Generátor reportů z penetračních testů

Bc. Richard Goldmann

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. Richard Goldmann**
Osobní číslo:	**A17225**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**prezenční**
Téma práce:	**Generátor reportů z penetračních testů**
Téma anglicky:	**A Penetration Test Report Generator**

Zásady pro vypracování:

1. **Seznamte se s problematikou penetračního testování.**
2. **Prostudujte požadavky na obsah a formát reportů dle normy ISO 27000.**
3. **Vhodným způsobem definujte požadavky na aplikaci.**
4. **Uveďte přehled možných technologií pro sestavení aplikace.**
5. **Navrhněte strukturu aplikace pro generování reportů.**
6. **Implementujte aplikaci za použití zvolené technologie.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **WEIDMAN, Georgia. Penetration testing: a hands-on introduction to hacking. San Francisco: No Starch Press, [2014]. ISBN 978-1-59327-564-8.**
2. **SELECKÝ, Matúš. Penetrační testy a exploitace [online]. Brno: Computer Press, 2012 [cit. 2018-11-29]. ISBN 978-80-251-3752-9.**
3. **GUILLAUME, Lazar a Robin PENEA. Mastering Qt 5. Packt Publishing Limited, 2016. ISBN 9781786467126.**
4. **ZHI ENG, Lee. Qt5 C++ GUI Programming Cookbook. Birmingham, United Kingdom: Packt Publishing Limited, 2016. ISBN 9781783280278.**
5. **ENGEBRETSON, Patrick. The Basics of Hacking and Penetration Testing. Syngress, 2011. ISBN 9781597496551.**
6. **Qt Documentation [online]. Dostupné také z: https://doc.qt.io/**

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 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 15.05.2019

Bc. Goldmann Richard, v.r
podpis diplomanta

ABSTRAKT

Hlavním cílem této diplomové práce je navrhnout aplikaci pro generovaní reportů z penetračních testů, které jsou prováděny v laboratoři PT LAB na Univerzitě Tomáše Bati ve Zlíně. Teoretická část se zabývá problematikou penetračního testování, souvisejícími normami a definicí požadavků na výslednou aplikaci.

V praktické části je vybrána vhodná technologie a navrhnuta struktura aplikace. S ohledem na definované požadavky, je dále sestavena aplikace pro generování reportů na základě dat zadaných uživatelem nebo načtených ze souboru. Popis sestavené aplikace včetně základních principů jejího používání, je uveden v závěru této práce.

Klíčová slova: Penetrační testování, OWASP, ISO 27000, Framework Qt

ABSTRACT

The main aim of this thesis is to design an application for penetration test report generating. These tests are performed at PT LAB laboratory at Tomas Bata University in Zlin. The theoretical part of this thesis deals with the principles of penetration testing, related standards and requirements for the target application.

The suitable technology was selected in the practical part as well as application structure was defined. Based on the predefined requirements, the application for penetration testing report generating was developed. The report generation could be based on user inputs or data imported from a file. Final application description, including basic use cases, is provided at the end of this thesis.

Keywords: Penetration tests, OWASP, ISO 27000, Framework Qt

Děkuji vedoucímu mé diplomové práce panu Ing. Peterovi Janků za jeho ochotu, cenné rady a vyvinutou snahu vedoucí k dokončení mé diplomové práce. Jsem vděčný panu Ing. Davidu Malaníkovi, Ph.D. za konzultace ohledně vyvíjené aplikace. Nakonec děkuji své rodině, za dlouholetou podporu ke studiu, kterou jsem od nich obdržel.

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

OBSAH

ÚVOD

V současné době je zabezpečení počítačových sítí velmi důležitou součástí každé společnosti, která ke své činnosti využívá výpočetní techniku. To platí ještě více pro společnosti, které zpracovávají osobní data uživatelů či klientů, jako jsou například sociální sítě. Ty se přímo živí zpracováváním osobních údajů jejich uživatelů a spolu s nimi nesou i zodpovědnost za jejich bezpečné uchování.

Kdyby se potenciálnímu hackerovi podařilo získat plný přístup k datům nějaké společnosti, mohl by je zpeněžit, například prodejem konkurenci. Tím by společnosti mohl způsobit velké finanční ztráty, zhoršit její postavení na trhu nebo zapříčinit úplný bankrot. Proto je v nejlepším zájmu, nejen společností provozujících sociální sítě, aby měly co možná nejlepší zabezpečení svých systémů.

Pro zvýšení kvality zabezpečení je nejprve nutné vědět, kde jsou nedostatky. Právě k tomu lze využít penetrační testování. Penetrační testování je postup, při němž mohou být nalezeny nedostatky v zabezpečení softwarů nebo sítí.

Výsledkem penetračního testu je hlášení, jehož důležitost je stejná jako kvalita samotného penetračního testu či zkušenosti osoby, která test provádí. Hlášení je důležitou zpětnou vazbou zadavateli. Právě z něj zadavatel zjistí jaké má jeho systém nedostatky a jak je napravit. Pokud by samotný test měl vysokou kvalitu, ale nebylo z něj podáno žádné hlášení nebo zpětná vazba zadavateli, byl by celý test zbytečný. A právě proto je důležité, aby hlášení bylo srozumitelné a obsahovalo vše, co je pro zadavatele podstatné.

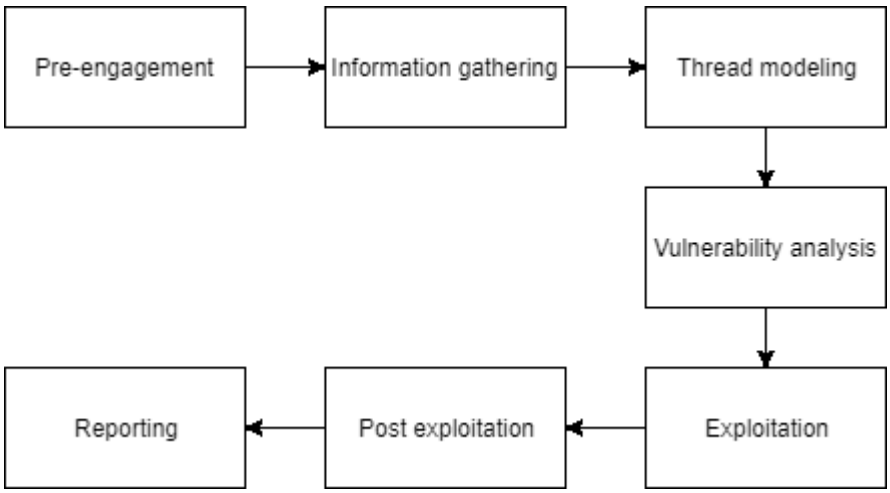
Penetrační testy provádí každá společnost případně jedinec dle vlastních nejlepších zkušeností a zvyků. Kvůli tomu jsou také hlášení z reportů individuální. Reporty, které vytvořily různé osoby či společnosti, se liší po stránce vzhledové i obsahové. Má aplikace by měla být využívána k jednoduššímu generování takovýchto reportů dle zvyků a metodik, které jsou využívány v laboratoři penetračního testování PT LAB na Univerzitě Tomáše Bati ve Zlíně.

I. TEORETICKÁ ČÁST

1 PENETRAČNÍ TESTY

Penetrační testování, někdy též nazýváno jako „etické hackování“ nebo „pentest“, je praktika, při níž testující osoba provede tzv. hodnocení zranitelností (Vulnerability assessment), při kterém se snaží najít slabiny daného počítačového systému, kterých by mohl zneužít případný útočník.

Penetrační test začíná přípravnou částí pre-engagement, během níž probíhá komunikace s klientem a domlouvají se podrobnosti plánovaného útoku. Po ní následuje informartion gathering, tedy sběr informací o smluveném cíli. Jakmile má osoba provádějící penetrační test (dále jen pentester) dostatek informací, přejde do části thread modeling, kdy vytváří plány, jak bude postupovat během útoku. Následně na základě vytvořeného plánu útoku bude pentester hledat zranitelnosti v části vulnerability analysis. Po nalezení zranitelností se pentester pokusí zranitelnosti zneužít a získat tak oprávnění v systémech, ke kterým by normálně neměl mít přístup. V případě, že se pentesterovi podaří získat přístup do některých systémů, přejde do části post exploitace. V ní zjišťuje, jak velké škody je možné napáchat a do kterých dalších systému by se mu mohlo podařit získat přístup. Nakonec pentester podává zadavateli detailní hlášení o průběhu útoku a jeho nálezech. Na základě hlášení pak zadavatel může jednat a bezpečností slabiny opravit. Průběh penetračního testu je znázorněn na Obr. 1. [1] [2]



Obr. 1 Diagram průběhu penetračního testu

1.1 Pre-engagement

Pentester se nejdříve sejde s klientem a spolu vyjednávají okolnosti provedení testu, které jsou detailně popsány níže. Chyba v této části by mohla vést např. k nepříjemné situaci, kdy klient očekává jen jednoduchý sken sítě, ale penetrační testy jsou mnohem více invazivní, než by mohl klient předpokládat. Během penetračních testů může dojít k výpadku služby, případně i k poškození zařízení. [1]

Pentester musí pochopit s jakým záměrem má test provádět, jaké jsou klientovy obavy nebo má-li v síti nějaká citlivá zařízení, na která bude v průběhu penetračního testu potřeba dát si pozor. Je dobré zjistit, pokud jde o klientův první penetrační test, co jej k němu přimělo. Je důležité pochopit na čem je založené podnikání klienta a co jej nejvíce ohrožuje. Např. hodinový výpadek webu velkého e-shopu by mohl pro firmu znamenat velkou finanční ztrátu, ale hodinový výpadek webu banky by sice naštval několik zákazníků, ale pravděpodobně nezavinil velké finanční ztráty. [1]

S klientem by měly být dohodnuty následující záležitosti, které jsou detailně popsány níže:

1.1.1 Rozsah testovaných IP adres

Je nutné zjistit, které IP adresy mají být testovány a které už ne a také jak moc nebezpečné a invazivní testy může na některých zařízeních provádět. Například může-li pentester zkoušet útoky, které by mohly potenciálně vyřadit nějaké zařízení nebo se má limitovat pouze na zjišťování potenciálních rizik a dále je netestovat. Je dobré klienta vyrozumět, že i jednoduché skenování portů routeru může potenciálně dočasně nebo i trvale router vyřadit. Dále pak je nutné se domluvit, jestli má pentester (pokud takovou službu nabízí) provádět i útoky sociálního inženýrství na zaměstnance. [1]

1.1.2 Časové okno testování

Je potřeba domluvit se s klientem, kdy mají být penetrační testy prováděny. V případě, že by mělo dojít k selhání nějakého zařízení následkem penetračního testování, tak pro klienta bude vhodné, aby se tak stalo v době, kdy testované zařízení není využíváno, případně by jeho vyřazení z provozu mělo co nejmenší dopad na chod společnosti. [1]

1.1.3 Kontaktní informace

V případě, že by pentester našel velmi závažné bezpečnostní riziko, je potřeba se s klientem okamžitě spojit a proto je nutné mít jeho kontaktní informace. Dále se musí domluvit na způsobu komunikace, jestli stačí telefonní hovor, email nebo je zapotřebí využívat nějakou formu šifrované komunikace. [1]

1.1.4 Oprávnění

Pentester si musí být jist, že má oprávnění provádět penetrační testy na dané zařízení. Testované cíle nemusí nutně patřit klientově organizaci, může jít o web který, je provozován na hardwaru třetí strany a pentester si musí být jistý, že klient je domluvený s třetí stranou a má souhlas s penetrační testováním. Pentester by měl mít ve smlouvě prohlášení, které omezí jeho zodpovědnost, pokud by se stalo něco nečekaného během provádění testů. [1]

1.1.5 Forma platby

S klientem je také nutné vyjednat formu platby a jestli bude platba provedena předem nebo až po dokončení testů. [1]

1.2 Information gathering

Následuje část, kdy osoba provádějící penetrační test potřebuje získat co možná nejvíce informací o cíli. Může k tomu využít různé informace dostupné volně na Internetu nebo po domluvě s objednatelem získat informace už před testem. Dále může využít nástroje pro skenovaní portů. Díky tomu začne pentester získávat představu o vnější struktuře cílené sítě, jaké služby na zařízeních jsou aktivní a tedy i potenciální slabá místa, kterých se dá následně využít. Dle množství informací, které zadavatel poskytne, se pak testování dělí do následujících tří kategorií. [1] [2]

1.2.1 Black-box testing

Při black-box testu objednatel neposkytne žádné informace o cíli, tím dochází k simulaci útoku hackerem bez znalostí cíleného systému nebo sítě. Výhodou je výsledek podobnější testu skutečné situaci, nevýhodou je pak skutečnost, že lze opomenout vážné bezpečností nedostatky, které by skutečný hacker přehlédnout nemusel. Současně je také test náročnější a vyžaduje vyšší kvalifikaci testera. [2] [3]

1.2.2 White-box testing

Při white-box testu objednatel poskytne rozsáhlé informace o cíli a tím zvýší pravděpodobnost nalezení slabin a zkrátit dobu potřebnou pro testování. [2] [3]

1.2.3 Grey-box testing

Dalším možný přístupem je tzv. grey-box testing. Ten se snaží kombinovat výhody obou předešlých možností poskytnutím částečných informací. [2]

1.3 Threat modeling

Na základě informací, které byly získány v předešlé části, si pentester vytvoří plány jak a za jakým účelem útočit. Například je-li cílem firma, která vyvíjí software, tak by případný hacker mohl zničit celou firmu, pokud by se mu podařilo dostal do interní sítě a získat zdrojové kódy softwaru, který je zde vyvíjen a prodat je konkurenční firmě. [1] [3]

1.4 Vulnerability analysis

Nyní se už pentester zaměří na hledání konkrétních zranitelností, pomocí kterých by se mu mohlo podařit získat oprávnění do některých systémů. Na základě informací, které získal, musí zvolit co nejvhodnější nástroje, díky kterým se mu podaří obejít přítomné zabezpečení.

