z obrázku číslo 14, jazyk C++ používá syntaxi jazyka C. C++ je mezi uvedenými jazyky jediný kompilovaný jazyk a je také nejtěžší na naučení. To je způsobeno převážně nutností manuální správy paměti, která je u předešlých dvou jazyků zcela automatická. Manuální správa paměti má však své výhody, jako jsou nižší nároky na běh programu a rychlejší běh aplikací, proto je jazyk C++ dále vyvíjen a používán. Tím, že je C++ kompilovaný jazyk, je možné jej zkompilovat na různé platformy (záleží na použití kompilátoru) a můžeme tedy označit jazyk C++ za multiplatformní. V případě, že chceme vyvíjet aplikaci v C++ s GUI, je vzhledem k rozšířenosti jazyka na výběr velké množství grafických knihoven. Některé knihovny mají podporu pro multiplatformní vývoj, jiné jsou zase zaměřeny pouze na jednu platformu. Mezi známé GUI knihovny pro C++ patří například wxWidgets, GTK nebo Qt. [32] [33] [34]

```cpp
#include <iostream>

int main(int argc, char *argv[]) {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Obrázek 14 Syntaxe jazyka C++ na příkladu „Hello world“

Komunita jazyka C++ je velmi rozsáhlá, proto i nabídka vývojových prostředí s podporou C++ je rozmanitá. Od základních, které postačí pro vývoj menšího projektu, kterými jsou například CodeLite (znázorněn na obrázku č. 15) nebo Code::Blocks, až po pokročilejší, kde lze zmínit například NetBeans, CLion nebo i Qt Creator. [29] [35] [36] [37] [38]



Obrázek 15 Příklad vývojového prostředí CodeLite [35]

5.4 Qt Framework

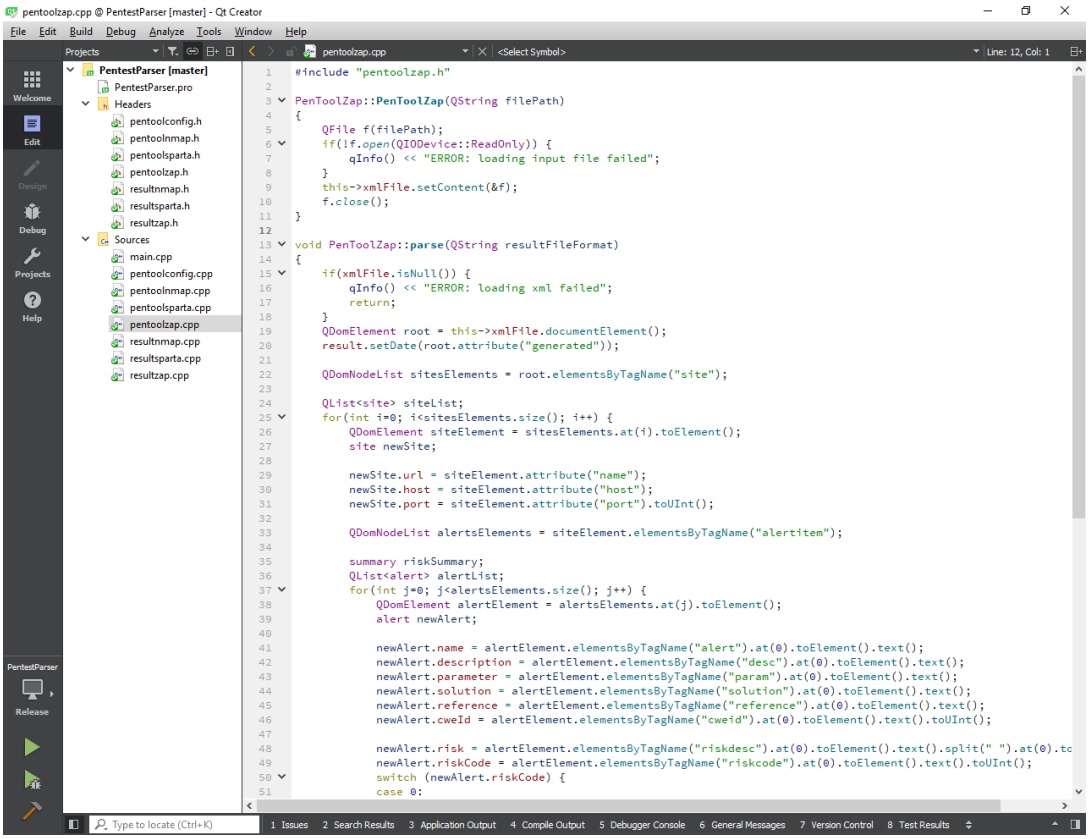
Qt framework je častěji označován zkráceně pouze jako „Qt“ a jedná se o framework jazyka C++ pro vývoj multiplatformních aplikací. Syntaxe je zde stejná jako v případě jazyka C++ (viz obrázek č. 16). Pomocí Qt lze vyvíjet aplikace jak konzolové, tak i s grafickým uživatelským prostředím. Grafické aplikace používají nativní vzhled podle operačního systému.