K analýze zranitelností lze využívat automatizované softwary, které jsou sice velmi šikovné, ale nedokážou nahradit kritické myšlení zkušeného pentestera. Proto se výsledky automatizovaných softwarů procházejí manuálně a případně i validují manuálními testy. [1] [3]

1.5 Exploitation

V této části se začne útočit na nalezené zranitelnosti a pentester se pokusí získat přístup do cílených zařízení. Využít lze například chyby ve starých službách, které správce sítě neaktualizoval nebo se přihlásit do robustního programu pomocí výchozích nebo lehce odhadnutelných přihlašovacích údajů. [1] [3]

1.6 Post exploitation

Post exploitation je možná nejdůležitější část celého penetračního testu. Pokud se pentesterovi podařilo získat přístup do některých systému cílené sítě, musí zjistit jaké riziko přítomné

slabiny představují. Musí zjistit jak velkou škodu je teď útočník schopen napáchat. Například pokud získal přístup do počítače, který je v izolované části sítě a jediné co může dělat je využívat připojenou tiskárnu, tak pravděpodobně nebude schopen napáchat velké škody jako když stáhne a smaže databázi klientů spolu s jejich zálohami. Pentester tedy hledá na systémech data, jejichž únik nebo smazání by představovalo pro cílenou firmu riziko. Může hledat soubory, ve kterých jsou uložena hesla, databáze nebo jiné citlivé informace. [1]

1.7 Hlášení

Poslední částí penetračního testu je podání hlášení zadavateli. Hlášení je výstup z penetračního testu. Musí obsahovat detailní informace o provedeném testu, informace o špatných zvycích, které byl pentester schopen identifikovat (např. si někdo ukládá hesla do souboru v čitelné formě, opakovaně využívá stejná hesla atd.) Hlášení musí být napsáno tak, aby z něj pochopil informace jak administrátor sítě, tak manažer, který bezpečnosti nerozumí. Např. „využil jsem exploit MS08-067 a získal shell“ neřekne manažerovi mnoho. Pokud bude však ve zprávě napsáno „byl jsem schopen číst vaše emaily“, tak i manažer nejspíš pochopí, že jde o zásadní bezpečnostní nedostatek a je třeba urychleně jednat. [1] [3]

Hlášení se skládá ze dvou částí. Manažerského shrnutí, které je určeno spíše pro manažerské pozice, tedy neobsahuje detailní popis průběhu samotného testu, ale jen základní náhled na test a jeho výsledky. Druhou částí je technické hlášení, to je určeno pro osoby zodpovědné za bezpečnost v podniku, protože obsahuje detailní informace o tom, jak byl test prováděn a o jeho nálezech. [1]

1.7.1 Manažerské shrnutí

V manažerském shrnutí jsou sepsány cíle testu a popsány nalezené skutečnosti.

Mělo by obsahovat: [1] [3]

Background – Popis cílů testu a definice pojmů využitých v hlášení, které by cíleným osobám mohly být nejasné.

Přehled – Přehled účinnosti testu a nalezené problémy. Obsahuje obecné informace o nalezených bezpečnostních nedostatcích např. nedostatečně časté aktualizace zařízení.

Rizikový profil – Celkové hodnocení zabezpečení systémů ve firmě v porovnání s jinými firmami. Současně by také mělo být uvedeno, dle čeho porovnávání probíhalo.

Nálezy – Obecný přehled nalezených problémů s bezpečností.

Doporučení – Přehled úkonů, které by firma měla provést k opravení nalezených nedostatků.

Rozcestník – Informace pro klienta, jaké by měly být jeho krátkodobé a dlouhodobé cíle z hlediska bezpečnosti. Např. doporučit okamžité doinstalování bezpečnostních záplat jako krátkodobé řešení, ale s tím, že je potřeba vytvořit dlouhodobý plán, jak spravovat aktualizace, aby se problém už neopakoval.

1.7.2 Technické hlášení

Technické hlášení obsahuje detailní informace o průběhu testu samotného, mělo by obsahovat:

Úvod – Přehled rozsahu testu.

Nalezené informace – Detaily nalezených informací z části „1.2 Information gathering“, tedy co se testerovi podařilo zjistit z veřejně dostupných zdrojů o cílené společnosti nebo systému.

Analýza zranitelností – Informace získané v části „1.4 Vulnerability Analysis“ o nalezených potenciálních zranitelnostech. Výstupy z využitých softwarů jako je například NMap.

Exploitace a ověření zranitelností – Zhodnocení výsledků exploitů. Jaké systémy se testerovi podařilo prolomit a pomocí kterých exploitů toho docílil.

Post exploitation – Zjištěné skutečnosti z části „1.6 Post exploitation“.

Vystavení riziku – Kvantitativní popis nalezených rizik. Je zde odhad ztrát a jaké představují jednotlivé nalezené bezpečností nedostatky riziko v případě, že by byly zneužity hackerem.

Závěr – Shrnutí celého testu. [1] [3]

2 OWASP

Ve spojitosti s penetračním testováním je úzce spojena zkratka OWASP, znamenající The Open Web Application Security Project. Jde o projekt, jehož cílem je zvýšit bezpečnost webových aplikací. Pod názvem OWASP se skrývá několik různých dílčích projektů se stejným cílem, kterého se snaží dosáhnout různými způsoby. [4] [5]

Obr. 2 Logo OWASP [6]

Projekty spadající pod OWASP lze rozdělit do dvou hlavních kategorií. Projekty dokumentační, které se snaží co nejlépe popsat a zdokumentovat záležitosti týkající se zabezpečení webových aplikací. A druhou kategorií jsou projekty nástrojové. Ty mají vývojářům nebo bezpečnostním specialistům zjednodušit práci různými nástroji. Veškeré nástroje a dokumentace, které spadají pod OWASP, jsou dostupné zdarma na oficiálních stránkách www.owasp.org. [7] [5]

2.1 Dokumentační projekty

Dokumentační projekty se snaží vývojářům a bezpečnostním specialistům pomoci různými texty, radami, směrnicemi a ukázkami. Patří mezi ně například OWASP Top Ten a OWASP Application Security Verification Standard, popsané v následujících kapitolách. [5]

2.1.1 OWASP Top Ten

OWASP Top Ten je dokument obsahující deset nejkritičtějších bezpečnostních nedostatků webových aplikací. Lidé, kteří se podíleli na tvorbě toho dokumentu jsou bezpečnostní specialisté z celého světa. Předposlední verze byla vydána v roce 2013 a poslední verze je z roku 2017. Obr. 3 znázorňuje přechod mezi verzemi a jejich obsah. [8] [9]

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Obr. 3 Přechod z OWASP Top Ten 2013 na rok 2017 [9]

2.1.2 OWASP Application Security Verification Standard (ASVS)

ASVS je projekt, který poskytuje seznam požadavků, které pomůžou s vývojem, údržbou, zabezpečením a testováním zabezpečení webových aplikací. Požadavky ASVS jsou kategorizovány do tří úrovní podle typu dat, jaké aplikace zpracovává. Čím citlivější data aplikace zpracovává, tím více bezpečnostních požadavků je na aplikaci kladeno. [10]

2.2 Nástrojové projekty

Nástrojové projekty poskytují nástroje, které mohou pomoci vývojářům či bezpečnostním specialistům s vývojem nebo analýzou zabezpečení webových aplikací. Nástroje mohou poskytovat například testování bezpečnosti aplikací nebo trénování bezpečnostních specialistů. Mezi nástrojové projekty patří například OWASP Zed Attack Proxy. [7]

2.2.1 OWASP Zed Attack Proxy (ZAP)

Zed Attack Proxy je nástroj umožňující automatizované i manuální skenování zranitelností webových aplikací. Díky tomu jej mohou využívat jak začátečníci, tak profesionální penetrační testeři. [11]

3 NÁSTROJE VYUŽÍVANÉ U PENETRAČNÍHO TESTOVÁNÍ

K provádění penetračních testů je nutné mít vhodné softwarové nástroje. Některé mohou být automatické, vyžadující jen minimální znalosti uživatele, jiné jsou užitečné jen v rukou zkušených profesionálů a některé jsou kombinací obou předešlých variant. Velké množství takových nástrojů je zdarma dostupné ke stažení na Internetu nebo přímo součástí některých distribucí operačních systémů. Mezi populární distribuce operačních systémů, které už mají v sobě předem nainstalované a připravené velké množství takových nástrojů, patří například Kali Linux (dříve známý jako BackTrack) nebo Parrot OS. Níže následuje stručný přehled některých programů využívaných k penetračním testům. [12] [13]

3.1 Nmap

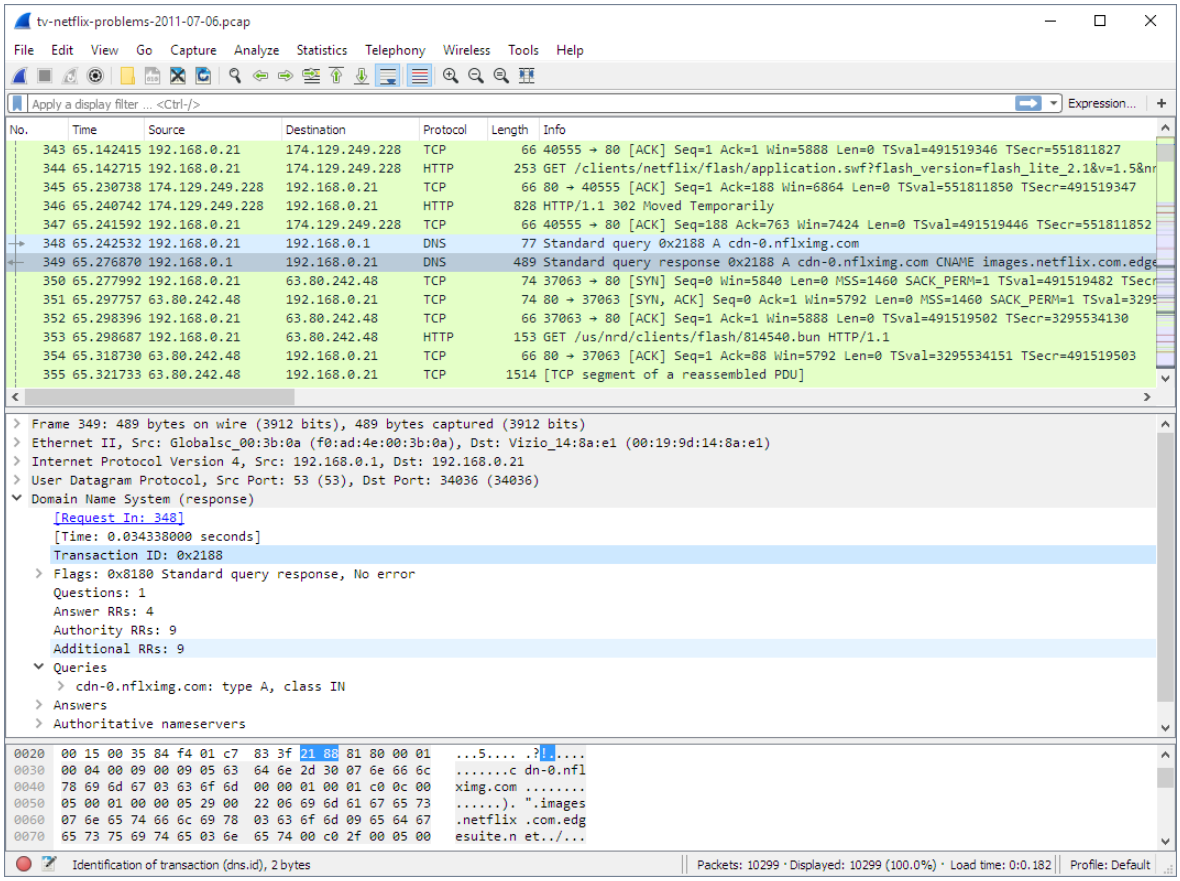
Npam je konzolový sofware distribuovaný jako open source pod licencí GPL. Využíván je především pro skenování sítí. Nástroj je schopen skenovat cílový rozsah IP adres a tím detekovat zařízení, která jsou v daném IP rozsahu aktivní a zjistit o aktivních zařízeních velké množství informací, mezi které patří například jaké porty mají zařízené otevřené, jaké služby a verze služeb jsou na daných portech a je také schopen detekovat operačním systém samotného zařízení. Zenmap je verze Nmapu, která obsahuje uživatelské rozhrání, přes které lze přímo spouštět skeny, nebo prohlížet výsledky předchozích skenů. Ukázkový výstup z Nmapu pro skenovaní portů zařízení je vidět na Obr. 4 [14]

```
[root@darkstar ~]#
[root@darkstar ~]# nmap -PN sS -O Scanme.Nmap.Org

Starting Nmap 5.21 ( http://nmap.org ) at 2010-04-01 11:19 IDT
Nmap scan report for Scanme.Nmap.Org (64.13.134.52)
Host is up (0.18s latency).
rDNS record for 64.13.134.52: scanme.nmap.org
Not shown: 993 filtered ports
PORT      STATE  SERVICE
25/tcp    closed smtp
53/tcp    open   domain
70/tcp    closed gopher
80/tcp    open   http
113/tcp   closed auth
8009/tcp  open   ajp13
31337/tcp closed Elite
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.15 - 2.6.26

OS detection performed. Please report any incorrect results at http://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 16.99 seconds
[root@darkstar ~]#
```

Obr. 4 Konzolový výstup z Nmapu [15]

3.2 Wireshark

K analýze síťového provozu lze použít například Wireshark. Jde o „protokolový analyzér“, který uživateli podá velmi detailní informace o síťovém provozu. Zná široké spektrum protokolů, je multiplatformní, poskytuje komplexní uživatelské rozhraní a výsledky analýzy je schopen exportovat v mnoha formátech. Může být využit například administrátorem sítě k řešení problémů spojených se sítí, bezpečnostním specialistou k prozkoumání podezřelého provozu na síti nebo jen zvídavými lidmi, kteří se chtějí dozvědět více o síťovém protokolu. Náhled na uživatelské rozhraní je na Obr. 5 [16]



Obr. 5 Uživatelské rozhraní programu Wireshark [16]

3.3 Sqlmap

Sqlmap je open source konzolový nástroj určený k penetračnímu testování webových stránek s SQL databází. Jde o automatizovaný software, který je schopen sám detekovat slabinu (například neošetřený uživatelský vstup) a pak ji také „zneužít“. Ukázka průběhu útoku je na Obr. 6. Sqlmap schopen prohlížet a stahovat obsahy databází a v případě velmi slabého