Samotný framework obsahuje nepřeberné množství vlastních knihoven a v průběhu vývoje aplikace není tedy potřeba používat knihovny třetích stran. V těchto knihovnách nechybí nástroje pro práci se soubory (včetně zpracování XML a JSON formátů), databázemi (SQL, MySql, SQLite a další) anebo nástroje pro správu vláken a vícevláknových procesů. Qt obsahuje i vlastní datové typy včetně datových struktur jako jsou pole, lineární seznamy a další. Qt je hojně používáno v oblasti mikropočítačů, kde se s jeho pomocí vytváří například interaktivní přístrojové desky nebo infotainment systémy vozidel. Mezi velké výhody Qt patří zejména jednoduchost vytváření GUI anebo jednoduchá a velmi přehledná dokumentace. [39] [40] [41]

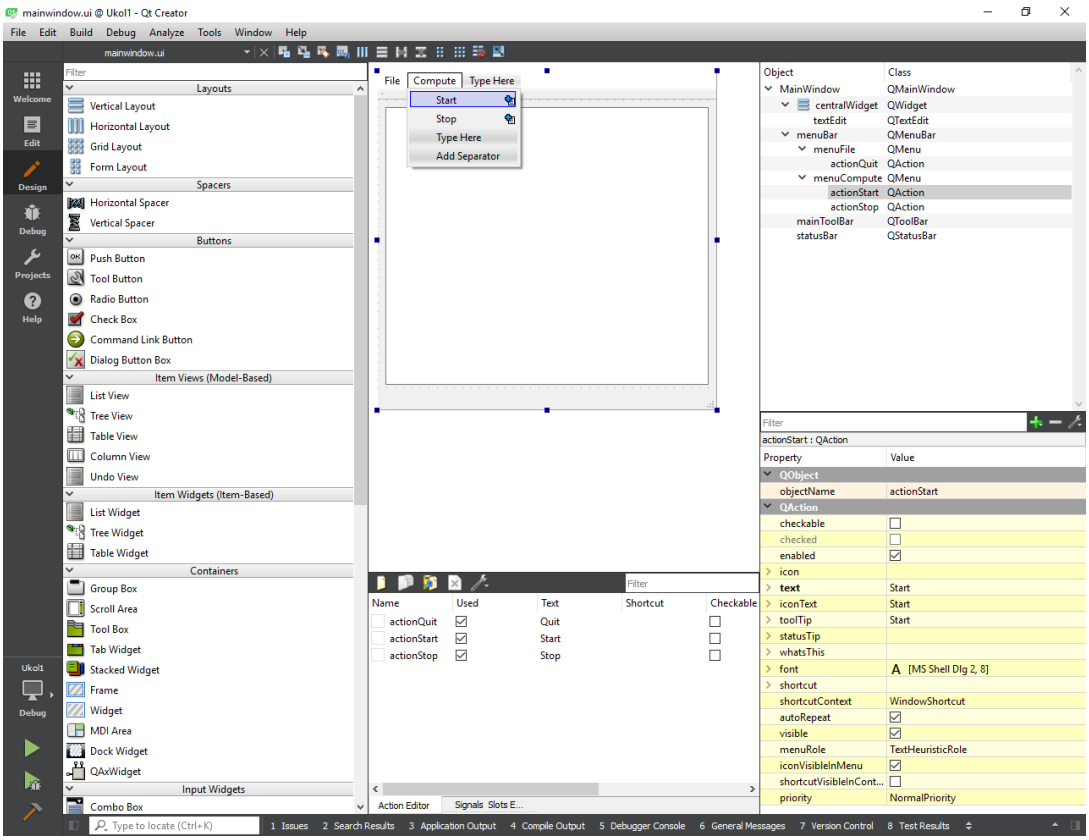
```
#include <QApplication>
#include <QPushButton>

int main(int argc, char **argv)
{
 QApplication app (argc, argv);

 QPushButton button ("Hello world !");
 button.show();

 return app.exec();
}
```

Obrázek 16 Syntaxe Qt na příkladu „Hello world“

Společnost Qt vytvořila pro vývoj aplikací vlastní vývojové prostředí nazývané Qt Creator, které je v nástrojích pro ulehčení práce programátora na podobné úrovni jako MS Visual Studio pro jazyk C#. Qt Creator je univerzální vývojové prostředí a lze jej použít i pro vývoj mimo Qt a pro svou komplexnost se stává čím dál více oblíbeným. Součástí Qt Creatoru je takzvaný Qt Designer, který může být spuštěn samostatně a který slouží pro návrh grafického rozhraní aplikací. Oba tyto nástroje jsou vyobrazeny na následujících dvou obrázcích. [38] [42] [43]



Obrázek 17 Příklad vývojového prostředí Qt Creator



Obrázek 18 Příklad nástroje Qt Designer integrovaného do Qt Creatoru

6 FORMA VÝSTUPNÍCH DAT

Pro aplikace zpracovávající různá data, kde výstupem má být soubor s obsahem zpracovaných dat, je důležité vybrat vhodný formát, v jakém budou výstupní data uložena. Tento výběr je nezbytný, víme-li, že budou data následně strojově zpracována v jiné aplikaci. Pro tyto účely bylo vytvořeno několik způsobů formátování dat, mezi kterými jsou nejznámější a nejpoužívanější XML a JSON.

6.1 XML

Zkratka XML znamená „eXtensible Markup Language“, což lze volně přeložit jako „rozšiřitelný značkovací jazyk“. Jak už název napovídá, jedná se o značkovací jazyk. Text nebo různé údaje se totiž vkládají za značky, takzvané tagy, které jsou z obou stran doplněné o ostré závorky. V případě ukončovacího tagu, který ukončuje aktuální tag, je za první závorkou ještě lomítko. XML je dost často používán pro serializaci různých dat, a proto je podporován velkým množstvím nástrojů a programovacích jazyků. Setkat se s XML můžeme převážně u výstupních dat (viz obrázek 19) a konfiguračních souborů aplikací. [44] [45]

```
<?xml version="1.0" encoding="UTF-8"?>
<hosts>
  <host ipAddr="10.56.8.65" uptime="0" networkDistance="3">
    <ports>
      <port id="22" protocol="tcp" state="open" service="OpenSSH" serviceName="ssh"/>
      <port id="80" protocol="tcp" state="open" service="Apache httpd" serviceName="http"/>
      <port id="443" protocol="tcp" state="open" service="Apache httpd" serviceName="http"/>
    </ports>
    <os>
      <osMatch name="Linux 2.6.32" accuracy="87"/>
      <osMatch name="HP P2000 G3 NAS device" accuracy="86"/>
      <osMatch name="OpenWrt 0.9 - 7.09 (Linux 2.4.30 - 2.4.34)" accuracy="85"/>
      <osMatch name="OpenWrt White Russian 0.9 (Linux 2.4.30)" accuracy="85"/>
      <osMatch name="OpenWrt Kamikaze 7.09 (Linux 2.6.22)" accuracy="85"/>
    </os>
    <trace>
      <hop ipAddr="192.168.10.1" rtt="2"/>
      <hop ipAddr="192.168.157.1" rtt="286"/>
      <hop ipAddr="10.56.8.65" rtt="287"/>
    </trace>
  </host>
</hosts>
```

Obrázek 19 Příklad výstupních dat ve formátu XML

6.2 JSON

Formát JSON je další z datových formátů. Jeho zkratka znamená „JavaScript Object Notation“, což v překladu znamená „JavaScriptový objektový zápis“. Z názvu vyplývá, že se jedná o formát, který přišel s jazykem JavaScript, ale i přesto je tento formát jazykově nezávislý. Často je používán v oblasti webových aplikací pro předávání dat mezi serverem a prohlížečem, většinou se s ním můžeme setkat u výsledků, které poskytují různá webová API (např. počasí, geolokační data a další). Data v JSON jsou na rozdíl od XML formátována jako pár atribut-hodnota, což lze vidět na obrázku níže. Co se týká podpory pro čtení a zápis JSON formátu, tak je tato situace podobná jako u XML, to znamená, že většina běžně používaných programovacích jazyků práci s JSON formátem plně podporuje. Někdy je formát JSON označován jako „odlehčená“ verze formátu XML. [46] [47] [48]

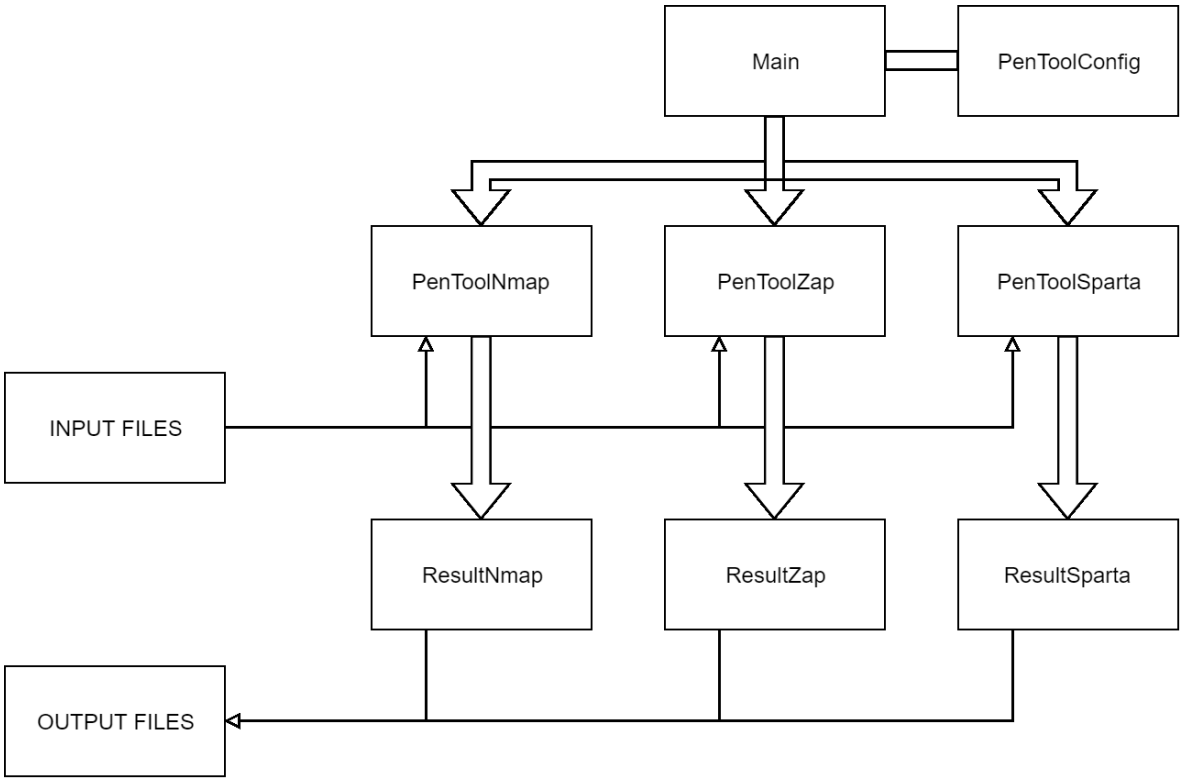
Obrázek 20 Příklad výstupních dat ve formátu JSON [49]

7 IMPLEMENTACE APLIKACE

Vzhledem k požadavkům byla aplikace vyvinuta jako konzolová aplikace a jako nástroj pro vývoj posloužil Qt framework s jazykem C++. Je to hlavně z důvodu multiplatformního přístupu a minimální potřeby knihoven třetích stran.