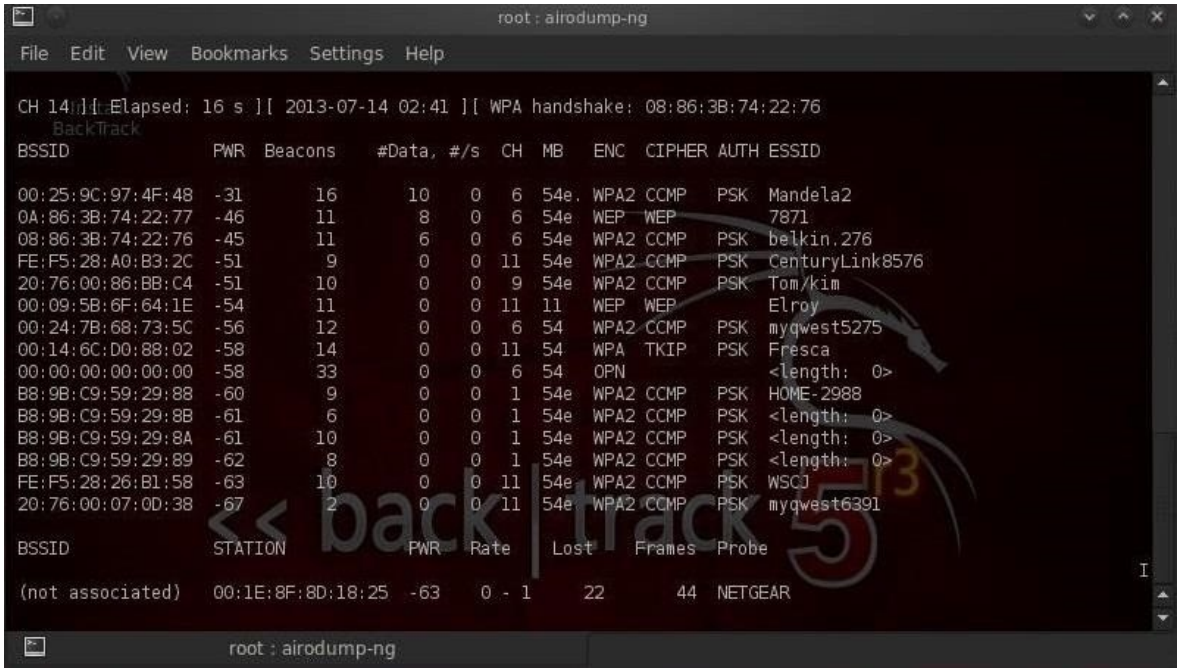
zabezpečení je schopen převzít úplnou kontrolu nad celým systémem, na kterém je databáze provozována. [17]

Obr. 6 Průběh útoku sqlmap [17]

3.4 Aircrack-ng

Jedním ze softwarů vhodných pro penetrační testování zabezpečení Wi-Fi sítí je například balíček nástrojů Aircrack-ng. Náhled na airodump-ng, jeden z nástrojů z balíčku aircrack-ng, je na Obr. 7. Mezi hlavní funkce Aircrack-ng patří:

- Monitorování – zachytávání síťových paketů a jejich export pro další zpracování v jiných nástrojích.
- Útočení – několik forem útoků vůči Wifi přístupovým bodům nebo jejich klientům.
- Testování – kontrola, jestli je dostupná síťová karta vhodná pro zachytávání a podvrhování paketů.
- Cracking – útoky na WEP a WPA PSK. [18]



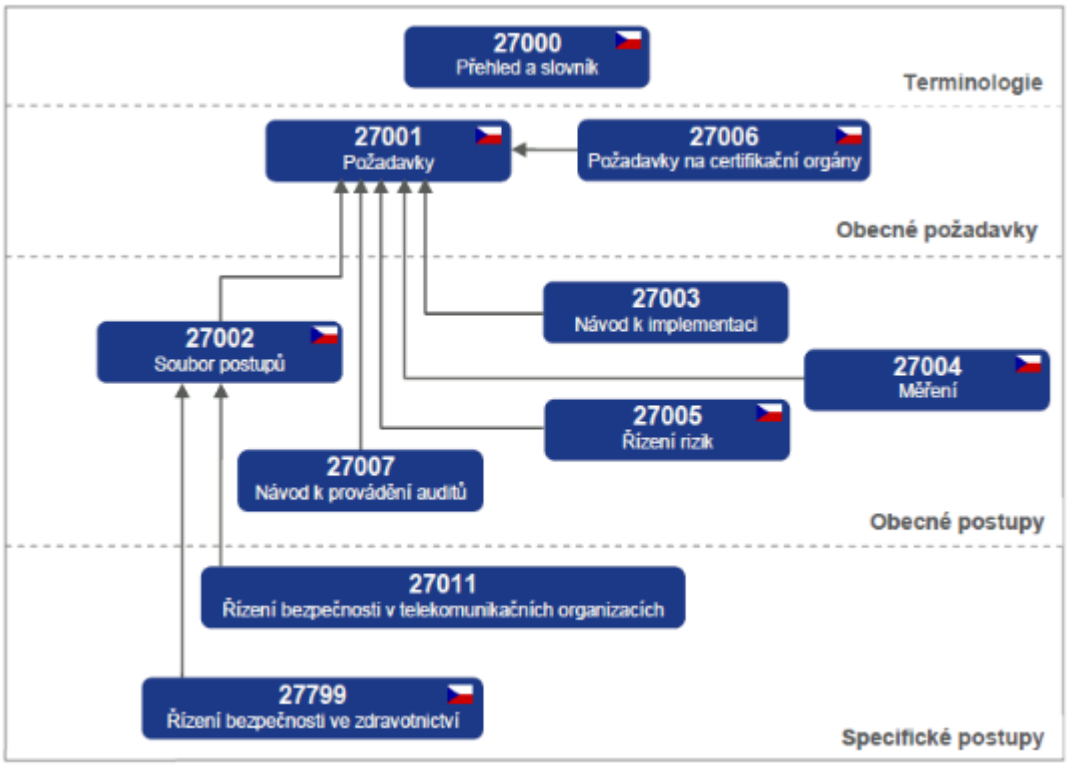
Obr. 7 Konzolový výstup z airodump-ng [19]

4 ŘADA NOREM ISO 27000

Řada norem 27000 (také nazýváno „ISO27K“ nebo „ISO 27000 Family of Standards“) je vyvíjena a publikována organizací International Organization for Standardization (ISO) a International Electrotechnical Commission (IEC). [20]

Normy pomáhají společnostem udržovat jejich aktiva v bezpečí pomocí kladení požadavků na bezpečnost řízení informačních systémů. Mají široký rozsah a využívají se v organizacích všech velikostí a ve všech sektorech. S vývojem nových technologií se neustále vyvíjejí i standardy pro řešení měnících se požadavků na bezpečnost informací v různých průmyslových odvětvích a prostředích. [20] [21]

Norma ISO 27001 je nejznámějším standardem z rodiny norem ISO 27000 a poskytuje požadavky na systém řízení bezpečnosti informací (ISMS). Další důležitou normou je 27002, která obsahuje seznam opatření a odborných praktik. Obě normy spolu úzce souvisí. Norma ISO 27001 specifikuje požadavky na zavedení ISMS, kdežto norma ISO 27002 poskytuje rejstřík bezpečnostních opatření, které mohou být během budování ISMS využity. Stručný popis některých norem z řady norem ISO 27000 je na Obr. 8. [21] [22]



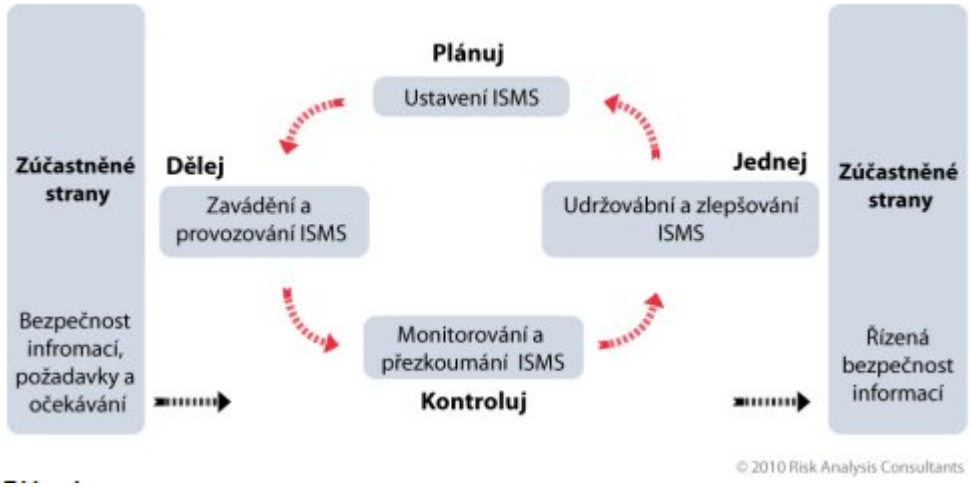
Obr. 8 Struktura norem řady ISO 27000 [23]

4.1 ISO 27001

Norma ISO 27001 se skládá z řad doporučení a „best practices“ jak provozovat management bezpečnosti. Pomáhá společnostem udržovat jejich informační aktiva v bezpečí díky její specifikaci požadavků ISMS. Samotná norma 27001 neobsahuje konkrétní nařízení, jak má konkrétní systém být zabezpečen, jen jak má být dané zabezpečení systému zavedeno. [24]

4.1.1 ISMS

Systém řízení bezpečnosti informací je systém, který má zvýšit bezpečnost informačního systému podniku či státního orgánu. Požadavky na něj jsou specifikovány normou 27001. Rozsah a velikost opatření závisí na velikosti organizace a důležitosti dat, jež organizace zpracovává. Samotný systém ISMS je veden ve čtyřech krocích označovaných jako PDCA (Plan Do Check Act), znázorněných na Obr. 9. [25]



Obr. 9 ISMS PDCA [26]

Plánuj – Nejprve se získají informace o řešeném problému. Na daný problém se pak navrhne konkrétní řešení, které by mělo problém eliminovat.

Dělej – Vypracovaný plán je testován.

Kontroluj – Kontrola, jestli plán plní své cíle dle očekávání a jestli je daný problém skutečně vyřešen.

Jednej – V případě, že plán problém neeliminoval, hledá se příčina. Pokud byl problém úspěšně odstraněn, je nový plán zaveden do stávajícího systému. [27]

4.2 ISO 27002

Norma ISO 27002 obsahuje seznam opatření pro výběr v rámci zavádění ISMS vycházející z normy ISO 27001. Lze ji také využít při vytváření směrnic pro správu bezpečnosti informací pro dané organizace s ohledem na jejich konkrétní rizikovost. Bezpečnost informací je podpořena využíváním vhodných systémů, zvyků a postupů. Pro maximalizaci účinnosti zvolených pravidel je nutno postupovat dle metodiky PDCA. [28]

4.3 Vztah mezi řadou norem ISO 27000 a obsahem a formátem reportů z penetračních testů

Využití penetračních testů je možné pro kontrolu ISMS v rámci PDCA v části „kontroluj“ k ověření, jestli nově navržené řešení problému je dostačující. Nicméně řada norem ISO 27000 sama neříká, jak mají penetrační testy probíhat, ani jak má vypadat jejich výsledný report nebo co má obsahovat. Současně také pro splnění požadavků pro certifikaci podle normy ISO 27001 nejsou penetrační testy nutné. [29] [30]

5 DEFINICE POŽADAVKŮ NA APLIKACI

Jak již bylo zmíněno dříve, tato práce si klade za cíl zjednodušit generování reportů z penetračních testů. Vyvinutá aplikace by měla být schopna generovat reporty z penetračních testů prováděných v laboratoři penetračního testování „PT LAB“ na Univerzitě Tomáše Bati ve Zlíně. Report by ve výsledku měl obsahovat všechny sekce a náležitosti, které se vyskytují ve stávajících hlášeních vytvořených v „PT LAB“.

Cílem aplikace je tedy generovat hlášení, která jsou obsahově shodná s obsahem stávajících hlášení, ale s co možná největším zjednodušením práce pro uživatele. Uživatel nebude muset pro každé hlášení vytvářet nový dokument a kopírovat do něj sekce, nadpisy a další statické texty, které se ve všech reportech opakují. Měl by jen vyplnit zjištěné skutečnosti (dynamická data) a to co možná nejjednodušším/nejrychlejším způsobem. Důležitou vlastností aplikace je, aby bylo možné změnit jak statické texty, tak vzhled výsledného hlášení. Dalším způsobem jak zjednodušit práci pro uživatele, by mělo být načítání předzpracovaných dat ze souboru.

Dalším z požadavků na aplikaci je, ačkoliv má mít možnost měnit statické texty a vzhled, aby se stávající, již vygenerované reporty neměnily. Změny statických textů a vzhledu by měly ovlivnit jen další, nově vytvořené reporty. Současně by aplikace měla poskytovat funkcionalitu stávající reporty přegenerovat, aby dříve vytvořený report obsahoval nové změny statických textů či vzhledu.

Vzhledem k faktu, že penetrační testy se obvykle neprovádějí z operačního systému Microsoft Windows, by aplikace měla být multiplatformní. Měla by být schopna fungovat na operačním systému Microsoft Windows, macOS X i Linux.