7.1 Struktura aplikace

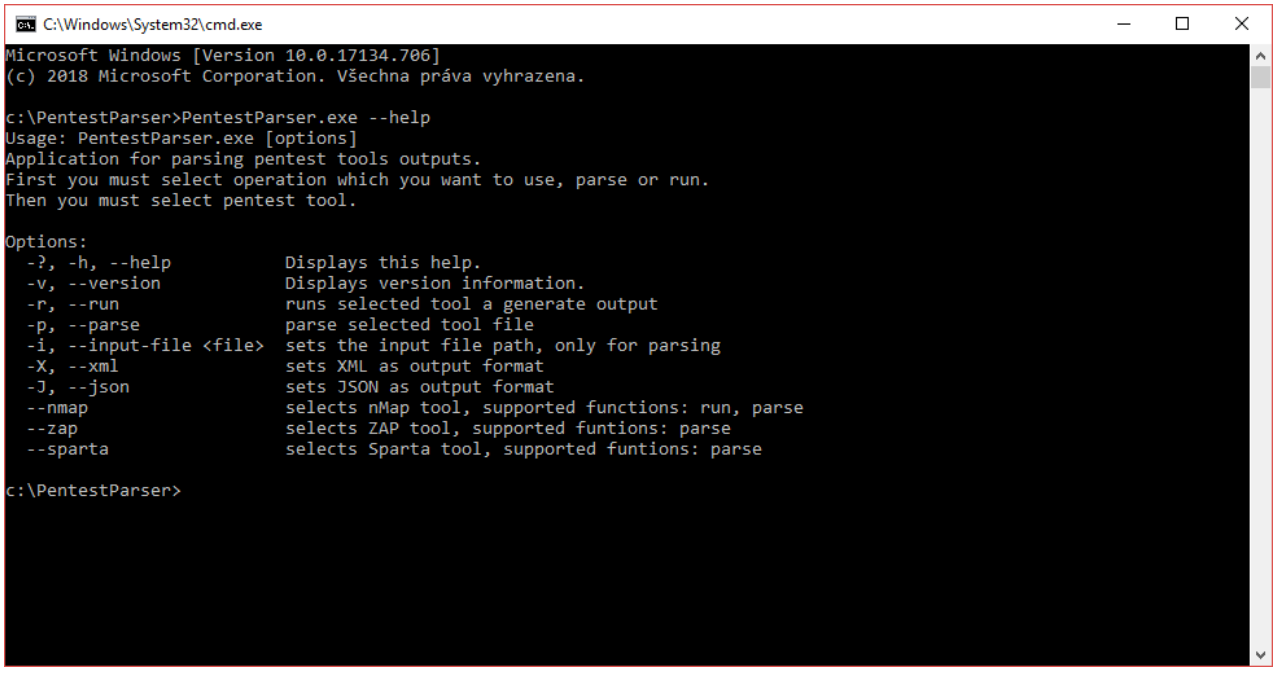
Aplikace je složena ze sedmi tříd a jednoho souboru obsahujícího vstupní funkci programu „main“. Třídy jsou dále rozděleny na tři skupiny. První skupina obsahuje pouze jednu třídu, která zajišťuje nastavení aplikace pro provádění automatizovaného spouštění penetračních nástrojů. Název této třídy je „PenToolConfig“. Další skupinou jsou třídy starající se o zpracování výstupů z penetračních nástrojů, jedná se o třídy začínající frází „PenTool“ a zbytek názvu tvoří název nástroje, jmenovitě jde o třídy „PenToolNmap“, „PenToolZap“ a „PenToolSparta“. Obdobná situace je i u poslední skupiny tříd, které mají na práci uchování zpracovaných údajů a jejich následné uložení do souborů typu XML nebo JSON. Název těchto tříd je podobný jako u minulé skupiny, liší se pouze frází na začátku názvu, v tomto případě to je fráze „Result“ a jedná se tedy o třídy s názvy „ResultNmap“, „ResultZap“ a „ResultSparta“. Strukturu aplikace stručně popisuje následující obrázek.



Obrázek 21 Zjednodušená struktura aplikace

7.2 Ovládání aplikace

Vzhledem k faktu, že je aplikace čistě konzolová, je její ovládání prováděno přes konzolové rozhraní operačního systému. Na následujícím obrázku je v konzoli znázorněna nabídka nápovědy aplikace, která stručně popisuje, jak se má s touto aplikací pracovat.



Obrázek 22 Nabídka nápovědy v konzolovém okně aplikace

O kompletní ovládání aplikace se stará třída stvořená pro tyto účely s názvem „QCommandLineParser“, která zajišťuje automatické zpracování vstupních parametrů aplikace. Pomocí této třídy se také nastaví popis a verze aplikace, které je možné v aplikaci zobrazit pomocí patřičných přepínačů (například „-v“ nebo „--version“ zobrazí verzi aplikace).

Pro jednotlivé parametry spouštění neboli přepínače se pak používá třída „QCommandLineOption“, pomocí které se vytvoří zmíněné přepínače aplikace. Nápověda aplikace je pak sama vygenerována zmíněnou třídou „QCommandLineParser“ z vytvořených přepínačů.

Celé nastavení a vytvoření tohoto zjednodušeného konzolového menu je obsaženo na začátku vstupní funkce aplikace (funkce „main“) a vypadá ve zdrojovém kódu následovně:

```
QCommandLineParser cliParser;
cliParser.addHelpOption();
cliParser.addVersionOption();
cliParser.setApplicationDescription("Application for parsing pentest tools outputs.\n"
                                    "First you must select operation which you want to use, parse or run.\n"
                                    "Then you must select pentest tool and provide input file path.\n"
                                    "It's necessary to select output file format.");

QCommandLineOption runOption(QStringList() << "r" << "run","Runs selected tool and generate output");
QCommandLineOption parseOption(QStringList() << "p" << "parse", "Parse selected tool file");

QCommandLineOption nmapOption("nmap", "Selects nMap tool, supported functions: run, parse");
QCommandLineOption zapOption("zap", "Selects ZAP tool, supported funtions: parse");
QCommandLineOption spartaOption("sparta", "Selects Sparta tool, supported funtions: parse");

QCommandLineOption inputFileOption(QStringList() << "i" << "input-file", "Sets the input file path", "file");
QCommandLineOption xmlOption(QStringList() << "X" << "xml", "Sets XML as output format");
QCommandLineOption jsonOption(QStringList() << "J" << "json", "Sets JSON as output format");

cliParser.addOptions({runOption, parseOption});
cliParser.addOptions({inputFileOption ,xmlOption, jsonOption});
cliParser.addOptions({nmapOption, zapOption, spartaOption});
cliParser.process(app);
```

Obrázek 23 Zdrojový kód přepínačů aplikace

Pro jednotlivé funkce je možné zvolit více variant přepínačů, například pro zvolení výstupního formátu XML je možné použít přepínač „-X“ nebo „--xml“. Tyto dvojí, případně trojí možnosti jsou použity skoro u všech přepínačů. Takto je to použito z důvodu, aby si uživatel sám vybral, co mu vyhovuje.

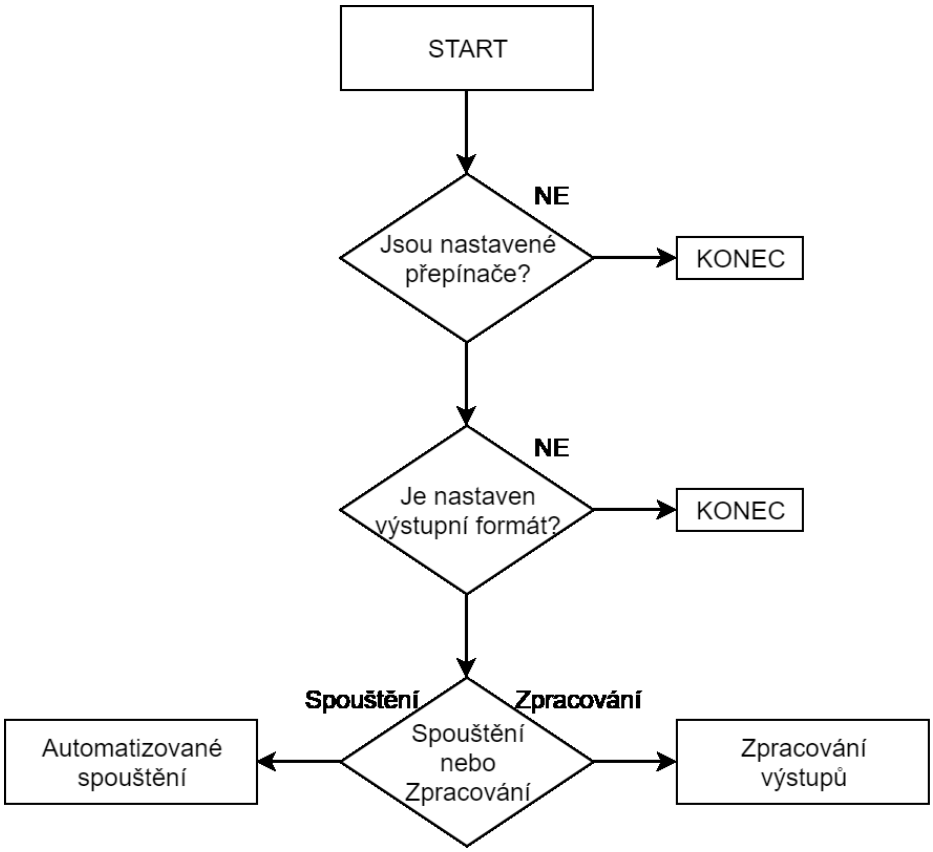
Pro přehlednost jsou v následující tabulce vypsány veškeré přepínače aplikace:

Přepínač	**Popis**
-?, -h, --help	Zobrazí nápovědu aplikace.
-v, --version	Zobrazí verzi aplikace.
-r, --run	Spustí vybraný penetrační nástroj.
-p, --parse	Zpracuje výstupní soubor z vybraného penetračního nástroje.
-i, --input-file	Zvolí soubor z penetračního nástroje pro zpracování.
-X, --xml	Nastaví XML jako výstupní formát zpracovaných dat.
-J, --json	Nastaví JSON jako výstupní formát zpracovaných dat.
--nmap	Vybere penetrační nástroj Nmap.
--zap	Vybere penetrační nástroj OWASP ZAP.
--sparta	Vybere penetrační nástroj Sparta.

Tabulka 3 Seznam přepínačů aplikace

7.3 Vstupní funkce aplikace

Vstupní funkcí aplikace je myšlena funkce, která je provedena bezprostředně po spuštění aplikace. Název této funkce je „main“ a je v ní obsažena rozhodovací část aplikace, která pracuje podle zvolených přepínačů zadaných při spouštění aplikace. Rozhodovací část funkce „main“ je stručně vyznačena na následujícím diagramu.



Obrázek 24 Zjednodušený diagram funkce „main“

První podmínka testuje, jestli vůbec byly aplikaci předány při spuštění nějaké parametry (nastaveny přepínače). Pokud ne, je vypsána na konzoli chybová hláška a následně je zavolána funkce „showHelp“ ze třídy „QCommandLineParser“, která vypíše na konzoli nápovědu aplikace a poté aplikaci validně ukončí (návratový kód je 0).

Druhá podmínka v pořadí zjišťuje, zda byl pomocí přepínačů vybrán výstupní formát, jelikož ten je požadován jak pro operaci automatizovaného spouštění, tak i pro zpracování výstupů penetračních nástrojů. Pokud není vybrán výstupní formát, je situace stejná jako u předchozí podmínky.