Souhrn požadavků na aplikaci:

- Obsah a vzhled reportů dle stávajících reportů z „PT LAB“
- Využití šablon pro statická data
- Uživatelské rozhraní pro zadání dynamických dat
- Načítání dynamických dat ze souboru
- Neměnnost již vygenerovaných reportů s možností přegenerování
- Multiplatformní provoz (MS Windows, macOS X, Linux)

II. PRAKTICKÁ ČÁST

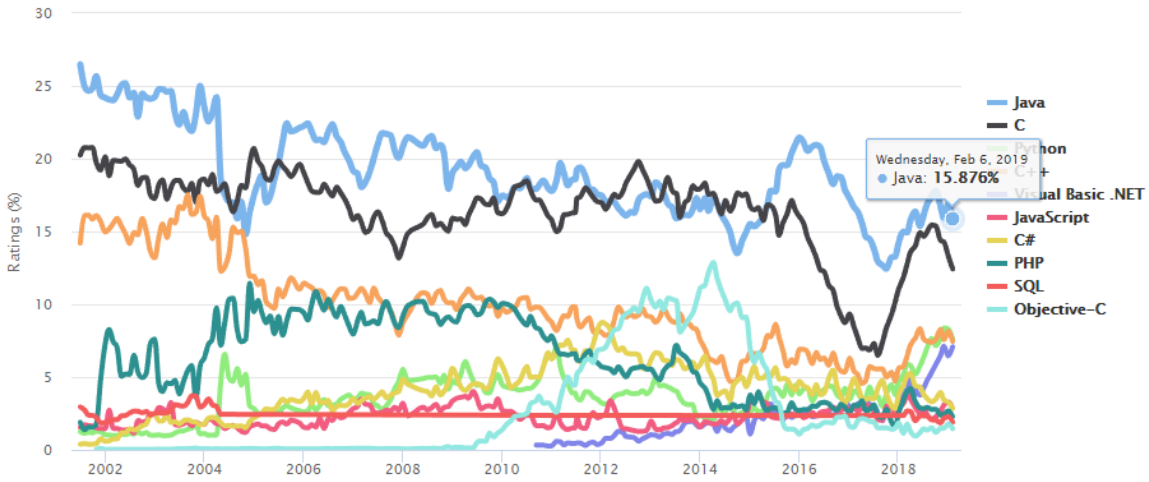
6 VÝBĚR VHODNÉ TECHNOLOGIE PRO TVORBU APLIKACE

Na základě požadavků zmíněných v kapitole 5. musí být zvolena multiplatformní technologie tak, aby bylo možné aplikaci spustit s MS Windows, Linux i macOS. Aplikace nevyužívá složitých algoritmů nebo komunikaci po síti, ale především slouží k zadávání dynamických dat od uživatele. Je tedy vhodné volit technologii, v které se dají dělat kvalitní uživatelská rozhraní. Níže je uveden přehled některých dostupných technologií.

6.1 Java swing

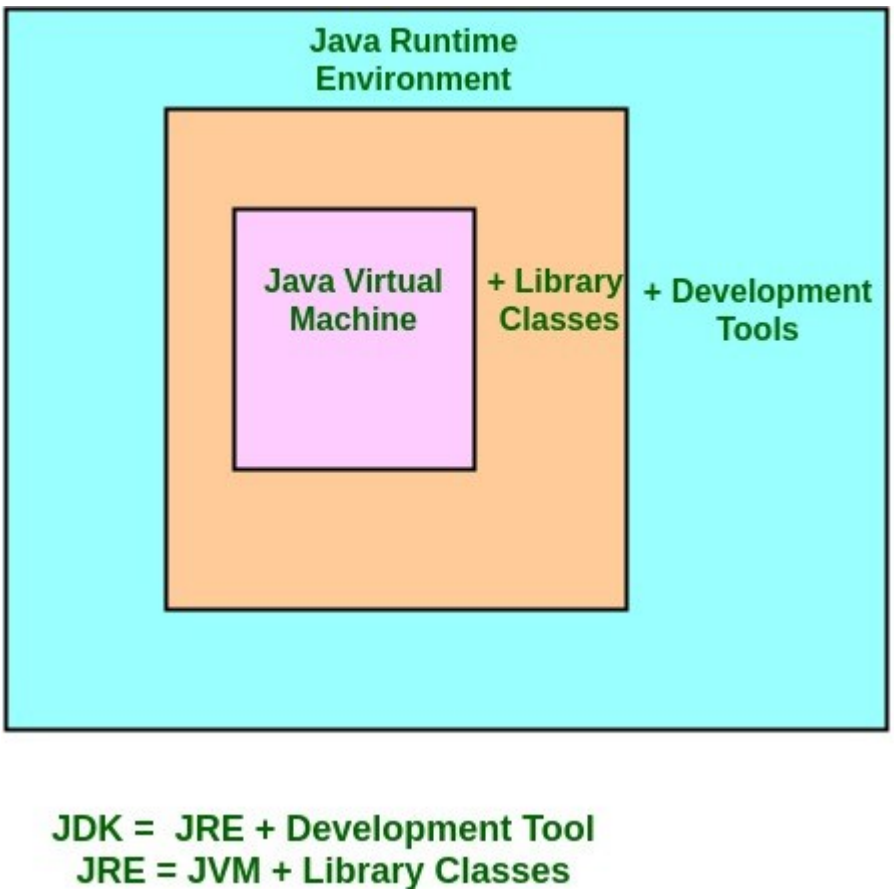
Obr. 10 Java logo [31]

Java je multiplatformní programovací jazyk, který vznikl v roce 1995 a byl vyvinut společností Sun Microsystems. V roce 2006 Sun Microsystems uvolnil zdrojové kódy Javy a od té doby je Java vyvíjena jako open source. Jde o nejpopulárnější programovací jazyk současnosti podle TIOBE indexu. Popularita některých programovacích jazyků dle TIOBE indexu je znázorněna na Obr. 11. [32] [33]



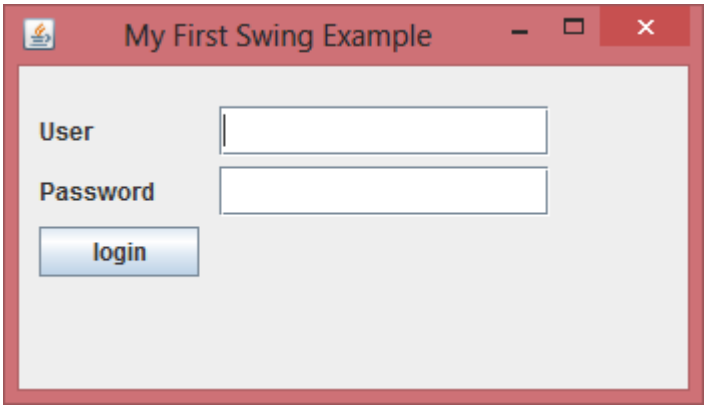
Obr. 11 Popularita programovacích jazyků [32]

Java je jazykem interpretovaným a potřebuje pro spuštění aplikace, aby na počítači byl nainstalován Java Runtime Enviroment (JRE), který obsahuje Java Virtual Machine (JVM), který interpretuje a spouští Java kód. Pro vývoj Java aplikací je zapotřebí mít nainstalovaný Java Development Kit (JDK). Rozdíly mezi JRE, JVM a JDK jsou znázorněny na Obr. 12. [31] [34]



Obr. 12 Rozdíl mezi JRE, JVM a JDK [34]

Swing je knihovna pro vytváření GUI aplikací v Javě z roku 1998 a nahradil tehdy stávající Abstract Window Toolkit (AWT). Uživatelská rozhraní vytvořena ve Swingu pak vypadají na všech operačních systémech stejně a nenativně. Typický vzhled Swing aplikace je vidět na Obr. 13 [35]



Obr. 13 Ukázka jednoduché Swing aplikace [36]

6.2 Framework Qt

Obr. 14 Qt logo [37]

Qt je multiplatformní framework pro vývoj desktopových, mobilních a embedded aplikací v C++. Mezi podporované platformy patří Linux, OS X, Windows, Android, iOS, BlackBerry a další.

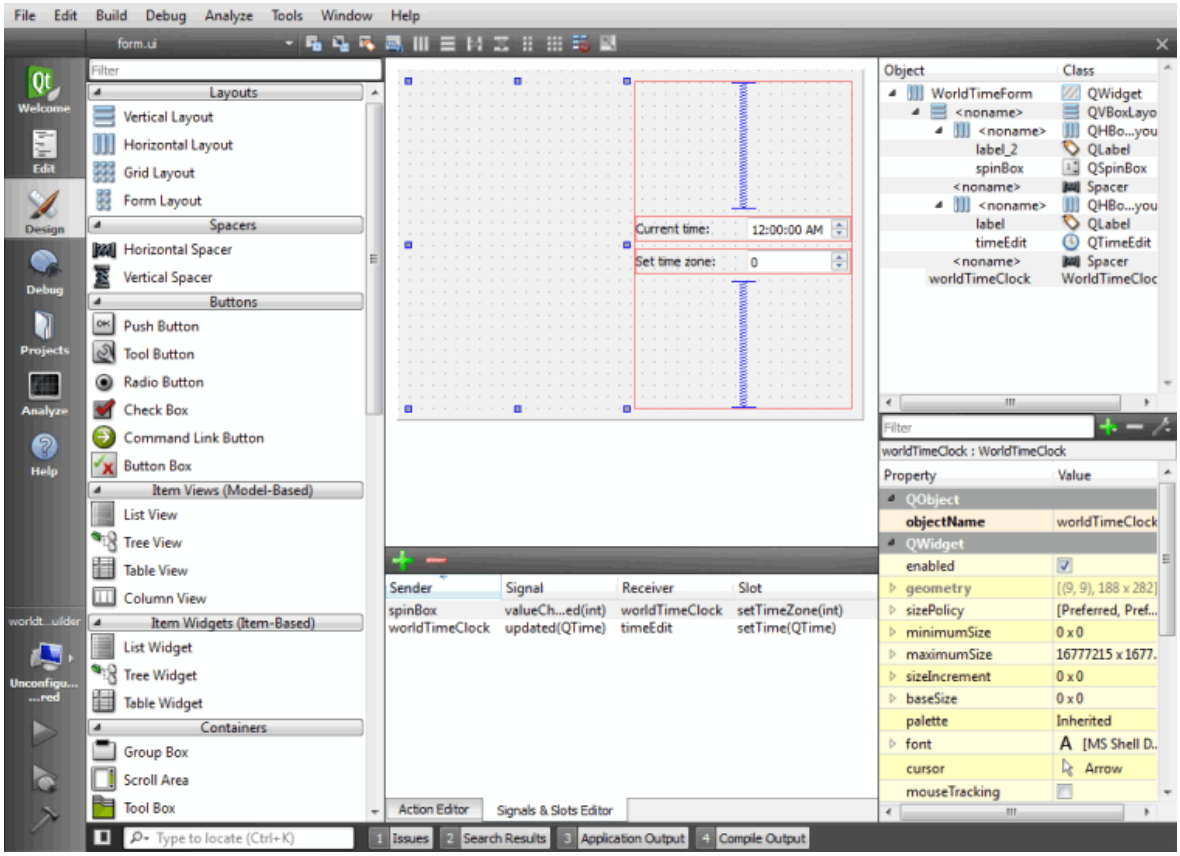
Kód, který se má kompilovat, je nejdříve předzpracován v MOC (Meta-Object Compiler), jehož výstupem je standardní C++, který se dá následně kompilovat standardními C++ kompilátory, například CMake. K překladu lze také využít překladač od Qt pojmenovaný qmake.

Pro vývoj s frameworkem Qt se dá využít například IDE Qt Creator, které lze nainstalovat na operační systém Linux, OS X a Windows. Obsahuje doplňování kódu, zvýrazňování syntaxe, debugger, profiler a také má podporu pro verzování. [38]

6.2.1 Qt Designer

Jedním ze způsobů, jak vytvářet v Qt Creatoru uživatelská rozhraní, je využít Qt Designer, který je vidět na Obr. 15. Umožňuje v grafickém rozhraní vytvářet uživatelská rozhraní bez nutnosti psát kód a přistupovat na jeho prvky pomocí signálů a slotů, které framework Qt

implementuje. Rozhraní se sice dá vytvářet bez nutnosti psaní kódu, ale jeho vlastnosti lze dynamicky pomocí kódu měnit. [37]



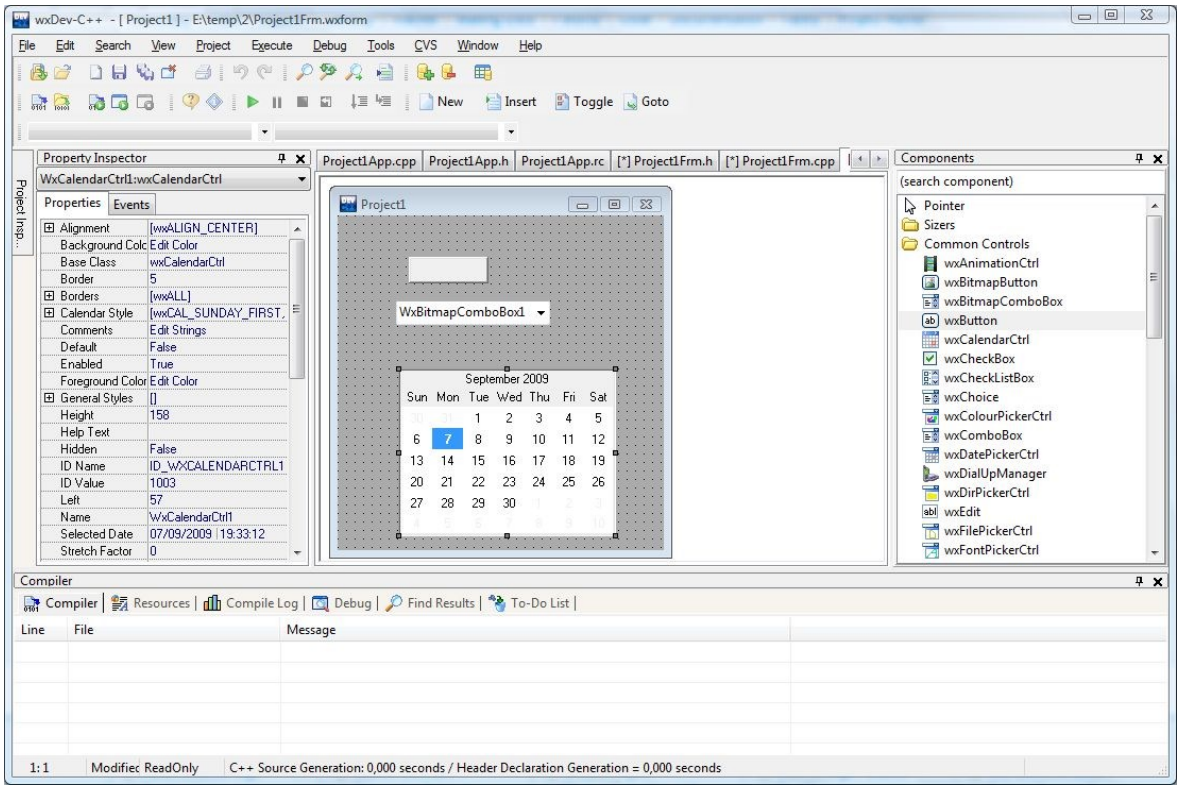
Obr. 15 Náhled na Qt designer [39]