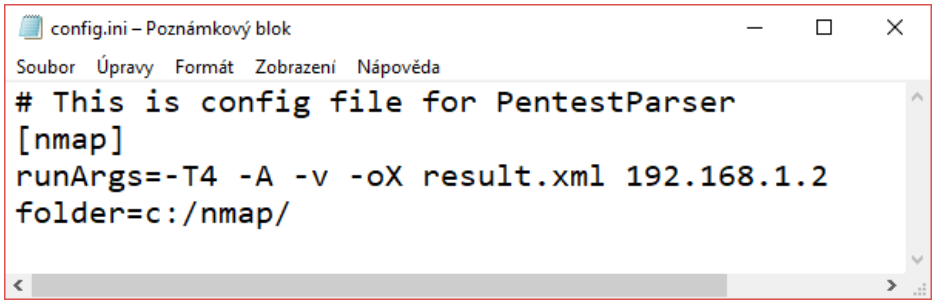
Poslední rozhodovací podmínka rozděluje aplikaci na dvě větve, první větví je automatizované spouštění penetračních nástrojů, druhou pak zpracování výstupů z penetračních nástrojů.

7.4 Automatizované spouštění penetračních nástrojů

V případě automatizovaného spouštění penetračních nástrojů se jedná o spuštění nástroje z této aplikace a po jeho doběhnutí se provede zpracování jeho výstupu. Spuštěný nástroj běží na pozadí a řídí se podle konfiguračního souboru. Díky tomu může pentester využívat pouze jedné aplikace namísto dvou.

7.4.1 Konfigurační soubor

Součástí aplikace je i konfigurační soubor pro nastavení běhu daného penetračního nástroje. O správu konfiguračního souboru se stará třída s názvem „PenToolConfig“, která provádí načtení a následné zpracování souboru pomocí třídy „QSettings“, která toto zajišťuje. Na následujícím obrázku je znázorněno, jak takový konfigurační soubor vypadá.



Obrázek 25 Příklad konfiguračního souboru aplikace

7.4.2 Spouštění nástrojů

Po načtení konfiguračního souboru, které je ošetřeno proti nežádoucím stavům, je na řadě samotné spouštění penetračních nástrojů. Pro spuštění, v tomto případě nástroje Nmap, je ve třídě „PenToolNmap“ obsažena funkce s názvem „run“. Tato funkce přebírá z konfiguračního souboru absolutní cestu k adresáři, ve kterém je k dispozici spustitelný soubor (například soubor s koncovkou „exe“) zvoleného nástroje a spouštěcí parametry pro daný penetrační nástroj. S těmito dvěma parametry je zavolána funkce „start“ ze třídy „QProcess“,

která na pozadí spustí vybraný nástroj a čeká na dokončení jeho běhu. Zmíněná funkce „start“ je programově uzpůsobena pro spouštění nástroje jak na operačním systému Windows, tak i na operačním systému Linux. O běhu nástroje je uživatel průběžně informován pomocí stavových informací na konzolovém rozhraní. Po doběhnutí penetračního nástroje je jeho výsledek uložen a následně dále zpracován.

7.5 Zpracování výstupů penetračních nástrojů

Druhou částí aplikace je zpracování již vygenerovaných výstupů z penetračních testů. Tyto výstupy jsou buď ve formě XML souboru (platí pro nástroj Nmap a OWASP ZAP) nebo ve formě SQLite databáze uložené v souboru s koncovkou „.sprt“ (platí pro nástroj Sparta). Je-li vybrána volba zpracování (přepínač „-p“ nebo „—parse“), je provedeno načtení požadovaného souboru a následně provedeno jeho zpracování odpovídající třídou podle vybraného nástroje.

7.5.1 Výstupy z nástrojů Nmap a OWASP ZAP

Po úspěšném načtení souboru se začne odpovídající třídou („PenToolNmap“ nebo „PenToolZap“) obsah souboru zpracovávat. Pro zpracování slouží funkce nazývající se „parse“, která si načte obsah souboru do třídy „QDomElement“, pomocí které se již přistupuje na jednotlivé položky vstupního souboru. Tímto způsobem jsou zpracovány veškeré požadované položky ze souboru. Dále jsou získaná data postupně uložena do objektu třídy, která zase odpovídá použitému nástroji, a to „ResultNmap“ nebo „ResultZap“.

7.5.2 Výstupy z nástroje Sparta

Pro případ penetračního nástroje Sparta je situace mírně odlišná. Odlišností je způsob, jakým Sparta výsledky uchovává. Výsledky jsou zde uloženy v jednom souboru, který obsahuje SQLite databázi s tabulkami obsahující výsledky jednotlivých modulů. Zpracování zde obstarává třída s názvem „PenToolSparta“ která využívá pro práci s databází třídy „QSqlDatabase“ a „QSqlQuery“. Jelikož se jedná o databázový soubor, jsou požadované položky získávány pomocí databázových dotazů. V případě modulu Nikto jsou navíc získané položky nadále zpracovávány regulárními výrazy pomocí tříd „QRegularExpression“ a „QRegularExpressionMatch“. Po získání jsou data, tak jako v minulém případě, uložena do objektu odpovídající třídy, v tomto případě „ResultSparta“.

7.6 Uložení zpracovaných dat

Po zpracování výstupů z penetračních testů jsou získaná data uložena do paměti aplikace a následně se provede i uložení do požadovaného souboru. Uživatel aplikace má na výběr uložení do dvou formátů, a to do formátu XML nebo JSON. Po uložení dat do výstupního souboru je běh aplikace validně ukončen.

7.6.1 Formát XML

Uložení do formátu XML je zprostředkováno pomocí funkce s názvem „toXmlFile“, která je zastoupena ve všech třídách obsahujících uložená data z výstupů nástrojů pro penetrační testování. Jmenovitě se jedná o třídy „ResultNmap“, „ResultZap“ a „ResultSparta“. Ještě před samotným zápisem dat do formátu je vytvořen výstupní soubor s názvem, který obsahuje název nástroje (Nmap, Zap nebo Sparta), dále následuje časové razítko ve tvaru rok, měsíc, den, hodina, minuta, sekunda a celý název je ukončen koncovkou „.xml“. Samotné ukládání je prováděno třídou s názvem „QXmlStreamWriter“, která disponuje funkcemi pro ukládání jednotlivých záznamů. Takto jsou všechna data uložena do souboru a uživatel je informován o uložení hláškou na konzoli, kde je vypsán i název vytvořeného souboru. Ukázka výstupního souboru z nástroje Nmap ve formátu XML je obsažena v příloze I.