6.3 wxWidgets

Obr. 16 wxWidgets logo [40]

WxWidgets je další knihovnou pro tvorbu multiplatformních grafických rozhraní aplikací psaných v C++. Využívá nativní grafické prvky každé platformy, aplikace tedy vypadají nativně. Rozhraní jsou tvořena především programově, ale dá se využít několika dialogových editorů pro tvorbu dialogů a panelů. Kromě C++ lze s wxWidgets použít také Python a Perl. Mezi podporované platformy wxWidgets patří Linux, Windows i macOS. [40]

Pro tvorbu uživatelských rozhraní za pomoci grafického editoru lze využít wxDev-C++ které je vidět na Obr. 17 [41] [42]



Obr. 17 Ukázka wxDev-C++ [42]

6.4 Electron

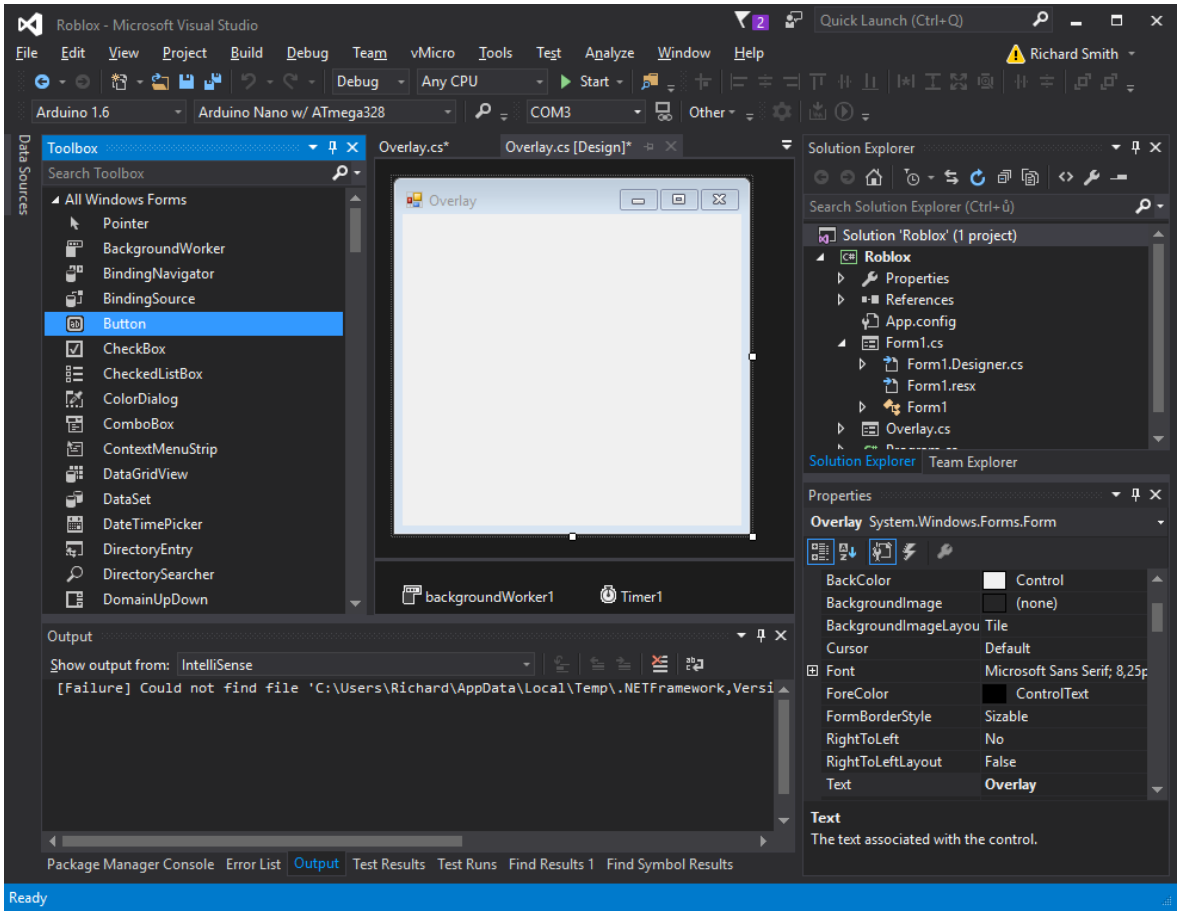
Obr. 18 Logo Electronu [43]

Electron je framework, se kterým lze vytvářet aplikace za pomoci HTML, CSS a JavaScriptu. Využívá Chromium (jádro prohlížeče Google Chrome) a Nojde.js. Aplikace vytvořené za pomoci frameworku Elektron, jsou víceméně webové aplikace, na které uživatel přistupuje přes prohlížeč. Kvůli tomu jsou aplikace vytvořené v elektronu náročnější na hardware, pomalejší a zabírají více místa. Nicméně díky využití HTML a CSS mohou být grafická rozhraní vzhledná, responzivní, robustní a aplikace se vytváří za pomoci jazyků, které jsou pro mnohé známé. Současně sebou ale využití technologie elektron přináší možná bezpečnostní rizika. [43] [44] [45]

6.5 Microsot .Net Windows Forms

Obr. 19. Logo frameworku Microsoft .Net [46]

Windows Forms je knihovna tříd ve frameworku Microsoft .NET Framework. Při použití IDE Visual Studio lze vytvářet uživatelská rozhraní Windows Forms v grafickém designeru, který je vidět na Obr. 20. V ovládacích prvcích využívá pro zpracování uživatelských vstupů „eventy“, na něž naváže programátor funkce, které se mají provést v případě, že je daný event vyvolán (obdobně jako signál a slot v Qt). Windows Forms obsahuje řadu předpřipravených ovládacích prvků, které může programátor hned využít a v případě, že by předpřipravené prvky nesplňovaly všechny požadavky, lze si vytvořit vlastní. K řazení prvků v uživatelském rozhraní lze využít layouty. [47] [48]



Obr. 20 Ukázka designeru v IDE Visual Studio

V současné době je Windows Forms pomalu nahrazován knihovnou WPF (Windows Presentation Foundation), ale je stále používán. WPF je také součástí frameworku Microsoft .NET a využívá k tvorbě GUI jazyk XAML. [49]

Microsoft .NET framework byl primárně určený pro platformu Windows, ale pro spuštění na jiných platformách lze využít „Mono“, což je framework vycházející z .NET frameworku, který je multiplatformní. Není ale 100% kompatibilní s .NET frameworkem, takže již existující aplikace je třeba portovat. [50] [51]

6.6 Shrnutí

Na základě funkčních a technických požadavků, byl ze zmíněných technologií zvolen framework Qt. Některé výhody Qt, v porovnání s ostatními zmíněnými technologiemi, znázorňuje Tabulka 1. Dalším důvodem pro volbu frameworku Qt je autorova zkušenost s tímto frameworkem.

Tabulka 1. Srovnání zmíněných technologií.

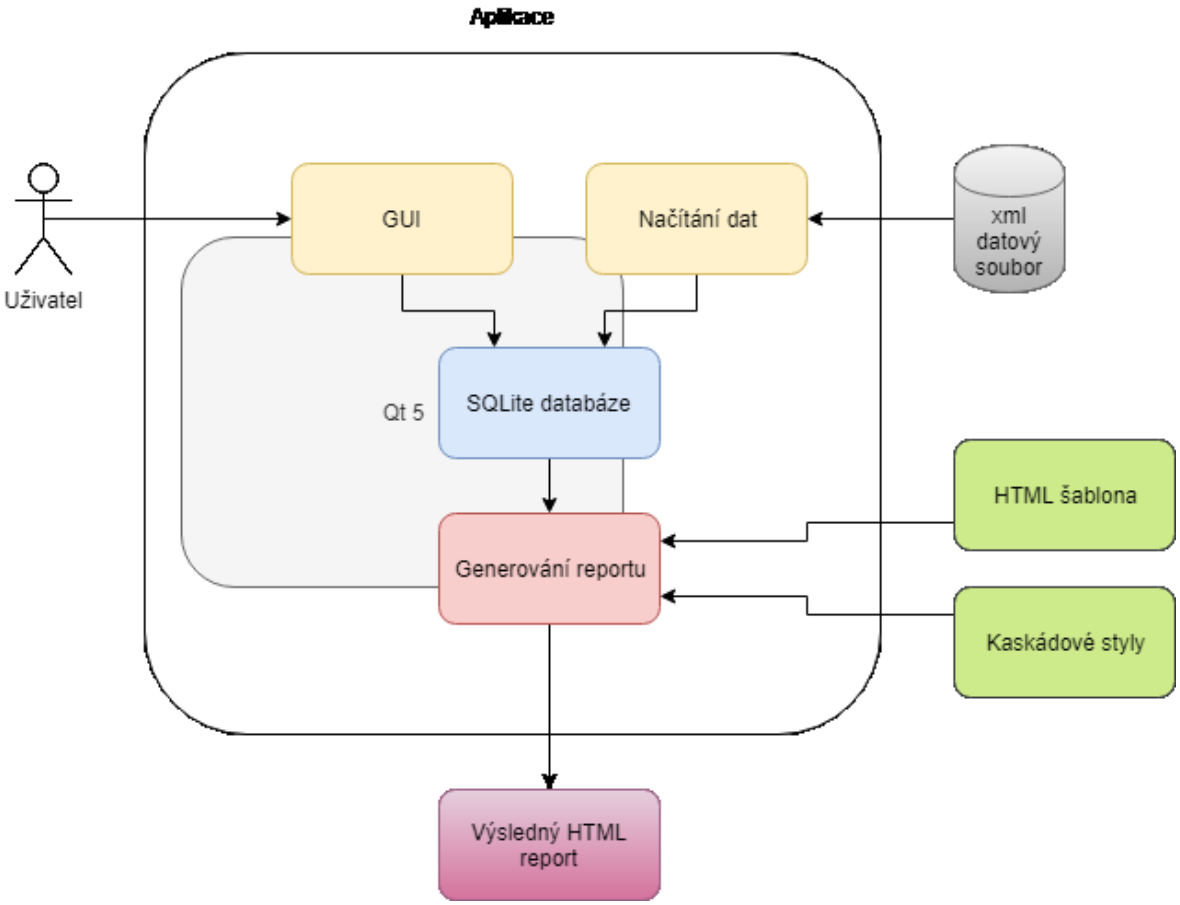
Název technologie	Nutnost dalšího SW	Integrovaný desig-ner GUI	Multiplaformnost	Nativní GUI prvky
Java	Ano	Dle zvoleného IDE	Ano	Ne
Qt	Ne	Ano	Ano	Ne
wxWidgets	Ne	Ne	Ano	Ano
Electron	Ne	Ne	Ano	Ne
MS .NET	Ano	Ano	Komplikovaná	Ne

7 NÁVRH APLIKACE

Jak je vidět na Obr. 21, struktura aplikace je rozdělena do několika částí, z nichž každá část zastává jinou funkci. Uživatel pomocí grafického rozhraní zadává dynamická data, která jsou ukládána do relační databáze SQLite. Dalším způsobem zadávání některých dynamických dat, je jejich načítání z XML souboru.

Aby bylo možné změnit vzhled nebo obsah výsledného hlášení, byl navržen systém HTML šablon, které obsahují klíčová slova. Při generování reportu jsou klíčová slova nahrazena dynamickými daty z databáze. Změnou obsahu šablony je tedy možné změnit obsah, pořadí prvků nebo jazyk výsledného reportu. Tím je splněn požadavek na možnost měnit statické texty ve výsledném hlášení.

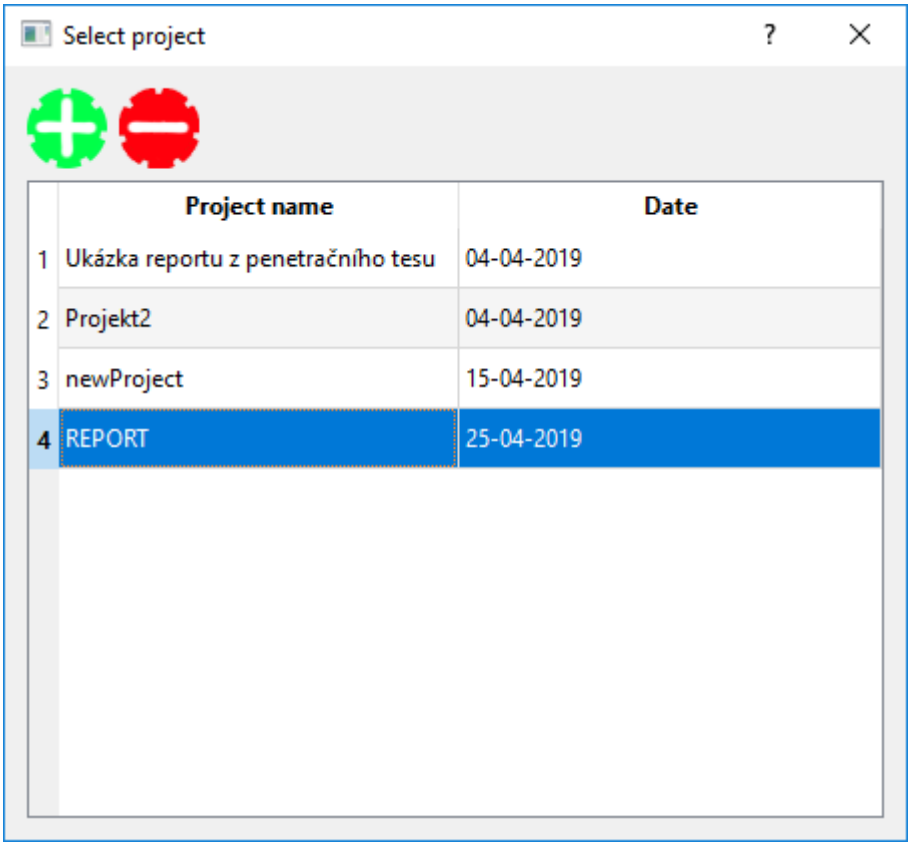
Za účelem splnění požadavku na možnost měnit vzhled výsledného reportu, jsou na HTML report aplikovány kaskádové styly (CSS), pomocí kterých je možné libovolně ovlivňovat vzhled jednotlivých elementů výsledného hlášení. K zajištění neměnnosti reportu, pokud by došlo ke změně souboru s kaskádovými styly, je obsah CSS souboru vložen na začátek reportu. Díky tomu vygenerovaný report už později nezmění svůj vzhled. Pokud by ale aktualizace vzhledu dle nového souboru kaskádových stylů byla žádoucí, je možnost report jednoduše přegenerovat s využitím nového CSS souboru.



Obr. 21 Struktura aplikace

7.1 Návrh uživatelského rozhraní

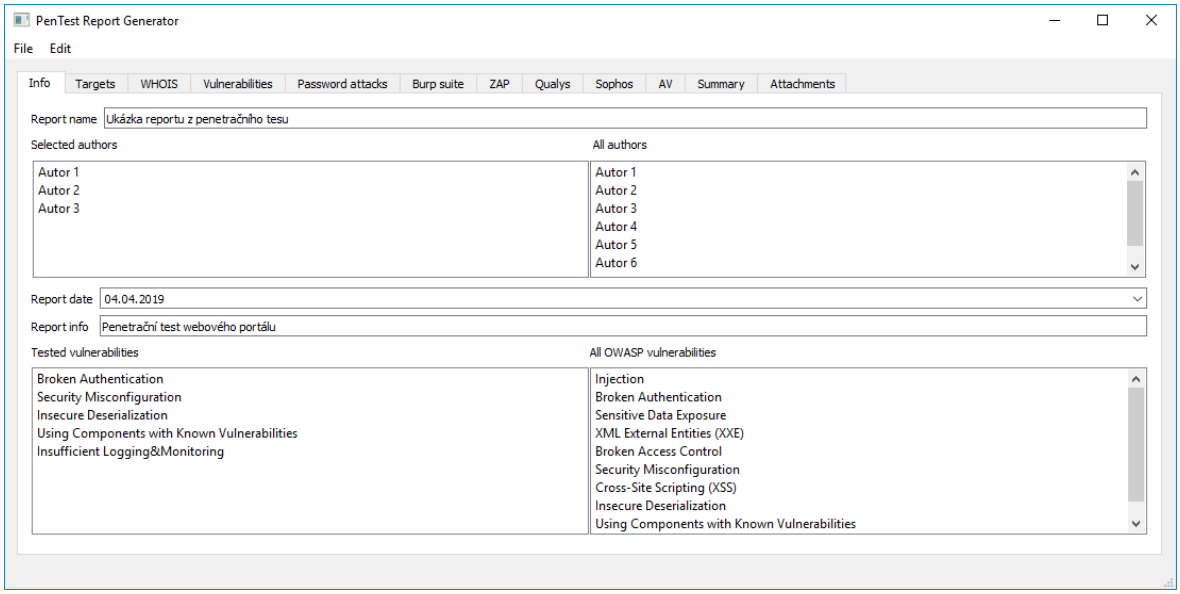
Po spuštění aplikace je nejdříve uživatel vyzván pomocí formuláře na Obr. 22 buď k výběru jednoho ze stávajících projektů dvojitým kliknutím na projekt, tvorbě nového projektu kliknutím na zelené tlačítko se symbolem „+“ nebo k odstranění projektu/ů kliknutím na červené tlačítko se symbolem „-„.



Obr. 22 Úvodní obrazovka s volbou projektu

Po volbě již vytvořeného projektu nebo vytvoření nového projektu se otevře hlavní okno aplikace, které je vidět na Obr. 23. To je rozděleno na 12 záložek, které odpovídají sekcím ve výsledném reportu.

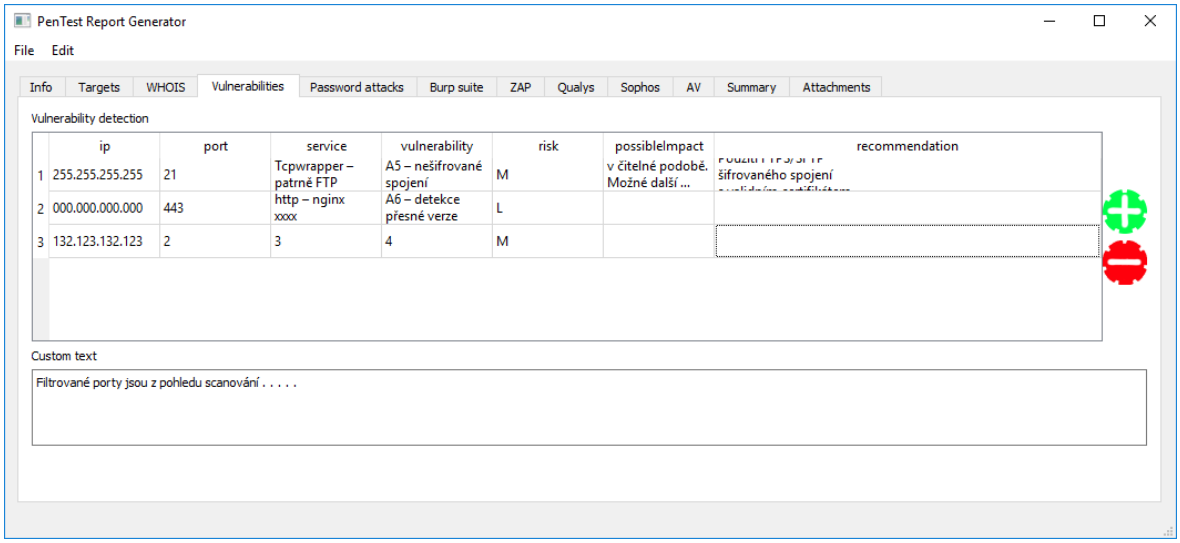
V první záložce „Info“ jsou obecné informace o projektu. Jeho název, datum, informace o reportu, lidé, kteří se podíleli na tvorbě tohoto reportu a seznam testovaných zranitelností.



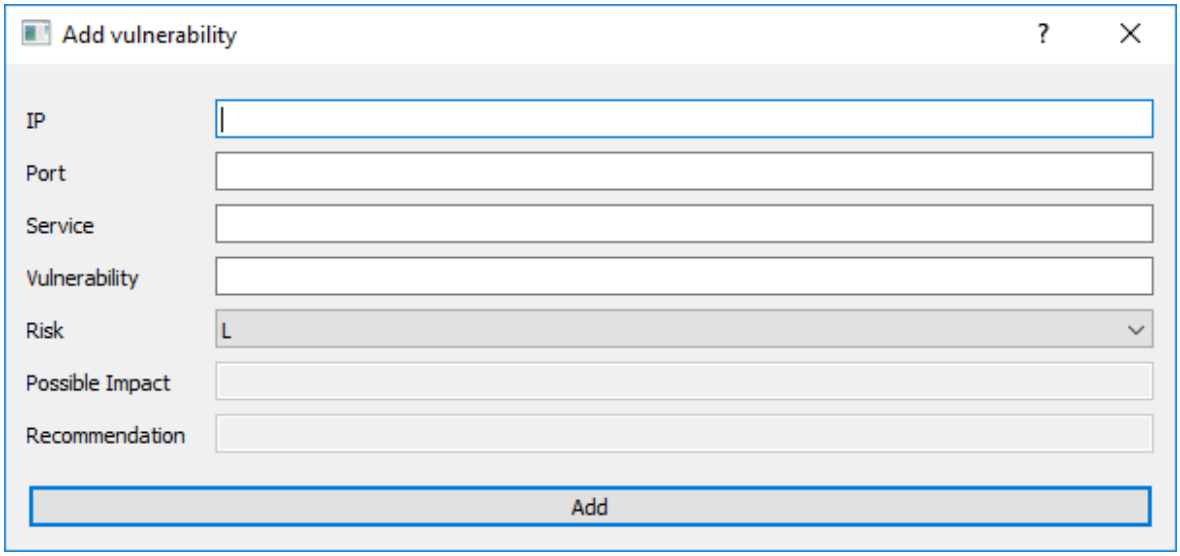
Obr. 23 Hlavní okno aplikace, záložka s informacemi o projektu

V další ukázce na Obr. 24 je náhled na záložku se seznamem zranitelností. Zde je tabulka detekovaných zranitelností. Další objevenou zranitelnost lze přidat kliknutím na zelené tlačítko se symbolem „+“. Tím se otevře dialogové okno, které je vidět na Obr. 25. To slouží pro přidání další nalezené zranitelnosti. Již přidané zranitelnosti lze měnit přímo v tabulce dvojitým kliknutím na položku, kterou chce uživatel měnit. Pro odebrání řádku z tabulky stačí označit řádek a kliknout na červené tlačítko se symbolem „-„.

Po tabulce se zranitelnostmi následuje pole pro zadání volitelného textu.

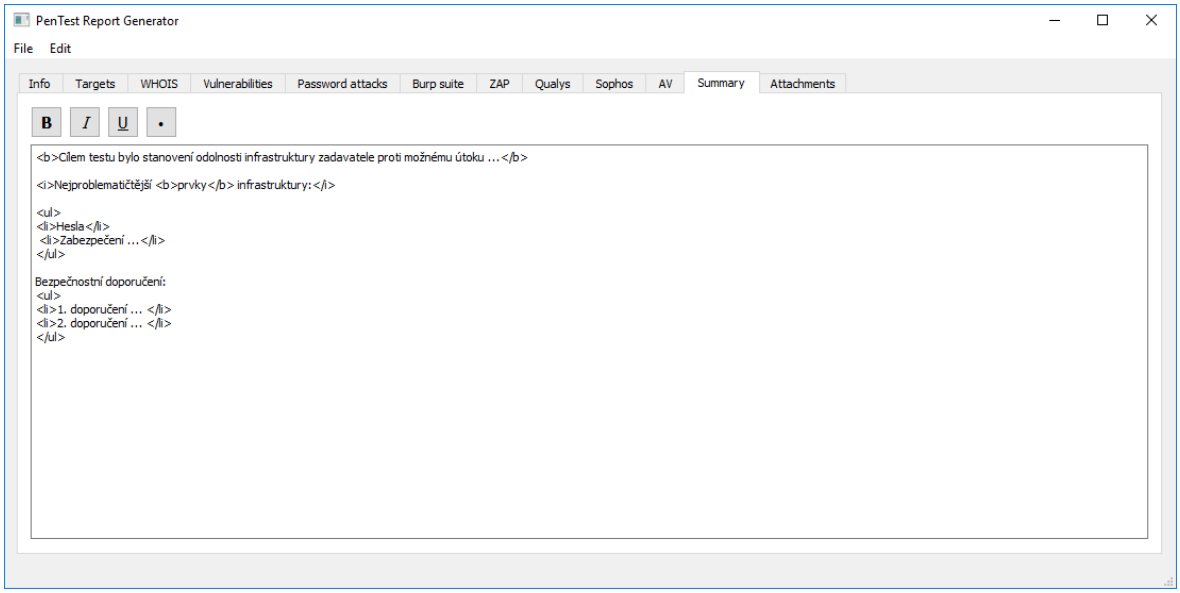


Obr. 24 Hlavní okno aplikace, záložka se seznamem zranitelností



Obr. 25 Dialogové okno pro přidání zranitelnosti

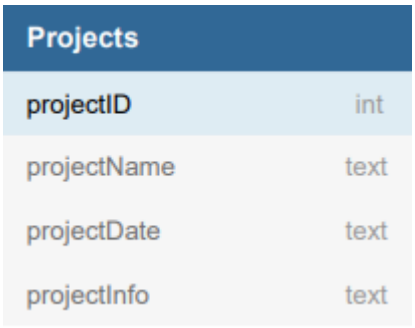
Obr. 26 zobrazuje záložku shrnutí. Shrnutí je realizováno pomocí pole pro psaní textu, které může využívat všechny standartní HTML tagy. Pro zjednodušení běžných úprav textu má program připravené tlačítka, která samy vloží označený text do příslušných HTML tagů. Konkrétně jsou zde tlačítka pro tučný text, kurzívu a podtržení textu. Poslední tlačítko vloží nečíslovaný seznam.



Obr. 26 Hlavní okno aplikace, záložka se shrnutím

7.2 Návrh databáze

V programu je využita relační databáze SQLite, jejíž hlavní tabulkou je tabulka projektů (reportů) na Obr. 27. Ta obsahuje ID projektu jako primární klíč, název projektu, datum vytvoření projektu a jeho informace.

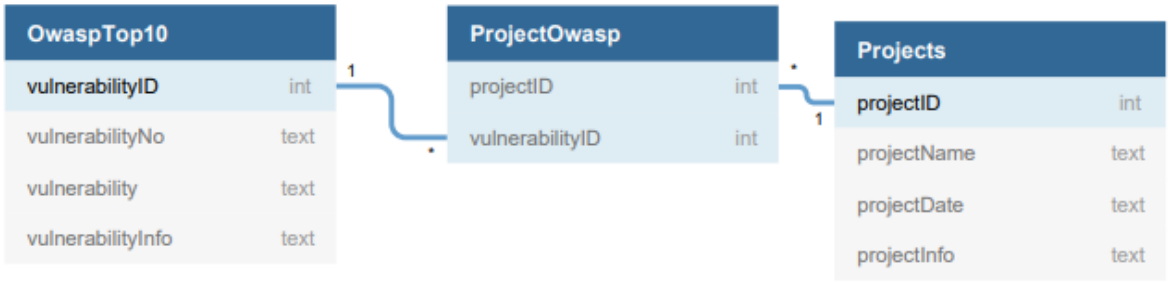


Obr. 27 Tabulka projektů

Na každém projektu se může podílet několik autorů a současně každý autor může pracovat na několika projektech. Jde tedy o vztah „many to many“ a je realizován pomocí spojovací tabulky, znázorněné na Obr. 28. Tabulka autorů obsahuje pouze ID a jméno autora.