7.6.2 Formát JSON

V případě ukládání do formátu JSON je situace podobná. Tentokrát je však použito vytvořené funkce s názvem „toJsonFile“. Stejně jako u XML tak i tady je předem vytvořen soubor s názvem nástroje, časovým razítkem a koncovkou souboru, v tomto případě „.json“. Samotné ukládání je u tohoto formátu mírně odlišné. Nejdřív je vytvořen objekt třídy „QJsonObject“, do kterého jsou postupně vkládány data z penetračního nástroje. Následně je tento objekt převeden do třídy „QJsonDocument“ a poté uložen do souboru vytvořeného výše. Stejně jako u formátu XML i zde je uživatel informován o uložení souboru hláškou na konzoli včetně názvu vytvořeného souboru. Také ukázka souboru ve formátu JSON je součástí přílohy II.

ZÁVĚR

Cílem této diplomové práce bylo prostudování problematiky penetračního testování a s tím nepřímo souvisejících norem řady ISO/IEC 27000. Dále bylo potřeba prostudovat možnosti nástrojů používaných pro penetrační testování, možnosti jejich výstupů a vytvořit vhodně vybranou technologií desktopovou aplikaci pro jejich zpracování.

V teoretické části této práce byly sepsány postupy a informace týkající se problematiky penetračního testování a informace ohledně skupiny norem ISO/IEC 27000. Dále byly popsány hojně používané penetrační nástroje Nmap, Sparta, Sqlmap a OWASP ZAP včetně jejich shrnutí. U těchto nástrojů jsou uvedeny informace ohledně podpory na operačních systémech a také možnosti a formáty jejich výstupů.

Praktická část se zabývá výběrem vhodné technologie pro vývoj aplikace umožňující automatizované spouštění penetračních nástrojů a zpracování jejich výstupů. Následně jsou vybrány výstupní formáty souborů pro zpracovaná data s přihlédnutím na možné další strojové zpracování, například pro generování reportů penetračních testů. Dále je navrhnuta struktura aplikace a je popsána její implementace včetně popisu ovládání aplikace a ukázkových výstupů ve formátu XML a JSON.

Výsledná aplikace umí, na základě vytvořeného konfiguračního souboru, automatizovaně spouštět penetrační nástroj Nmap. Další funkcí aplikace je zpracování výstupních souborů z penetračních nástrojů Nmap, OWASP ZAP a Sparta do souborů XML nebo JSON. Tyto soubory obsahují vytříděné informace z nástrojů v přehledné struktuře a jsou připravené pro vložení do závěrečné zprávy z penetračního testování. Aplikace disponuje jednoduchým, a přitom komplexním konzolovým ovládáním. V případě nutnosti by bylo možné aplikaci ještě více zautomatizovat pomocí dávkových soborů nebo bash skriptů.

SEZNAM POUŽITÉ LITERATURY

[1] WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking*. San Francisco: No Starch Press, 2014. ISBN 978-1593275648.

[2] *7 Penetration Testing Phases to Achieve Amazing Results* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://cyberx.tech/penetration-testing-phases/

[3] *Řada norem ISO/IEC 27000* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://www.iso27000.cz/

[4] *ISO 27000* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://managementmania.com/cs/iso-27000

[5] *ISO/IEC 27001:2013* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://www.iso27000.cz/rac/homepage.nsf/CZ/27001

[6] Certifikace systémů managementu bezpečnosti informací dle ISO 27001. *TÜV SÜD Czech* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://www.tuv-sud.cz/cz-cz/cinnosti/audit-a-certifikace-systemu/certifikace-systemu-managementu-bezpecnosti-informaci-dle-iso-27001

[7] *ISO/IEC 27001 - Základní informace* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://www.info-iso.cz/iso-27001/iso-27001-zakladni-informace

[8] *Information Security Management System* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://www.aec.cz/cz/isms

[9] *ISMS Systém řízení bezpečnosti informací* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://www.qcom.cz/systemy-rizeni/isms/

[10] *Taking the First Step with the PDCA* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://www.bulsuk.com/2009/02/taking-first-step-with-pdca.html

[11] *ISO/IEC 27002:2013* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://www.iso27000.cz/rac/homepage.nsf/CZ/27002

[12] *ISO 27002 - nejlepší bezpečnostní praktiky* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://managementmania.com/cs/iso-27002-nejlepsi-bezpecnostni-praktiky

[13] *ČSN EN ISO/IEC 27002 (369798) Informační technologie - Bezpečnostní techniky - Soubor postupů pro opatření bezpečnosti informací* [online]. b.r. [cit. 2019-05-07]. Dostupné z: http://www.technicke-normy-csn.cz/369798-csn-en-iso-iec-27002_4_95679.html

[14] *Nmap* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://nmap.org/

[15] *Zenmap* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://nmap.org/zenmap/

[16] *SPARTA Network Infrastructure Penetration Testing Tool* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://sparta.secforce.com/index.html

[17] *Sqlmap* [online]. b.r. [cit. 2019-05-07]. Dostupné z: http://sqlmap.org/

[18] *OWASP Zed Attack Proxy Project* [online]. b.r. [cit. 2019-05-07]. Dostupné z: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[19] *Introduction to the C# Language and the .NET Framework* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[20] *Windows Forms* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/framework/winforms/

[21] *Hello World -- Your first program (C# Programming Guide)* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/programming-guide/inside-a-program/hello-world-your-first-program

[22] *Visual Studio* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://visualstudio.microsoft.com/cs/

[23] *Creating a basic Visual Basic Windows Forms application* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://help.qlik.com/en-US/sense-developer/November2018/Subsystems/NetSDKAPI/Content/Sense_NetSDKAPI/GettingStarted/Net-Sdk-VB-HelloWorld.htm

[24] *What is Java?* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://opensource.com/resources/java

[25] *Sun liberates JDK, delivers on open-source Java promise* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://arstechnica.com/uncategorized/2007/05/sun-liberates-jdk-delivers-on-open-source-java-promise/