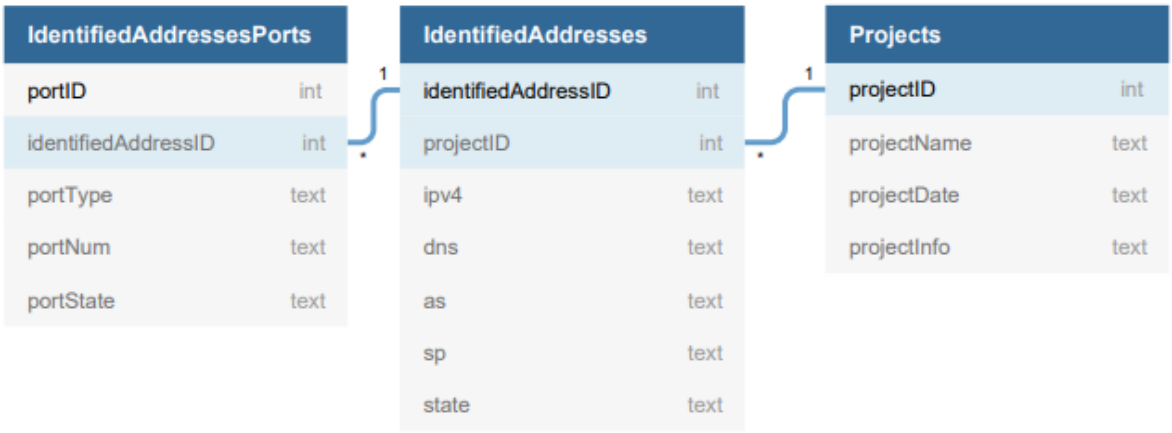
Obr. 28 Propojení tabulky autorů a tabulky projektů

V každém projektu je prošetřováno několik možných zranitelností a všechny vycházejí z OWASP TOP 10. V každém reportu mohou být testovány jiné zranitelnosti, jde tedy opět o relaci „many to many“ znázorněnou na Obr. 29. Každá OWASP zranitelnost v tabulce sebou nese své ID, označení, název a popis.



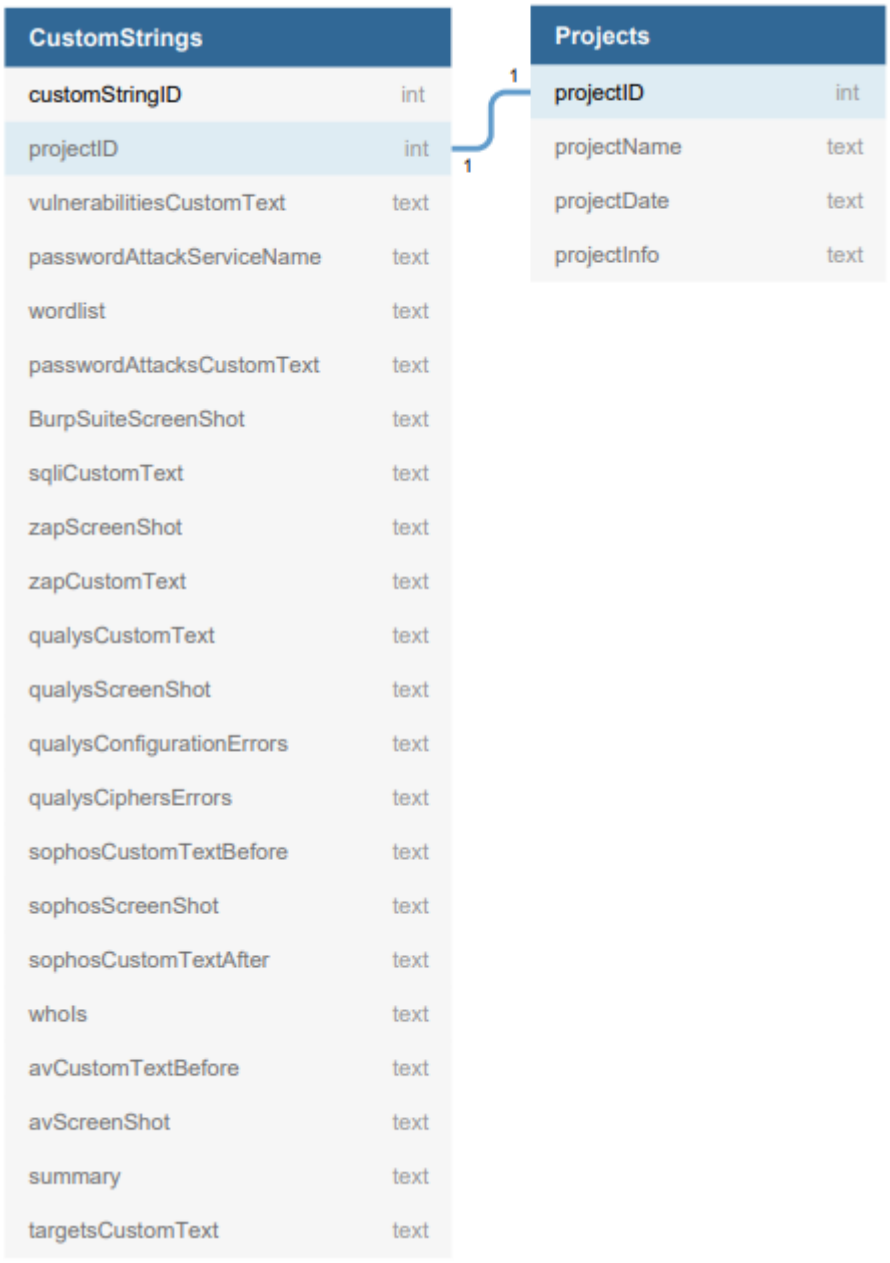
Obr. 29 Propojení tabulky OWASP TOP 10 s tabulkou projektů

V každém projektu je seznam cílů (IP adres), které se mají prověřit a po provedení skenování portů je ke každému cíli identifikováno několik portů. Tabulka se seznamem cílů (identifikovaných adres) je na Obr. 30 a nese své ID, ID projektu, ke kterému adresa patří, IPv4, DNS, AS, SP a stát, kde se nachází. Každý nalezený port obsahuje své ID, ID identifikované adresy, ke které náleží, typ portu (TPC nebo UDP), jeho číslo a stav. Každý projekt tedy může mít několik cílů a každý cíl může mít několik portů.



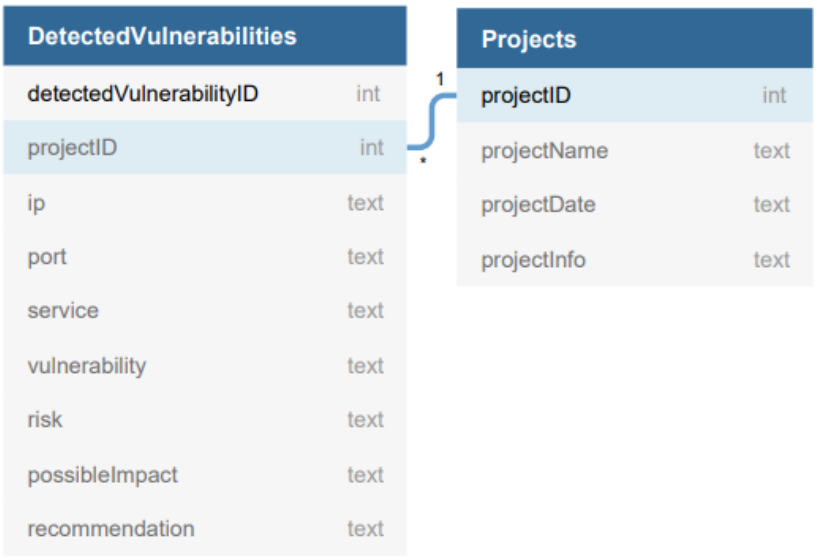
Obr. 30 Propojení tabulky se seznamem cílů a projektem

S každým reportem je vždy spojen právě jeden záznam v tabulce se seznamem volitelných textů. Tabulka s volitelnými texty obsahuje texty nebo názvy obrázků, jež uživatel v aplikaci zadá a je dopředu znám jejich počet a účel. Její vztah k tabulce projektů je tedy „one to one“ a je znázorněn na Obr. 31.



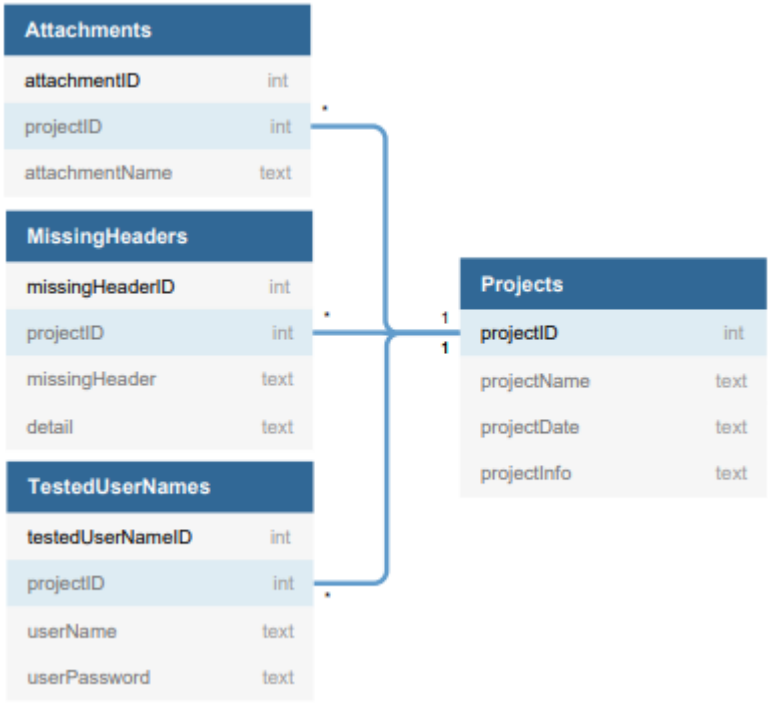
Obr. 31 Tabulka s volitelnými texty

Další tabulkou v databázi, znázorněnou na Obr. 32, je tabulka obsahující nalezené zranitelnosti. Ta obsahuje ID nalezené zranitelnosti, ID projektu, ke kterému náleží, IP na které byla zranitelnost nalezena, port, služba, název zranitelnosti, riziko které přináší, možný dopad a doporučení k potlačení rizika. V každém projektu může být nalezeno několik zranitelností, jde tedy o vztah „one to many“.



Obr. 32 Seznam nalezených zranitelností

Zbývající tabulky v databázi jsou na Obr. 33 Jde o seznam příloh, chybějící hlavičky a testovaná uživatelská jména, Všechny mají vůči tabulce s projekty vztah „one to many“ a jejich obsah je patrný z následujícího obrázku.



Obr. 33 Tabulky seznam příloh, chybějící hlavičky a testovaná uživatelská jména

Kompletní schéma databáze je dostupné v příloze P I: SCHÉMA DATABÁZE

7.3 Návrh šablony

Šablona obsahuje např. nadpisy, statické texty a první řádky tabulek. V místech, kde mají být dynamické texty, tedy data z databáze aplikace, jsou využita pomocná klíčová slova začínající symbolem „$“. Ta budou následně aplikací nahrazena dynamickými daty. Díky tomu je splněn požadavek na možnost měnit obsah a vzhled reportu pomocí šablon. Lze tak měnit statické texty, případně jazyk textů, vkládat obrázky, odkazy atd.

V ukázce na Obr. 34 je nejprve vidět statický text mezi HTML tagy „<p>“, který bude vždy ve výsledném reportu, za předpokladu, že bude využita tato šablona. Po statickém textu následuje tabulka s ID „crackedUsers“. ID může být využito k individuálnímu stylování vzhledu tabulky v souboru s kaskádovými styly. Tabulka má nadepsaný první řádek, který obsahuje sloupce s texty „Uživatelské jméno“ a „Heslo“. Další řádky tabulky budou vloženy dynamicky aplikací z její databáze nahrazením klíčového slova „$crackedUsers“. Po tabulce následuje klíčové slovo „$passwordAttacksCustomText“, které bude nahrazeno textem z databáze, který si uživatel může napsat ke každému reportu jiný. Výsledek po nahrazení dynamickými daty a aplikaci CSS je vidět na Obr. 35.

```
<p>
  Test nalezl následující validní kombinace uživatelského jména a hesla.
</p>

<table id="crackedUsers">
  <tr>
    <th>Uživatelské jméno</th>
    <th>Heslo</th>
  </tr>
  $crackedUsers
</table>

<p>
  $passwordAttacksCustomText
</p>
```

Obr. 34 Ukázka statického a dynamického textu a tabulky v HTML šabloně

Test nalezl následující validní kombinace uživatelského jména a hesla.

Uživatelské jméno	Heslo
admin	admin
user3	passw0rd

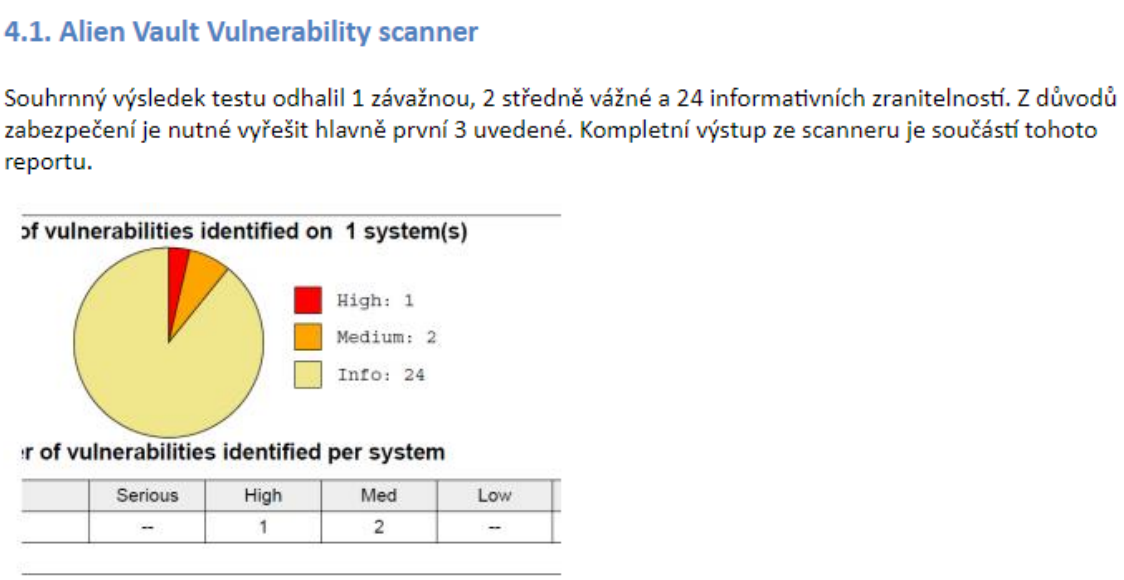
Test s pomocí slovníku rockyou trval přes 12h a přinesl 2 rozluštitelná hesla. Celkem bylo vyzkoušeno 1 500 000 kombinací (při rychlosti cca. 340/min).

Obr. 35 Ukázka statického a dynamického textu s daty z databáze po aplikaci CSS

Další ukázka na Obr. 36 zobrazuje, jak jsou v šabloně realizovány nadpisy a obrázky. Celá sekce je mezi tagy „<p>“ a nadpis je realizován standartním HTML tagem „<h3>“ pro nadpis úrovně 3. Po nadpisu následuje klíčové slovo pro dynamický text „$avCustomTextBefore“, který bude nahrazen textem z databáze. Nakonec je pomocí HTML tagu „<img>“ vložen obrázek, jehož zdroj je dynamický text „$avScreenShot“. Cesta k obrázku bude tedy doplněna aplikací z databáze. Ukázka z výsledného reportu je na Obr. 37.

```
<p id="av">
  <br>
  <h3>Alien Vault Vulnerability scanner</h3>
  <p>$avCustomTextBefore</p>

  <img src="$avScreenShot">
</p>
```

Obr. 36 Ukázka nadpisu a obrázku v HTML šabloně



Obr. 37 Ukázka nadpisu a obrázku s daty z databáze po aplikaci CSS

Kompletní seznam použitých zástupných symbolů začínajících „$“ je dostupný v příloze P II: SEZNAM POMOCNÝCH SYMBOLŮ V HTML ŠABLONĚ.

ZÁVĚR

Hlavním cílem této diplomové práce bylo navrhnout a implementovat software pro generování hlášení z penetračních testů, které jsou prováděny v laboratoři PT LAB na Univerzitě Tomáše Bati ve Zlíně. Důležitou vlastností reportů generovaných tímto softwarem měla být shodnost obsahu a vzhledu se současnými reporty z laboratoře PT LAB.

V teoretické částí bylo nejdříve nastíněno, co je penetrační testování a k čemu ho lze využívat. Byly vysvětleny jednotlivé části penetračního testu a popsán obsah jeho výstupního hlášení. Byla zde popsána hlavní myšlenka projektu OWASP, některé jeho části a jeho spojitost s penetračními testy. Následovala krátká rešerše několika nástrojů, které mohou být využívány během provádění penetračních testů, bylo vysvětleno k čemu jednotlivé zmíněné nástroje lze použít a na obrázcích bylo ukázáno, jak vypadají jejich výstupy. Dále bylo popsáno k čemu slouží rodina norem ISO 27000, co je ISMS a na jakém principu funguje. Na konci teoretické části proběhla definice požadavků na aplikaci, na základě kterých pak byla samotná aplikace vyvíjena.

Na začátku praktické části byla provedena rešerše některých dostupných technologií, které je možné využít při implementaci aplikace a na základě jejich rozdílných vlastností byl zvolen framework Qt. S ohledem na požadavky na aplikaci bylo rozhodnuto, že aplikace bude pro zadávání dat využívat uživatelské rozhraní a načítání dat z XML souborů. Data budou ukládána do SQLite databáze a výsledné reporty budou generovány v HTML formátu, který lze jednoduše převést do PDF pomocí tisku ve virtuální tiskárně v prohlížeči. Pro možnost editování statického obsahu reportů byl navržen systém HTML šablon, který je kombinován se soubory kaskádových stylů pro editaci vzhledu.

V poslední kapitole byly ukázány některé části výsledného uživatelského rozhraní a bylo nastíněno, jakým způsobem se s aplikací pracuje. Následně bylo podrobně popsáno schéma databáze, která je aplikací využívána k ukládání dat a nakonec byly uvedeny dvě ukázky z šablony pro generování reportů.

8 SEZNAM POUŽITÉ LITERATURY

[1] WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking*. San Francisco: No Starch Press, 2014. ISBN 15-932-7564-1.

[2] SELECKÝ, Matúš. *Penetrační testy a exploitace: a hands-on introduction to hacking*. Brno: Computer Press, 2012. ISBN 978-80-251-3752-9.

[3] Penetracni testy eticky hacking socialni inzenyrstvi. *Comguard* [online]. b.r. [cit. 2019-02-15]. Dostupné z: https://www.comguard.cz/sluzby/sluzby-v-oblasti-informacni-ict-bezpecnosti/penetracni-testy-eticky-hacking-socialni-inzenyrstvi/

[4] ČERMÁK, Miroslav. Bezpečnost webových aplikací: rychlé seznámení s OWASP. *Cleverandsmart.cz* [online]. 2012 [cit. 2019-04-28]. Dostupné z: https://www.cleverandsmart.cz/bezpecnost-webovych-aplikaci-rychle-seznameni-s-owasp/

[5] OWASP: za webové aplikace bezpečnější. *Root.cz* [online]. 2010 [cit. 2019-04-28]. Dostupné z: https://www.root.cz/clanky/owasp-za-webove-aplikace-bezpecnejsi/

[6] DROLET, Michelle. What is OWASP, and why it matters for AppSec. *Csoonline.com* [online]. 2017 [cit. 2019-04-28]. Dostupné z: https://www.csoonline.com/article/3171079/what-is-owasp-and-why-it-matters-for-appsec.html

[7] *The OWASP™ Foundation* [online]. the OWASP Foundation, 2019 [cit. 2019-04-28]. Dostupné z: https://www.owasp.org/index.php/Main_Page

[8] OWASP Top Ten Project. *Owasp.org* [online]. the OWASP Foundation, 2019 [cit. 2019-04-28]. Dostupné z: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[9] OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks. *Owasp.org* [online]. The OWASP Foundation, 2017 [cit. 2019-04-28]. Dostupné z: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

[10] OWASP ASVS. *Ripstech.com* [online]. RIPS Technologies GmbH, b.r. [cit. 2019-04-28]. Dostupné z: https://www.ripstech.com/product/compliance/owasp-asvs/

[11] OWASP Zed Attack Proxy Project. *Owasp.org* [online]. the OWASP Foundation, 2019 [cit. 2019-04-28]. Dostupné z: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[12] Introduction. *Docs.kali.org* [online]. Offensive Security, b.r. [cit. 2019-05-11]. Dostupné z: https://docs.kali.org/category/introduction

[13] The Parrot System. *Parrotsec.org* [online]. Parrot Team, b.r. [cit. 2019-05-11]. Dostupné z: https://www.parrotsec.org/docs/intro/what-is-parrot/

[14] Nmap. *Nmap.org* [online]. b.r. [cit. 2019-05-11]. Dostupné z: https://nmap.org/

[15] Nmap. *Wikipedia.org* [online]. 2017 [cit. 2019-05-12]. Dostupné z: https://cs.wikipedia.org/wiki/Nmap

[16] Introduction. *Wireshark.org* [online]. b.r. [cit. 2019-05-11]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

[17] Sqlmap. *Sqlmap.org* [online]. b.r. [cit. 2019-05-11]. Dostupné z: http://sqlmap.org/

[18] Introduction. *Aircrack-ng.org* [online]. 2018 [cit. 2019-05-11]. Dostupné z: https://www.aircrack-ng.org/doku.php

[19] Getting Started with the Aircrack-Ng Suite of Wi-Fi Hacking Tools. *Null-byte.wonderhowto.com* [online]. NULL BYTE, 2017 [cit. 2019-05-12]. Dostupné z: https://null-byte.wonderhowto.com/how-to/hack-wi-fi-getting-started-with-aircrack-ng-suite-wi-fi-hacking-tools-0147893/

[20] Iso27000 family. *Itgovernance* [online]. IT Governance Ltd, b.r. [cit. 2019-04-17]. Dostupné z: https://www.itgovernance.co.uk/iso27000-family

[21] ISO/IEC 27000 family - Information security management systems. *Iso* [online]. 1214 Vernier, Geneva Switzerland: ISO, b.r. [cit. 2019-04-17]. Dostupné z: https://www.iso.org/isoiec-27001-information-security.html

[22] ISMS: normy ISO 27001 a ISO 2700. *Rac* [online]. Risk Analysis Consultants, b.r. [cit. 2019-04-17]. Dostupné z: http://www.rac.cz/rac/homepage.nsf/CZ/BS7799

[23] ISMS (Information Security Management System). *Qcom.cz* [online]. b.r. [cit. 2019-04-29]. Dostupné z: http://www.qcom.cz/systemy-rizeni/isms/

[24] ISO 27001. *Whatis.techtarget* [online]. TechTarget, 2009 [cit. 2019-04-17]. Dostupné z: https://whatis.techtarget.com/definition/ISO-27001

[25] ISMS. *Wikisofia.cz* [online]. b.r. [cit. 2019-04-29]. Dostupné z: https://wikisofia.cz/wiki/ISMS

[26] ISO/IEC 27001:2013. *Rac.cz* [online]. Risk Analysis Consultants, b.r. [cit. 2019-04-29]. Dostupné z: http://www.rac.cz/rac/homepage.nsf/CZ/27001

[27] STŘELEC, Jiří. PDCA cyklus. *Vlastnicesta.cz* [online]. 2012 [cit. 2019-04-29]. Dostupné z: https://www.vlastnicesta.cz/metody/pdca-cyklus-1/

[28] ČSN EN ISO/IEC 27002. *Shop.normy.biz* [online]. b.r. [cit. 2019-04-29]. Dostupné z: https://shop.normy.biz/detail/95679

[29] Does ISO 27001 Require Penetration Testing?. *Krypsys* [online]. KRYPSYS Limited, b.r. [cit. 2019-04-17]. Dostupné z: https://www.krypsys.com/iso-27001/iso-27001-require-penetration-testing/

[30] H. ALLEN, Julia. "Plan, Do, Check, Act". *Us-cert.gov* [online]. Carnegie Mellon University, 2013 [cit. 2019-04-29]. Dostupné z: https://www.us-cert.gov/bsi/articles/best-practices/deployment-and-operations/plan-do-check-act

[31] JOHARI, Aayushi. What is java. *Edureka* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://www.edureka.co/blog/what-is-java/

[32] Tiobe index. *Tiobe* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://www.tiobe.com/tiobe-index/

[33] Java timeline. *Oracle* [online]. b.r. [cit. 2019-02-21]. Dostupné z: http://oracle.com.edgesuite.net/timeline/java/

[34] Differences between JDK, JRE and JVM. *Geeksforgeeks* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://www.geeksforgeeks.org/differences-jdk-jre-jvm/

[35] WORKMAN, Kevin. Swing. *Happycoding* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://happycoding.io/tutorials/java/swing

[36] SINGH, CHAITANYA. Java-swing-tutorial. *Beginnersbook* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://beginnersbook.com/2015/07/java-swing-tutorial/

[37] Qtdesigner manual. *Doc* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://doc.qt.io/qt-5/qtdesigner-manual.html

[38] About Qt. *Wiki.qt* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://wiki.qt.io/About_Qt

[39] Creator using qt designer. *Doc.qt* [online]. b.r. [cit. 2019-02-21]. Dostupné z: https://doc.qt.io/qtcreator/creator-using-qt-designer.html

[40] About wxwidgets. *Wxwidgets* [online]. b.r. [cit. 2019-02-25]. Dostupné z: https://www.wxwidgets.org/about/

[41] Wxwidgets general. *Wxwidgets* [online]. b.r. [cit. 2019-02-25]. Dostupné z: https://www.wxwidgets.org/docs/faq/general

[42] Main page. *Wxdsgn* [online]. b.r. [cit. 2019-02-25]. Dostupné z: http://wxdsgn.sourceforge.net/

[43] About electron. *Electronjs* [online]. b.r. [cit. 2019-02-25]. Dostupné z: https://electronjs.org/docs/tutorial/about

[44] Electron nodeintegration bypass. *Trustwave* [online]. b.r. [cit. 2019-02-25]. Dostupné z: https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/cve-2018-1000136-electron-nodeintegration-bypass/

[45] NALEGAVE, Shardul. Electron | Pros And Cons. *Medium.com* [online]. 2018 [cit. 2019-05-04]. Dostupné z: https://medium.com/@nalegaveshardul40/electron-pros-and-cons-8f58fd6313d5

[46] .net framework 4. *Findesolutions* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://web.findesolutions.com/net-framework-4/

[47] Windows forms overview. *Docs.microsoft* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview

[48] ČÁPKA, David. C sharp tutorial windows forms okenni aplikace uvod. *Itnetwork* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://www.itnetwork.cz/csharp/formulare/winforms/c-sharp-tutorial-windows-forms-okenni-aplikace-uvod

[49] C sharp tutorial wpf uvod a prvni formularova aplikace. *Itnetwork* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://www.itnetwork.cz/csharp/formulare/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace

[50] About mono. *Mono-project* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://www.mono-project.com/docs/about-mono

[51] JOHNATHAN, Pobst. Porting winforms applications. *Mono-project* [online]. b.r. [cit. 2019-02-28]. Dostupné z: https://www.mono-project.com/docs/gui/winforms/porting-winforms-applications/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GUI	Graphical User Interface
JRE	Java Runtime Enviroment
JVM	Java Virtual Machine
JDK	Java Develipment Kit
MOC	Meta-Object Compiler
IDE	Integrated development environment
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
ISMS	Information security management system
OWASP	The Open Web Application Security Project
ASVS	Application Security Verification Standard
ZAP	Zed Attack Proxy
PDCA	Plan Do Check Act
GUI	Graphical User Interface
JRE	Java Runtime Enviroment
GPL	General Public License
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
PSK	Pre-shared key
SQL	Structured Query Language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SCHÉMA DATABÁZE

PŘÍLOHA P II: SEZNAM POMOCNÝCH SYMBOLŮ V HTML ŠABLONĚ

$reportName

$reportAuthors

$reportDate

$reportInfo

$testedVulnerabilities

$identifiedTargets

$targetsCustomText

$whoIs

$AdressesAndPorts

$identifiedVulnerabilities

$MHCVulnerabilities

$vulnerabilitiesCustomText

$passwordAttackServiceName

$wordlist

$testedUserNames

$crackedUsers

$passwordAttacksCustomText

$BurpSuiteScreenShot

$sqliCustomText

$zapScreenShot

$zapCustomText

$qualysCustomText

$qualysScreenShot

$qualysConfigurationErrors

$qualysCiphersErrors

$sophosCustomTextBefore

$sophosScreenShot

$sophosCustomTextAfter

$missingHeaders

$avCustomTextBefore

$avScreenShot

$summary