[26] *JavaFX 12* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://openjfx.io/

[27] *Java/Hello World* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://cs.wikibooks.org/wiki/Java/Hello_World

[28] *Eclipse IDE* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.eclipse.org/eclipseide/

[29] *NetBeans IDE - The Smarter and Faster Way to Code* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://netbeans.org/features/index.html

[30] *IntelliJ IDEA* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.jetbrains.com/idea/

[31] *NetBeans IDE Features* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://netbeans.org/features/java/debugger.html

[32] *The Features of C++ as a Language* [online]. b.r. [cit. 2019-05-09]. Dostupné z: http://www.cplusplus.com/info/description/

[33] *History of C++* [online]. b.r. [cit. 2019-05-09]. Dostupné z: http://www.cplusplus.com/info/history/

[34] *C++ Memory Management: new and delete* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.programiz.com/cpp-programming/memory-management

[35] *What is CodeLite?* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://codelite.org/

[36] *The open source, cross platform, free C, C++ and Fortran IDE.* [online]. b.r. [cit. 2019-05-09]. Dostupné z: http://www.codeblocks.org/

[37] *CLion* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.jetbrains.com/clion/

[38] *Qt Creator IDE – Making software development fast, easy & fun* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.qt.io/qt-features-libraries-apis-tools-and-ide/#ide

[39] *About Qt* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://wiki.qt.io/About_Qt

[40] *Qt Automotive Suite* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.qt.io/qt-automotive-suite/

[41] *Qt Documentation* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://doc.qt.io/

[42] *Qt Creator* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://wiki.qt.io/Qt_Creator

[43] *Qt Designer Manual* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://doc.qt.io/qt-5/qtdesigner-manual.html

[44] *Introduction to XML* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.w3schools.com/xml/xml_whatis.asp

[45] *Co je XML?* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.interval.cz/clanky/co-je-xml/

[46] *Úvod do JSON* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.json.org/json-cz.html

[47] *JSON : jednotný formát pro výměnu dat* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat/

[48] *JSON: The Fat-Free Alternative to XML* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.json.org/xml.html

[49] *Example of a JSON File* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://www.kodingmadesimple.com/2014/12/how-to-insert-json-data-into-mysql-php.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
ISMS	Information Security Management System
PDCA	plan-do-check-act
OSVDB	Open Sourced Vulnerability Database
GUI	Graphical User Interface
JVM	Java Virtual Machine
MS	Microsoft
API	Application Programming Interface

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: UKÁZKA VÝSTUPNÍHO SOUBORU XML Z NÁSTROJE NMAP

```
<?xml version="1.0" encoding="UTF-8"?>
<nmap startTime="1555397000" stopTime="1555397196" duration="196" version="7.70" args="nmap -Ap1-65535 -
oX testNmap.xml 192.168.250.248" message="">
    <hosts>
        <host ipAddr="192.168.250.248" macAddr="00:0C:29:F1:63:64" vendor="VMware" uptime="2528520"
networkDistance="1">
            <ports>
                <port id="22" protocol="tcp" state="open" service="OpenSSH" serviceName="ssh"
serviceVersion="7.6p1 Ubuntu 4ubuntu0.3"/>
                <port id="25" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
                <port id="443" protocol="tcp" state="open" service="Apache/2.4.29 (Ubuntu)" serviceName="
https" serviceVersion=""/>
                <port id="8008" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
                <port id="8111" protocol="tcp" state="open" service="" serviceName="unknown"
serviceVersion=""/>
                <port id="8112" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
                <port id="8115" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
                <port id="10000" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
                <port id="20000" protocol="tcp" state="open" service="Apache httpd" serviceName="http"
serviceVersion="2.4.29"/>
            </ports>
            <os>
                <osMatch name="Linux 3.2 - 4.9" accuracy="100"/>
            </os>
            <trace>
                <hop ipAddr="192.168.250.248" rtt="0.25"/>
            </trace>
        </host>
    </hosts>
</nmap>
```

PŘÍLOHA P II: UKÁZKA VÝSTUPNÍHO SOUBORU JSON Z NÁSTROJE NMAP


```
{
    "args": "nmap -Ap1-65535 -oX testNmap.xml 192.168.250.248",
    "duration": 196,
    "hosts": [
        {
            "ipAddr": "192.168.250.248",
            "macAddr": "00:0C:29:F1:63:64",
            "networkDistance": "1",
            "os": [
                {
                    "accuracy": "100",
                    "name": "Linux 3.2 - 4.9"
                }
            ],
            "ports": [
                {
                    "id": "22",
                    "protocol": "tcp",
                    "service": "OpenSSH",
                    "seviceName": "ssh",
                    "seviceVersion": "7.6p1 Ubuntu 4ubuntu0.3",
                    "state": "open"
                },
                {
                    "id": "8112",
                    "protocol": "tcp",
                    "service": "Apache httpd",
                    "seviceName": "http",
                    "seviceVersion": "2.4.29",
                    "state": "open"
                },
            ],
            "trace": [
                {
                    "ipAddr": "192.168.250.248",
                    "rtt": "0.25"
                }
            ],
            "uptime": "2528520",
            "vendor": "VMware"
        }
    ],
    "message": "",
    "startTime": 1555397000,
    "stopTime": 1555397196,
    "version": "7.70"
}
```